**什么是汇编语言？**

汇编语言是所有程序设计语言中最古老的语言，也称为符号语言，**汇编语言(assembly language)是一种用于电子计算机、微处理器、微控制器或其他可编程器件的低级语言，亦称为符号语言**。**可以直接访问硬件**

•用助记符代替机器指令的操作码

•机器指令 55，对应的汇编指令是push ebp

•用地址符号或标号代替指令或操作数的地址

汇编语言的优点和缺点？

优点：•高速度、高效率

汇编语言给了程序员极大的自由，可以直接与机器对话，不需要依靠各种“翻译人员”

汇编语言可以节省内存空间

汇编语言可以完全控制机器码，执行速度快

汇编语言会加快数据加密速度

•程序的深入理解和分析

•计算机病毒分析、漏洞发掘

•软件盗版、破解、代码盗用

•汇编语言和逆向工程是系统开发、系统安全的基础课程

缺点：

•需要了解硬件和系统的知识

•可移植性差，不同的CPU使用不同的指令集

•适用范围有限

虚拟机的概念：抽象出计算机软硬件之间的相互关系

**大端序与小端序定义及举例**

一个多字节组成的数据，最高位被存储在内存的低地址上是大端序

一个多字节组成的数据，最低位被存储在内存的低地址上是小端序

**软件保护技术的原理和局限**

软件保护技术的原理是利用各种方法和技术，使软件的代码和数据难以被分析、修改和复制，从而保护软件的版权和安全。软件保护技术使得软件具有抗御反向工程和篡改的能力，但它只能保护软件在被编译后不易被反汇编及被修改，而并非绝对安全的措施。

序列号保护，警告窗口，时间限制，菜单公能限制，KEYFILE保护。

**漏洞发掘**

**取证**

**性能分析**

**软件保护**

**导出表获取函数地址的步骤**

**Windows 装载器在进行 PE 装载时，会将导表中登记的所有 DLL一并装入，然后根据DLL的导出表中对导人函数的描述修正导人表的 IAT 值。通过导出表，DLL 文件向调用它的程序或系统提供导出函数的名称、序号，以及入口地址等信息。**

**1. AddressOfName定位“函数名称数组”**

**2. 在“函数名称数组”中，通过比较字符串（strcmp），查找指定的函数名称**

**此时的数组索引称为name\_index**

**3. 利用AddressOfNameOridinals成员，定位ordinal数组**

**4. 在ordinal数组中，通过name\_index查找相应的ordinal值**

**5. AddressOfFunctions，定位“函数地址数组”（EAT）**

**在“函数地址数组”中，利用ordinal值作为索引，获得指定函数的起始地址**

通过Loadlibrary(GetModelHandle)将dll模块映射进内存并返回一个可以被GetProcAddress函数使用的句柄,再利用GetProcAddress得到dll的加载地址,通过遍历导出表就可以得到该函数的地址.

**代码标号和数据标号定义和举例**

标号是充当指令或数据位置标记的标识符

数据标号：标识变量的地址

举例：标识变量的地址，方便变量的引用

count DWORD 100

array DWORD 100, 101, 102, 103

相对.data数据段在内存起始地址的偏移

代码标号：标识代码的地址.跳转、循环指令的目标地址

target：

mov eax，100h

…

jmp target

**ollygbg 里断点种类及原理**

断点（Breakpoint）是调试器的重要功能，使执行的程序中断在指定的地方，从而方便对其进行分析。

软件断点

INT 3指令替换原始CPU指令，操作码是0xCC

INT 3指令产生异常，停止程序执行,可以设置无数个断点取得检测函数的地址，然后读取它的第1个字节，判断它是否等于“CC"

硬件断点

将断点的内存地址存储到CPU的DRx寄存器,基于CPU中断DRx寄存器设置,如果一致，CPU产生异常，中止程序执行，将控制权转移给调试器

缺点：只有4个CPU寄存器存储断点地址，DR0到DR3

内存断点

对所设的内存地址页设置不可访问或者不可写属性，当被调试程序访问或者写入内存页时，产生异常，中断执行，CPU将控制权交给OD

OD同一时间，只能设置1个内存断点

消息断点

Windows是由消息驱动的，当某个特定窗口函数接收到某个特定消息时，消息断点将使程序中断

条件断点

条件断点是带有条件表达式的INT 3断点,在INT3断点的基础上添加一个条件表达式,在调试过程中，在满足一定条件时断点才会触发，这类断点称为条件断点。OllyDbg的条件断点可以按寄存器、内存、消息等设断点

## **常用断点：内存访问断点。**

**如何识别局部变量、参数、全局变量，如何识别 if 语句、while 语句等**

**使用栈传递参数、存储局部变量**

**返回值通过eax寄存器返回**

**1.全局变量的地址是固定的，不会随着函数的调用而改变，它们的地址通常用一个标号或一个立即数来表示，例如mov eax, [global]或mov eax, [0x12345678]，其中global或0x12345678就是全局变量的地址。全局变量在程序编译完成后地址就已经确定下来了，只要程序启动，全局变量就已经存在了，启动后里面是否有值取决于声明时是否给定了初始值，如果没有，默认为0**

**2.局部变量的地址是相对于栈帧基址的，会随着函数的调用而改变，它们的地址通常用一个偏移量来表示，例如mov eax, [ebp-4]或mov eax, [esp+8]，其中ebp-4或esp+8就是局部变量的地址。局部变量在程序编译完成后并没有分配固定的地址**

**在所属的方法没有被调用时，局部变量并不会分配内存地址，只有当所属的程序被调用了，才会在堆栈中分配内存**

**参数通过寄存器传递**

**3.IF语句的识别特征：jxx的跳转和一个无条件jmp指令**

**4.Switch:使用IF方式.使用跳转表**

**5.循环：FOR循环有4个组件：**

**初始化**

**比较**

**指令执行体**

**递增或递减**

**false的分支有一个指回去的循环过程**

**6.While语句的识别特征：jxx的向下跳转和一个无条件jmp指令向上跳转。**

**7.do:Do循环的识别特征 执行影响标志位，jxx向上跳转**

**Err**

**逆向分析题: 函数调用过程 15 分**

**1.以字母开头的十六进制常量前面必须加0**

**2.标识符是程序员选择用来标识变量、常量、过程、代码的符号**

**包含1~247个字符**

**大小写不敏感（MASM默认）**

**第一个字符必须是字母、下划线、@、？或$**

**第一个字符不能是数字（对比十六进制整数）**

**操作数：操作数是指令的操作对象**

**寄存器**

**内存**

**常量**

**I/O端口**

**什么是伪指令？ 我们为什么要学习伪指令？**

**用于定义变量、段、过程、汇编器选项等，**并不是可执行指令，没有机器代码****，****只用于汇编过程中为汇编程序提供汇编信息。****

****定义段（Segment）****

****.data、.code、.stack****

****定义过程（Procedure）****

****PROC、ENDP****

****允许或禁止汇编器的某些特性****

****OPTION、.386、.MODEL****

****[变量名] 数据定义伪指令 初始值****

****?: 表示在程序运行的时候初始化该变量****

****DUP为字符串或者数组分配内存空间****

****BYTE 20 DUP（0）；20个字节的内存空间****

****BYTE 4 DUP（“Hello”）：20个字节，连续的4个“Hello”，每个“Hello”5字节****

****控制转移（transfer of control）是一种改变汇编语句执行顺序的方法。****

****无条件转移：****

****将CPU控制权直接转移到指定的汇编语句****

****修改EIP为指定的内存地址****

****CPU从EIP指定的内存地址读取下一条机器指令****

****无条件转移： JMP****

****条件转移：****

****修改EIP为指定的内存地址****

****CPU从EIP指定的内存地址读取下一条机器指令****

****CMP指令执行从源操作数中减掉目的操作数的减法操作****

****用于条件跳转指令的条件判断****

****不改变目的操作数和源操作数，只影响eflags的标志位****

****设置相应的标志位****

****标志位：OF、SF、ZF、AF、PF、CF****

**LOOP 目的地址**

**LOOP指令可以指定循环执行的次数 （loop count）**

**ECX寄存器作为循环计数器**

**LOOP指令执行时，ECX减1**

**如果ECX不等于0，跳转到目的地址**

**如果ECX等于0，不跳转，顺序执行**

**编译器是如何将这些高级语言中的函数、过程翻译成汇编语言的？**

**汇编语言把过程定义为以返回语句结束的命名语句块。**

**使用PROC和ENDP伪指令来声明过程**

**必须定义一个过程名字（标识符）**

**除启动过程之外，其它过程以ret指令结束**

**链接库（Link Library）是一个文件，包含已经编译成机器码的过程。**

**MOV指令**

**两个操作数的尺寸必须一致**

**两个操作数不能同时为内存操作数**

**目的操作数不能是CS、EIP和IP**

**立即数不能直接送至段寄存器**

**变量名（数据标号）**

**数据段内偏移地址**

**MOVZX指令 (move with zero-extend)**

**填充0**

**MOVSX（move with sign-extend）符号扩展传送指令，最高位循环填充所有扩展位1**

**判断这个数的首位是1还是0即可。**

**LAHF（load status flags into AH）指令把EFLAGS寄存器的低字节复制到AH寄存器**

**XCHG（exchange data）指令交换两个操作数的内容**

**XCHG reg, reg**

**XCHG reg, mem**

**XCHG mem, reg**

**相同尺寸！！**

**OFFSET操作符返回数据标号的偏移地址**

**CS的值一般是0，OFFSET等同内存虚拟地址，占用四个字节**

**ALIGN伪指令可以对代码段的指令进行对齐操作。**

CPU 处理偶数地址的数据要比处理奇数地址的数据更快。

LENGTHOF操作符计算数组中元素的数目，元素由出现在同一行的值定义，不同行有逗号的话，两行就连着，没有逗号，就不连着。

SIZEOF操作符的返回值等于LENGTHOF和TYPE返回值的乘积

注意：label后面的数据类型不能用缩写，

例如dword不能写成dd

间接操作数

任何一个 32 位通用寄存器（EAX、EBX、ECX、EDX、ESI、EDI、EBP 和 ESP）加上方括号就能构成一个间接操作数 操作数必须说明大小。

**ARMv8-AArch32**

**可以使用A32指令集：32位等长指令字**

**或T32指令集：16位和32位可变长指令字**

**R0-R14通用寄存器 13个**

**R13为堆栈指针，R14为链接寄存器，R15程序计算机PC指针**

**衡量一个CPU实时性就是最短响应中断时间以及单位时间内响应中断次数**

**ARMv8-AArch64**

**只能使用A64指令集：32位等长指令字**

**有31个通用寄存器**

**Double Word占用八个字节**

**立即数寻址指令中的地址码就是操作数本身**

**若操作数为常量，用#表示常量**

**0x或&表示16进制数，否则表示十进制数**

**LDR（Load Register）指令用于读取内存数据**

**STR（Store Register）指令用于写入内存数据**

**前索引：#在前面，先加后赋值**

**LDR R0, [R1, #4]! 注：！表示回写地址**

**后索引：#在后面，先赋值后加**

**LDR R0, [R1],#4!**

**栈的4种管理方式**

**x86 CPU支持FD （Full Descending Stack） ：满递减栈，每次传送后栈指针减4，对应DA**

**FA（Full Ascending Stack）：满递增栈，每次传送后栈指针加4，对应IA**

**ED （Empty Descending Stack） ：空递减栈，每次传送前栈指针减4，对应DB**

**EA （Empty Ascending Stack） ：空递增栈，每次传送前栈指针加4，对应IB**

**ARM指令集**

**不可大小写混用**

**A64指令特点：**

**31个（X0-X30）个64bit通用寄存器（用作32bit时是W0-W30）**

**支持48bit虚拟寻址空间**

**PE文件结构**

**.exe,.dll,.sys文件使用的是PE文件结构：可移植可执行文件结构**

**理解PE文件结构是逆向技术的基础**

**节：**

**代码节、数据节**

**各个节按页边界对齐**

**节是一个连续结构，没有大小限制**

**每个节都有自己的内存属性**

**相对虚拟地址：**

**模块的地址冲突问题**

**模块的加载顺序和加载地址是不确定的。**

**在可执行文件中，有许多地方需要内存地址**

**例如，引用全局变量时需要指定它的地址**

**PE文件有一个首选的载入地址(基地址)**

**PE文件可以载入到进程空间任何地方**



**RVA是相对于PE文件载入地址的偏移位置**

**PE文件储存在磁盘中，某个数据的位置相对于文件头的偏移量称为文件偏移地址**

**每个PE文件都是以一个16位的DOS程序开始的**

**DOS MZ头与DOS stub合称为DOS文件头**

**PE 文件前两个字节是4D5A**

**PE文件头(PE Header)紧跟在DOS stub的后面**

**IMAGE\_DOS\_HEADER结构的e\_lfanew字段定位PE Header的起始偏移量，加上基址，得到PE文件头的指针**

**0002h代表文件可执行，2000h代表是DLL文件**

PE文件来说，这个字段的值一般是010fh,而对于DLL文件来说，这个字段的值一般是210eh。

**可执行可读可写 2 4 8**

**代码 可初始化 未初始化 2 4 8**

**局部变量（ local variable）**

**前缀: var\_**

**后缀: 相对EBP的偏移值**

**偏移值为负值**

**参数（argument）**

**前缀： arg\_**

**后缀：相对于EBP的偏移值**

**偏移为正值**

retn操作：先eip=esp，然后esp=esp+4

retn N操作：先eip=esp，然后esp=esp+4+N

**Add cdecl**

**Mov stdcall**