Linux x64 下的万能 Gadget

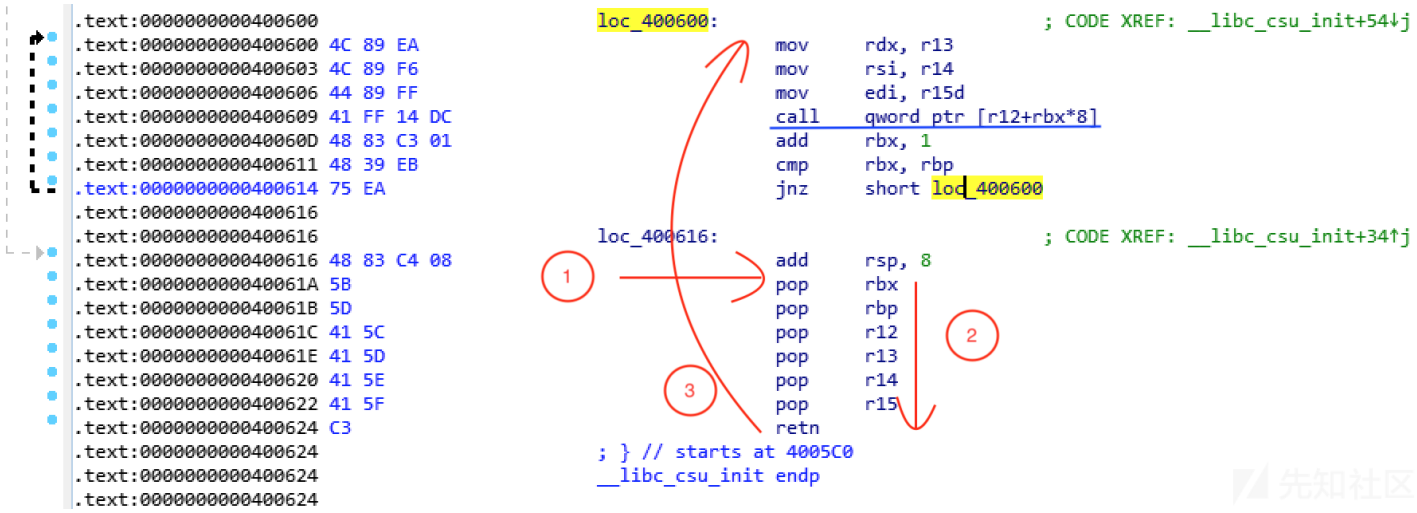
香依香偎 / 2019-07-09 09:11:00 / 浏览数 5640 安全技术 二进制安全 顶(1) 踩(0)

Linux x64 下的万能 Gadget

来源：香依香偎@闻道解惑

一、通用 Gadget

蒸米在《一步一步学ROP之linux_x64篇》中提到，在栈溢出的场景下，只要 x64 程序中调用了 libc.so，就会自带一个很好用的通用Gadget：__libc_csu_init()。



如图，先从 0x40061A 开始执行，将 rbx/rbp/r12/r13/r14/r15 这六个寄存器全部布置好，再 ret 到 0x400600 ，继续布置 rdx/rsi/rdi，最后通过 call qword ptr[r12+rbx*8] 执行目标函数。

这个通用 Gadget 好用的地方在于，不仅可以通过函数地址的指针（通常会用记录库函数真实地址的 got
表项）来控制目标函数，还可以控制目标函数的最多三个入参（rdi/rsi/rdx）的值。此外，只要设置 rbp=rbx+1而且栈空间足够，这个 Gadget
可以一直循环调用下去。

计算一下一次调用需要的空间。

可以看出，这个 Gadget 需要布置六个寄存器（rbx/rbp/r12/r13/r14/r15）加一个 ret 返回地址，x64 下至少需要 56 个字节的栈空间。如果再算上将 rip
指令跳转进来（0x40061A）的一个 ret 地址，那就是 64 字节的栈空间。

栈的布置如下：

栈	说明
0x40061A	ret到pop rbx
0	pop rbx
1	pop rbp
目标函数的got地址	pop r12 指向目标函数的指针
第三个入参	pop r13 mov rdx,r13
第二个入参	pop r14 mov rsi,r14
第一个入参	pop r15 mov edi,r15d
0x400600	ret到mov rdx,r13

二、隐藏 Gadget：pop rdi,ret

其实，这个通用 Gadget 里，还隐藏了两个更简单的 Gadget。

```
.text:000000000040061E 41 5D                pop     r13
.text:0000000000400620 41 5E                pop     r14
.text:0000000000400622 41 5F                pop     r15
.text:0000000000400624 C3                   retn
.text:0000000000400624                  ; } // starts at 4005C0
.text:0000000000400624                  __libc_csu_init endp
```

将地址 0x400622 上 pop r15,ret 的三字节指令(0x41 0x5F 0xC3)拆散看，会发现后两个字节组成了一组新的指令 pop rdi,ret。

```
.text:0000000000400620 41 5E                pop     r14
.text:0000000000400620                  ; ---------------------------------------------------------
.text:0000000000400622 41                   db 41h
.text:0000000000400623                  ; ---------------------------------------------------------
.text:0000000000400623 5F                   pop     rdi
.text:0000000000400624 C3                   retn
```

这已经足够完成单入参的函数调用了。

通常栈溢出之后，需要进行如下两步：

1、通过类似 puts(puts) 的方式，泄漏libc库函数的地址，从而通过偏移计算出 system 函数和 "/bin/sh" 字符串的地址

2、执行 sytem("bin/sh") 获得系统 shell

发现没有？大多数情况我们只需要一个入参的函数调用， __libc_csu_init() 函数最后的这个 pop rdi,ret 可以完美实现上述两个步骤。

空间上，只需要 24 个字节（一个 QWORD 存放 ret 进来的地址，两个 QWORD 作为入参和被调用函数地址）的溢出空间就足够啦。

栈的空间布置如下：

栈	说明
0x400623	ret到pop rdi
第一个入参	pop rdi
目标函数地址	ret

那，如果需要调用两个入参的函数呢，这个 Gadget 也行么？是的。

三、隐藏 Gadget：pop rsi,...,ret

将地址 0x400620 上 pop r14 的两字节指令（0x41 0x5E）拆散，会发现后一个字节是单字节指令 pop rsi，可以用来控制第二个入参。

```
.text:000000000040061E
.text:0000000000400620 41                  ; ----------------------------
.text:0000000000400621                     db 41h
.text:0000000000400621 5E                  ; ----------------------------
.text:0000000000400622 41 5F               pop     rsi
.text:0000000000400624 C3                  pop     r15
.text:0000000000400624                     retn
                                           ; } // starts at 4005C0
```

和前述的地址 0x400623 上的指令 pop rdi,ret组合起来，就可以完成两个入参的函数调用。

```
.text:0000000000400621                     ; ----------------------------
.text:0000000000400621 5E                  pop     rsi
.text:0000000000400622 41 5F               pop     r15
.text:0000000000400624 C3                  retn
.text:0000000000400624                     ; } // starts at 4005C0

.text:0000000000400623                     ; ----------------------------
.text:0000000000400623 5F                  pop     rdi
.text:0000000000400624 C3                  retn
.text:0000000000400624                     ; } // starts at 4005C0
```

只需要将栈布置如下就可以啦。

栈	说明
0x400621	ret到pop rsi
第二个入参	pop rsi
anything	pop r15
0x400623	ret到pop rdi
第一个入参	pop rdi
函数地址	ret到目标函数

四、总结

1、只要Linux x64 的程序中调用了 libc.so，程序中就会自带一个很好用的通用Gadget：__libc_csu_init()。

2、__libc_csu_init() 的 0x400600 到 0x400624 其中包含了 pop rdi、pop rsi、pop rdx、ret
等指令，通过巧妙的组合可以实现调用任意单参数、双参数、三参数的函数，从而顺利泄漏libc函数地址并且获取系统 shell。

3、__libc_csu_init() 不只是一个通用 Gadget，完全就是"万能 Gadget"！

参考阅读：

[1] 蒸米《一步一步学ROP之linux_x86篇》：https://zhuanlan.zhihu.com/p/23487280

[2] 蒸米《一步一步学ROP之linux_x64篇》：https://zhuanlan.zhihu.com/p/23537552

上一篇：PHDays的IDS Bypass... 下一篇：浅析php-fpm的攻击方式

1. 0 条回复
   - 动动手指，沙发就是你的了！