# 软件安全—恶意代码机理与防护

## C2 课后思考与课后练习

# C2 课后思考

- 恶意软件如何在自身被结束、系统重启、系统重装、以及硬盘更换之后继续获得控制权？
- PAE模式下，32位系统下每个进程的线性地址依然只有32位，系统如何能够使得进程使用4G以上地址范围的物理内存？
- 如何查看、修改进程中指定内存页面的读写属性？
- 如果硬盘分区表被完全破坏，如何重构分区表？
- MBR分区格式为何无法支持4个以上主分区项以及2T以上的单个硬盘分区？
- 为什么数据被误删除之后，应当尽量减少对其所在分区的写操作？
- 对文件进行安全删除的原理什么？
- 在使用数据恢复软件时，为什么标准格式文件（如doc、jpg等）比非格式文件更容易被恢复？为何有时候恢复出来的大文件只有前半部分是正确的？
- 恶意代码除了以文件的方式出现，它还可以隐藏在哪些区域以躲避被发现和查杀？
- 二进制恶意样本中存在的哪些信息可用于对开发者进行溯源？
- 某备份U盘每日均存入文件，有人新拷贝一个文本文件放入U盘，并将文件创建时间修改为半年之前，从磁盘存储规律的角度来说，如何辨别其时间真伪？

# C2 课后练习（1/2）

- 熟悉自己电脑硬盘分区，并进行实践：
  - 熟悉winhex或010Editor的使用，定位硬盘引导程序，可尝试对引导代码进行分析。
  - 画出硬盘的分区结构图：硬盘包含哪些分区，每个分区的开始扇区和总扇区数是多少？系统是否存在资源管理器无法看到的隐藏分区？这些分区有什么作用？
  - 在不查看分区表的情况下，还原分区表的重要内容（开始和结束位置）。
  - 尝试编程实现指定扇区读写，分区表/文件定位与恢复，文件安全删除等功能。

# C2 课后练习（2/2）

- 编写一个调用MessageBoxA进行弹框（内容为“Hello World!”）的程序并编译生成二进制可执行程序
  - 分析编译过程产生的中间文件，进一步理解二进制文件生成的过程。
  - 用Ollydbg调试该程序，分析该程序以及其所加载的典型dll文件在进程线性内存空间的范围，以及模块映像开始位置对应的内存物理地址，并验证（物理地址与线性/虚拟地址指向的数据一致）。
  - 运行程序的两个复制版本，使用Windbg定位者两个程序内存中MessageBoxA函数开始位置对应的物理地址（是否一致？），修改其中一个程序内存空间中MessageBosA函数的开始位置的字节，查看其对应的内存物理地址是否发生变化。解释原因。
  - 给出至少5种方法，使得系统重新启动之后可以自动打开该弹框程序。哪些不需要管理员权限？
  - 定位该二进制程序在磁盘中的具体存储位置（所在扇区）【NTFS及FAT32分区】
  - 通过Shift+Del删除该二进制程序，然后用Winhex手工恢复它。