Chapter 7 操作系统安全

赵晨阳 2020012363

请描述栈溢出攻击和堆溢出攻击的基本原理

栈溢出攻击和堆溢出攻击都是常见的缓冲区溢出攻击，可以利用程序中缓冲区溢出的漏洞，执行恶意代码，篡改程序的执行流程或窃取敏感信息。

栈溢出攻击的基本原理是：当程序使用栈空间存储局部变量、函数参数、返回地址等信息时，如果输入的数据超过了栈空间的边界，就会覆盖栈中的其他信息，包括返回地址等重要信息。攻击者可以构造特定的输入数据，来覆盖返回地址，从而使程序跳转到攻击者精心构造的代码区域执行，从而达到攻击的目的。

堆溢出攻击的基本原理是：当程序使用堆空间分配动态内存时，如果输入的数据超过了堆空间的边界，就会覆盖堆中的其他信息。攻击者可以通过构造特定的输入数据，来覆盖动态内存中存储的信息，从而达到执行恶意代码的目的。

栈溢出攻击和堆溢出攻击的区别在于，栈溢出攻击针对的是栈空间，而堆溢出攻击针对的是堆空间。栈空间通常用于存储函数调用过程中的数据，而堆空间通常用于存储动态分配的数据。在程序运行过程中，栈空间和堆空间通常都是不断变化的，攻击者可以通过构造特定的输入数据，来利用缓冲区溢出漏洞，篡改程序的执行流程或窃取敏感信息。

为了防止栈溢出攻击和堆溢出攻击，需要采取相应的防御措施，例如使用编译器提供的栈保护、使用内存分配器提供的堆保护、对输入数据进行严格的检查和过滤等措施。同时，在编写程序时，应尽可能避免使用不安全的函数、不进行边界检查的函数等，以减少攻击者利用缓冲区溢出漏洞的机会。

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

1. W^X (Write XOR eXecution)
2. ASLR (Address Space Layout Randomization)
3. Stack Canary

面向返回地址编程（ROP）和全局偏置表劫持攻击（GOT Hijacking）是常见的内存攻击技术。

ROP攻击的基本原理是：利用程序中的已有代码，构造一系列的 gadget（即短小的程序片段），并利用缓冲区溢出等漏洞将这些 gadget 组合起来，形成一段完整的恶意代码。攻击者通过篡改程序的返回地址，将程序执行流程跳转到这段恶意代码中，从而达到攻击的目的。

GOT Hijacking 攻击的基本原理是：为了使进程可以找到内存中的动态链接库，系统需要维护位于数据段的全局偏置表（Global Offset Table, GOT) 和位于代码段的程序链接表（Procedure Linkage Table, PLT)。攻击者可以恶意算改 GOT 表项，使进程调用攻击者指定的库函数，实现控制流劫持。

W^X 防御机制（Write XOR eXecution）是一种内存保护机制，用于防止内存区域既被写入数据又被执行代码。ASLR（Address Space Layout Randomization）是一种内存随机化机制，用于在程序运行时随机分配内存地址，以防止攻击者准确地预测内存地址。Stack Canary 是一种栈保护机制，用于检测栈溢出攻击，即在栈中插入一个随机数，一旦检测到栈溢出，就会触发异常。

ROP 绕过防御机制

1. ROP 利用的是代码段中的代码实现恶意行为，不需要在内存中注入新的代码或数据。代码段本身具有执行权限，W^X 机制失效。
2. 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 机制无法防御 ROP。
3. Stack Canary 机制可以防止 ROP 攻击，因为攻击者无法直接篡改栈中的 Stack Canary 值。

GOT Hijacking 绕开防御机制

GOT Hijacking 通过栈溢出或者堆溢出实对现 GOT 表项的修改。

- 装载共享库函数的页必须有可执行权限，且 GOT 表位于数据段，数据段可读可写，因此 W^X 机制失效。
- 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，得到动态库当中函数的地址，因此 ASLR 机制失效。
- 如前文所述，Stack Canary 机制主要用于检测栈溢出攻击。若修改 GOT 表项的操作以 ROP 实现，则 StackCanary 可以一定程度上防御全局偏置表劫持攻击。