CH7操作系统安全

- 为什么存在安全问题 P5-8
  - 1. 现代操作系统是规模庞大的软件系统，现代操作系统是**系统之系统**，各个模块之间的依赖关系复杂
  - 2. 其二，现代操作系统的设计以性能为最主要目标，而非安全性
- 攻击前提 P9
  - 攻击者可以构造任意输入并接受受害进程输入校验；
  - 攻击者无法**直接**读写系统下进程的内存，无法**直接**干预处理器上指令的执行
- 攻击目标 P10
- 操作系统基础攻击方案
  - Linux 内核为每一个进程维护一个**独立的**线性逻辑地址空间，以便于实现进程间内存的相互隔离
  - 这一线性逻辑地址空间被分为**用户空间**和**内核空间**；用户态下仅可访问用户空间，系统调用提供接口以访问内核空间；内核态下亦无法访问用户空间
  - 1. 文本段：进程的可执行二进制源代码
  - 2. 数据段：初始化了的静态变量和全局变量
  - 3. BSS 段 ： 未初始化的静态变量和全局变量
  - 4. 堆区 ： 由程序申请释放
  - 5. 内存映射段：映射共享内存和动态链接库
  - 6. 栈区 ： 包含了函数调用信息和局部变量
  - 栈区内存的作用 P19
  - 密切相关的寄存器 P20
  - 栈区溢出攻击 P30
    - 1. 返回至溢出数据 P32
    - 2. 返回至库函数 P35
  - 总结 P37
  - 基础堆区攻击 P38
  - 堆溢出攻击 P44
    - 1. P45 直接覆盖malloc_chunk首部为无意义内容，在堆管理器处理管理元数据时将造成崩溃
    - 2. P46~49 构造堆块重叠：堆块重叠是一种病态堆区内存分配状态，同一堆区逻辑地址被堆管理器多次分配。如右图所示，造成Heap Overlap之后攻击者可以通过写入一个堆块，实现对另一堆块内容的写入；同理，读出被覆盖堆块当中的数据。
    - 3. P50 更加复杂的堆区溢出攻击：利用堆管理其他机制。例如，基于unlink机制的堆区溢出攻击
    - 4. Use-After-Free 51是进程由于实现上的错误，使用已被释放的堆区内存。被free函数释放的堆块内存仍然可以被继续使用，当再次调用malloc分配内存时，会同时有两个指针指向同一堆块造成 堆块重叠。
    - 5. Double-Free 52 进程多次释放统一堆块，被多次释放的堆块将被堆管理器分配多次，最终产生堆块重叠。
    - 6. Heap Over-Read 53 直接越界读出堆区数据，造成信息泄露
    - 7. Heap Spray 堆喷：堆喷申请大量的堆区空间，并将其中填入大量的滑板指令（NOP）和攻击恶意代码；堆喷使用户空间存在大量恶意代码，若EIP指向堆区时将命中滑板指令区，受害进程最终将"滑到"恶意代码。堆喷对抗地址的随机浮动类型的防御方案 ，并实现了恶意代码的注入。
- 操作系统基础防御方案 P55
  - 防御技术的缺陷
    - 1. 对于Stack Canary，作为Canary的内容可能被泄露给攻击者，或被暴力枚举破解
    - 2. 对于ASLR已有去随机化方案，泄露内存分布信息
    - 3. ROP等进程控制流劫持方案亦可以绕过ASLR、NX的保护机制
- 高级控制流劫持方案 P64
  - P93 两种具体实现
    - 1. 直接改写vtable指向的函数指针，可通过构造堆块覆盖完成
    - 2. 覆盖vtable字段，使其指向攻击者控制的内存，然后在其中布置函数指针
  - P90 虚假vtable劫持Fake vtable Hijacking
  - P88 ~ 89具体步骤与 GOT Hijacking 的总结
  - P82 全局偏置表劫持GOT Hijacking
    - 1. ASLR将随机浮动共享库的基地址，导致共享库的位置无法被硬编码
    - 2. 并非动态链接库当中的所有库函数都需要被映射（降低内存开销）
  - ROP 总结
  - P71 面向返回地址编程 ROP Gadget
  - 共享库机制 P67
  - 进程执行的更多细节 P65
- 高级操作系统基础防御方案 P94
  - P95~99 控制流完整性保护
    - 1. 通过二进制或者源代码程序分析得到控制流图（CFG），获取转移指令目标地址的列表
    - 2. 运行时检验转移指令的目标地址是否与列表中的地址相对应；控制流劫持会违背原有的控制流图，CFI 则可以检验并阻止这种行为
  - 指针完整性保护 P101
  - 信息流控制 P103
  - I/O子系统保护 P107