深入理解计算机系统(CSAPP) 实验详解：AttackLab

原创

Jackson1997__ 于 2020-11-23 17:40:35 发布 2480 收藏 20
分类专栏： CSAPP 文章标签： c++

> 相比于boomLab我个人更加喜欢这个attackLab

更新历史

- `20201123` 开始更新

本文介绍的是CSAPP书籍中的第三个lab: Attack lab。通过这个lab我们能够更加清楚和深入的了解到缓冲区溢出的隐患，以及如何利用缓冲区溢出这个漏洞对现有程序进行控制流劫持，执行非法程序代码，和对程序进行攻击以及破坏。

知识预热

1. 阅读《深入理解计算机系统》的3.10.2~3.10.5
2. 仔细阅读Attack lab的writeup
3. `cgdb` 和 `objdump` 等调试和分析能力

介绍完上面必须的知识之外，我们就开始吧！

实验构成

本实验一共分为两个实验，第一个是没有任何保护的程序 `ctarget`，第二个是加了保护的程序 `rtarget` 使你更加难以做一些坏事□。下面我们简单介绍这两个程序。

ctarget
可以执行栈里的代码
rtarget
最主要的特点就是你**不可以栈中执行代码**，只能通过用跳转地址的形式去到该程序中已有的目标代码处去实现你的攻击

攻击可被实现的原理

C语言中对于数组的引用不进行任何边界检查，而且局部变量和状态信息（如保存的寄存器值和返回地址）都存放在栈中。当对越界的数组元素的写操作时，则会破坏存储在栈中的状态信息。一种常见的破坏就是缓冲区溢出。就是在栈中分配某个字符数组保存一个字符串，但是字符串的长度超出了为数组分配的空间。

目录结构

大致浏览一下整个lab的目录，一共6个文件：

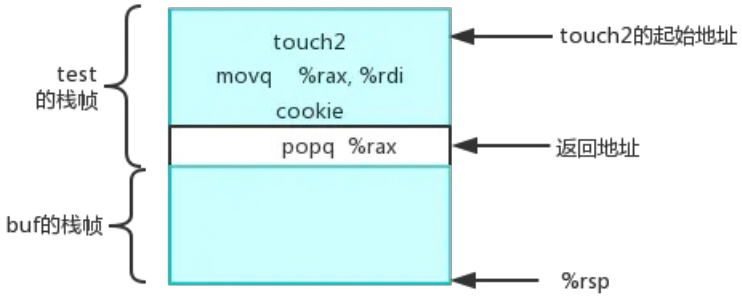
- `cookie.txt` 一个8位的16进制数，作为攻击的特殊标志符
- `farm.c` 在ROP攻击中作为gadgets的产生源（这个不理它也行，因为ROP攻击中是它助攻，靠它编译出来的汇编我们才实现攻击的）
- `ctarget` 代码注入攻击的目标文件
- `rtarget` ROP攻击的目标文件
- `hex2row` 将16进制数转化为攻击字符，因为有些字符在屏幕上面无法输入，所以输入该字符的16进制数，自动转化为该字符

代码注入攻击

这里看这位大佬的吧，写的很清楚，不再重复造轮子

ROP的理解

这里再补充一些我对ROP的理解



主要过程

先复习一下`return`的动作，在`return`时会将当前`%rsp`指向的值当作地址弹出，使得`%rip`程序计数器指到该地址的指令处。我们以此来作为攻击的手段，因为我们没有向栈中注入代码，我们是直接跳转到某个地址处的。

需要注意的地方

指令是指令，栈是栈不要搞混，不管我们指令如何跳，只要不对`%rsp`进行操作，我们栈指针就是不会改变，`return`会间接改变栈指针，因为它有两个动作（**先弹出地址给`%rip`，然后在给`%rsp`加`8`**）。

明白了上面说的，我们就可以开始讲ROP的过程了

指令	索引
touch2	4
movq %rax,%rdi	3
cookie	2
popq %rax	1

1. 上面图，返回地址处`index1`是最后`getbuf`函数执行到`return`时指向的位置。
2. 这个返回地址刚刚好就是一段指令的地址，它的指令就是`popq $raz`。还记得我们上面说的么，`return`先弹出地址给`%rip`，然后再给`%rsp`加`8`，刚刚好，我们的栈指针就指向了`cookie`这个地址`index2`。
3. 接着指令`popq $raz`执行，它会弹出当前栈的值，同时给栈指针又加8（index3）
4. 紧接着它又`return`了，刚刚好这个栈值`index3`又是一个指令的地址，所以`return`指令又将指令地址弹出给`%rip`然后将栈指针加`8`，然后就来到了`index4`
5. 随后执行指令，接着又`return`，恰好该栈值处又是一个地址，即`touch2`的地址，这个时候`%rip`就直接跳转到该指令地址处了。**但是要注意的是栈指针不会改变**。

到这里就结束了，祝各位变得更强□