Chapter 7 操作系统安全

沙之洲 2020012408

1 请描述栈溢出攻击和堆溢出攻击的基本原理。

栈溢出攻击是指攻击者越界访问并修改栈帧当中的返回地址，以达到控制进程目的的攻击。具体来说，攻击者通过越界写入数据，覆盖掉栈帧中保存的返回值地址，让 EIP 寄存器在函数返回的时候，指向恶意代码段从而实现进程的控制流劫持。

堆溢出攻击指的是攻击者通过越界访问并篡改堆管理的数据结构，进而实现恶意内存读写的攻击。通过恶意读写，可以进一步实现进程控制流的劫持。

2 请简述面向返回地址编程(ROP)和全局偏置表劫持攻击(GOT Hijacking)的原理，并分析他们能否绕过以下三种内存防御机制，并简述原因：a. W^X (Write XOR eXecution) b. ASLR (Address Space Layout Randomization) c. Stack Canary

面向返回地址编程基于栈溢出攻击。具体来说，是将返回地址设置为代码段中的合法指令，组合现存指令修改寄存器，劫持进程控制流。本质上，ROP 利用进程内存空间中现存的指令，编写了一个恶意程序，劫持了进程控制流。

对防御机制的绕过：

- ROP 利用的是代码段中的代码实现恶意行为，而代码段本身具有执行权限，W^X 机制失效。
- ASLR 只会对动态库基地址，堆和栈的基地址进行随机初始化，而不会对代码段产生影响，所以 ASLR 机制失效。
- 由于 ROP 仍然需要通过越界写修改返回值地址，所以仍然需要暴力破解 Stack Canary，因此 Stack Canary 不能被 ROP 绕过。

全局偏置表劫持是恶意篡改 GOT 表项，是进程调用攻击者指定的库函数，实现控制流劫持。具体来说，GOT Hijacking 是通过栈溢出或者堆溢出实现 GOT 表项的修改的。

- 因为装载共享库函数的页必须有可执行权限，同时 GOT 表位于数据段，数据段可读可写，因此 W^X 机制失效
- 因为 ASLR 无法随机初始化代码段的位置，攻击者仍然可以通过 PLT 表恶意读取 GOT 表项，然后得到动态库当中函数的地址，因此 ASLR 机制失效
- 如果 GOT Hijacking 是通过 ROP 的方式实现表项的篡改，那么 Stack Canary 机制可以一定程度上防御 GOT Hijacking 的攻击。否则，因为 GOT Hijacking 修改的是数据段中的表项而不是函数运行栈附近的表项，Stack Canary 机制将会失效。