Software Security – ROP

Jian Guo,Zhilong Wang

版本：1.0

日期：March 24, 2021

摘　　要

本文档是南京大学软件安全课程实验的说明文档。此文档介绍如何进行基本的 ROP 攻击。ROP 的全称为 Return-oriented programming，这是一种高级的内存攻击技术，可以用来绕过现代操作系统的各种通用防御（比如栈不可执行等）。关于 ROP 的基础理论，我们推荐读者阅读文章 [1]：

1　实验环境

1.1　实验环境搭建

实验环境搭建、VirtualBox 的使用请参考 Buffer Overflow 实验手册。

1.2　关闭平台保护

本实验需要开启栈不可执行 (Non-Executable Stack) 保护，但是需要关闭地址空间随机化 (Address Sapce Randomization) 和栈保护（Stack Guard），具体的关闭过程请参考 Buffer Overflow 实验手册。

2　实验目标

本实验要求通过目标程序中的 buffer overflow 漏洞，利用 ROP 攻击方式调用 unlink 系统调用达到删除当前文件夹下的名为 data 文件目的，然后调用 exit 系统调用正常返回。

3　实验流程

3.1　溢出返回地址

利用 Buffer Overflow 实验手册介绍的方法确定需要溢出的返回地址的位置。

3.2　研究实验所需 ROP gadgets

我们利用系统调用实现预期的攻击效果，通过 ROP 链实现系统调用的具体方法如下：

- 设置`%eax` 寄存器为系统调用号；
- 寄存器`%ebx`、`%ecx`、`%edx`、`%esi` 和`%edi` 依次存放需要的参数。本次实验两个系统调用均只需一个参数。即设置%ebx 为需要的参数，

- 执行 int $0x80；

`unlink` 系统调用号为 0xa，需要一个 `char *` 指针参数，指向需要删除的文件的文件名；`exit` 系统调用号为 0x1，需要一个 int 型参数 0。选取 gadget 的出发点就是这些 gadget 要完成上述两个系统调用。其它系统调用的使用方法可参考：http://syscalls.kernelgrok.com/

3.3 搜索可用 ROP gadgets

通常的程序都需要链接 libc 库，libc 库也非常大，一般而言，我们可以搜索到足以完成上述系统调用的 gadget。使用下列指令确定文件使用的动态库位置：

```
$ ldd ./exec
```

我们可利用下列方法搜索需要的 gadgets：

3.3.1 手动搜索

利用 IDA 反汇编工具来确定你选择使用的 gadget 在库中的地址。确定库文件位置后用 IDA 打开，反汇编 libc-***.so，切换到 IDA 的 Hex View-A 视图，根据 gadget 的机器码搜索相应的 gadget。

以搜索 gadget：`inc %eax; ret;` 为例：

- `inc %eax` 的机器码是 40，`ret` 的机器码是 C3；
- Alt-t 快捷键打开搜索框，搜索 40 C3；
- 搜索到的 gadget 地址为 26F52，记录此地址即为 gadget 相对地址；
- gadget 相对地址加上 libc 库的基地址即为 gadget 在内存中的地址；

3.3.2 利用工具搜索

ROP 查找和构造工具有：ROPgadget，angrop 等，请查考链接学习此类工具的使用：ROPgadget，angrop

注意： libc 库中没有以 ret 指令结尾的 int 0x80 指令，因此我们在实验的目标代码中放置了此类型指令供构造 ROP gadget。我们提供了两个 int 0x80 指令，但是他们都具有副作用，建议大家都尝试一下如何消除两者的副作用（考点）。

3.4 构造 ROP 攻击链

在上述步骤中，确定要溢出的返回地址的位置以及需要用到的 gadgets 在库文件中的位置，我们利用3.3章节中搜索到的 gadget 的地址和某些 gadget 可能用到的一些数据构造输入溢出返回地址以及其紧邻的地址区域。例如，使用如下两个 gadget 将`%eax` 设置为 0 时：

- gadget1：`pop %eax; ret;`。地址为 0xAAAAAAAA；
- gadget2：`inc %eax; ret;`。地址为 0xBBBBBBBB；

payload 文件中应该填充 $\backslash xAA, \backslash xAA, \backslash xAA, \backslash xAA, \backslash xFF, \backslash xFF, \backslash xFF, \backslash xFF, \backslash xBB, \backslash xBB, \backslash xBB, \backslash xBB$。这些填充字节，将按照从低地址到高地址的方向写入栈中。根据你所选择的 gadget 排列其顺序，建议画出堆栈示意图来辅助理解 gadget 的执行。

参考文献

[1] SHACHAM H, et al. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86).[C]//ACM conference on Computer and communications security. [S.l.]: New York,, 2007: 552-561.