返回值作用 :返回结果数据，标注程序运行成功还是失败，返回一个引用方便其他人使用

内存 数据段 静态 堆栈 堆动态内存

五、调试内容 跟踪写入 读取 执行的地址 跟踪dll加载和调用 跟踪消息和内存上下文搜索

七、识别高级语言的关键结构 可能有些是用内联汇编写的

1、函数 call ret 交叉引用 函数可以有多个出口 多ret jmp跳转到另外的ret

手工调用函数 jmp

学会跳到函数尾 观察有多少call 函数一般不定义在函数中

起始标志 push ebp mov ebp,esp sub esp,xx ebp一般用于对局部变量寻址

打开堆栈 移动指针开启空间

结束标志 恢复ebp （假如寻址编译成是使用ebp而不是esp的话）

Pop ebp

裸函数 没有标志 必须自己插入asm ret

内置函数 \_\_inline

2、启动函数 启动代码-winmain 四个参数 push 代码启动可能从别处开始

Dllmain不一定能找到 要对操作进行分析

Main 两个参数是全局变量 3、虚函数 派生 纯虚函数 虚函数表 构造 检查空指针

析构 全局构造

5、对象，结构体，数组 指针加偏移 索引\*元素大小 数组元素是相同的，结构体是不同的 对象和虚函数和this指针 new和delete可以自动识别函数

9、函数的参数 值传递 引用传递（寄存器或者堆栈） 调用约定 参数数目 传递方式

10、堆栈对参数寻址 一个堆栈只能存一个类型？ 需要两个堆栈 保存返回地址（近调用和远调用） 不优化的编译用ebp 优化的用esp

寄存器传参

传递浮点 协处理器 必须内存调用 内置数学库

This指针 默认参数

11、返回值 一般都是返回到eax寄存器 没有空间寄存 会有一个隐式引用 也可能是void 什么都不返回 返回浮点

12、局部堆栈变量 参数在ebp上方 局部变量在ebp下方

13、全局变量 直接内存寻址[] 交叉引用

14、常量和偏移量（固定的内存地址数值） 实际上都是常量

15、字符串 类型

16、if else

17、switch

18、循环

19、运算符

八。、反调试 防止汇编

调试工作原理 标志寄存器陷阱位 int1调试异常中断 调试寄存器dr 0-3 检查点 dr7检查点条件是否执行 是否修改

0Xcc断点太明显 CRC

单独调试线程的缺点 seh 异常过滤器

异常处理

反跟踪

防断点

自修改代码 cs段 ds es ss fs

写内存 打开进程 写内存

堆栈执行代码 堆栈内存是动态的

可重定位代码的缺点