《逆向工程》课程期末复习资料

《逆向工程》课程讲稿章节目录：

6.7 TEB 和 PEB

第 7 章 Windows 下的异常处理

7.1 异常处理的基本概念

7.2 SEH 的概念及基本知识

7.3 SEH 异常处理程序原理及设计

7.4 向量化异常处理

7.5 x64 平台上的异常处理

7.6 异常处理的实际应用

第 8 章 PE 文件格式

8.1 PE 的基本概念

8.2 PE 文件头

8.3 区块

8.4 输入表

8.5 绑定输入、输出表

8.6 基址重定位

8.7 资源

8.8 TLS 初始化、调试载入目录、 延迟载入数据 _

8.9 程序异常数据 NET头部

8.10 编写 PE编辑工具

一、客观部分：（单项选择、多项选择、判断）

★考核知识点： 基础知识

参见讲稿章节： 1.1

1. 在关于逆向工程（ reverse engineering ）的描述中，正确的是（ ）

A. 从已经安装的软件中提取设计规范

B. 按照“ 输出—＞处理—＞输入 ”的顺序设计软件

C. 用硬件来实现软件的功能

D. 根据软件处理的对象来选择开发语言和开发工具

2. 以下说法错误的是（）

A.逆向工程是指根据已有的产物和结果，　通过分析来推导出具体的实现方法。

B.在软件汉化和软件解密的过程中，　首要问题是对被汉化和解密的软件进行分析。

C.通过静态分析我们可以真正了解软件中各个模块的技术细节。

D.对软件分析来说，静态分析只是第一步，动态跟踪才是分析软件的关键。

附（考核知识点解释）：

逆向工程 (Reverse Engineering)　是指根据已有的产物和结果，通过分析来推导出具体的实现方法。对软件来说，“ 可执行程序—反编译—源代码 ”　　的过程就是逆向工程。　逆向工程的内容可以分为如下　3 类。

（1）软件使用限制的去除或者软件功能的添加。

（2）软件源代码的再获得。

（3）硬件的复制和模拟。

参见讲稿章节：　1.3

1. 判断题：计算机中储存的信息都是用二进制数表示的，　但如果要处理文本，并不需要先把文本转换为相应的二进制数。　（）

2. 判断题：Unicode 是 ASCII 字符编码的一个扩展，只不过在　Windows中用 2 字节对其进行编码。（）

附（考核知识点解释）：

文本字符

计算机中储存的信息都是用二进制数表示的，　屏幕上显示的字符都是二进制数转换之后的结果。如果要处理文本，就必须先把文本转换为相应的二进制数。在学习过程中，我们会与各类字符打交道。　这些字符在　Windows里扮演着重要的角色。

字符集

字符集是一个系统支持的所有抽象字符的集合。

字符是各种文字和符号的总称：

文字

标点符号

图形符号

数字

IP 地址

字节存储顺序

计算机领域在描述“ 关于字节该以什么样的顺序传送的争论 ”时引用了“ endian ”一词，翻译为“ 字节序 ” ，表示数据在存储器中的存放顺序，主要分为大端序 (Big-endian) 和小端序 (Little-endian) ，其区别如下。

Big-endian: 高位字节存入低地址，低位字节存入高地址。

Little-endian: 低位字节存入低地址，高位字节存入高地址。

Intel 处理器使用的是小尾方式存储（ Little Endianness ）

低位字节存入低地址，高位字节存入高地址

参见讲稿章节： 1.6

1. 判断题：Windows是一个消息 (Message) 驱动式系统。 Windows消息提供在应用程序与应用程序之间、应用程序与 Windows系统之间进行通信的手段。

附（考核知识点解释）：

Windows消息机制

Windows是一个消息 (Message) 驱动式系统。 Windows消息提供在应用程序与应用程序之间、 应用程序与 Windows系统之间进行通信的手段。 应用程序想要实现的功能由消息触发，通过对消息的响应和处理完成。 Windows系统中有两种消息队列 : 一种是系统消息队列 ; 另一种是应用程序消息队列。 计算机的所有输入设备由 Windows监控。当一个事件发生时， Windows先将输入的消息放入系统消息队列，再将输入的消息复制到相应的应用程序队列中， 应用程序中的消息循环在它的消息队列中检索每个消息并发送给相应的窗口函数。 一个事件从发生到到达处理它的窗口函数必须经历上述过程。 值注得意的是消息的非抢先性， 即不论事件的急与缓， 总是按到达的先后排队 (一些系统消息除外) ，而这可能导致一些外部实时事件得不到及时的处理。

★考核知识点： OllyDdg 调试器

参见讲稿章节： 2.1

1. 判断题： OllyDbg 只能进行动态调试。 （）

2. 判断题： OllyDbg 是调试 Ring 0 级程序的首选工具。 （）

OllyDbg 调试器

附（考核知识点解释）：

OllyDbg(简称“ OD”) 是由 Oleh Yuschuk (www.ollydbg.de) 编写的一款具有可视化界面的用户模式调试器。 OllyDbg 结合了动态调试和静态分析，具有 GUI 界面，非常容易上手，对异常的跟踪处理相当灵活。这些特性使 OllyDbg 成为调试 Ring 3 级程序的首选工具。它的反汇编引擎很强大，可识别数千个被 C 和 Windows 频繁使用的函数， 并能将其参数注释出来。 它会自动分析函数过程、循环语句、代码中的字符串等。

★考核知识点：OllyDdg 的 INT3 断点和硬件断点

1. 判断题： INT3 断点不改变原程序的机器码。 （）

2. 判断题：硬件断点最多可以设置 4 个。（）

参见讲稿章节： 2.4

附（考核知识点解释）：

断点

断点（Breakpoint ）是调试器的一个重要功能， 使执行的程序中断在指定的地方，从而方便对其进行分析。

INT3 断点

INT 3 断点指令，其机器码是 OxCÇ 也常被称为“ CC指令 ”。当被调试进程执行 INT 3 指令导致一个异常时， 调试器就会捕捉这个异常， 从而停在断点处，然后将断点处的指令恢复成原来的指令。

使用 INT 3 断点的优点是可以设置无数个断点， 缺点是改变了原程序机器码，容易被软件检测到。

硬件断点

硬件断点和 DRx调试寄存器有关。硬件断点的原理是使用 DR0, DR1, DR2, DR3设定地址，并使用 DR7设定状态，因此最多设置 4 个断点。硬件执行断点与

CC断点的作用一样，但因为硬件执行断点不会将指令首字节修改为“　　CC”，所以更难检测。

★考核知识点：OllyDdg　内存断点

参见讲稿章节： 2.5

选择题： OllyDbg 可以设置（）个内存断点

A. 1 个　　B.4 个　　C.5 个　　D.7 个

附（考核知识点解释）：

内存断点

OllyDbg 可以设置内存访问断点或内存写入断点，　原理是对所设的地址的内存页设置不可访问或不可写属性，这样当内存访问或写入的时候就会产生异常。OllyDbg 截获异常后，比较异常地址是不是断点地址，如果是就中断，让用户继续操作。

因为每次出现异常时都要通过比较来确定是否应该中断，　所以内存断点会降低 OllyDbg 的执行速度。 OllyDbg 只能设置 1 个内存断点。

★考核知识点： WinDbg调试器

参见讲稿章节： 2.10

判断题： WinDbg最强大的地方还是命令行，通常结合　GUI 和命令行进行操作。()

多选题： WinDbg支持哪些调试（）

A. 以打开、附加的方式调试应用程序

B. 可以分析 Dump文件

C. 可以进行远程调试

D. 内核调试

★考核知识点： 文件类型分析

参见讲稿章节： 3.1

单选题： IDA 的原始嵌入式脚本语言叫作（） 。

A.FLIRT B.FLAIR C.IDC D.Perl

单选题： IDA 分析数据时，数据类型可以在（）之间转换

A.db、dw、df B.db 、dw、dd C.db 、dd、df D.dw 、dd、df

多选题：下列说法正确的是（）

A. PEiD 这类文件分析工具是利用导入函数搜索来完成识别工作的

B. 开发语言都有固定的启动代码，利用这一点就可以识别程序是由何种语言编译的

C. 被加密程序处理过的程序中会留下加密软件的相关信息，利用这一点就可以识别程序是被何种软件加密的

D. PEiD提供了一个扩展接口文件 userdb.txt ，用户可以自定义一些特征码，这样就可以识别新的文件类型了

附（考核知识点解释）：

文件类型分析

逆向分析程序的第一步就是分析程序的类型， 了解程序是用什么语言编写的或用什么编译器编译的， 以及程序是否被某种加密程序处理过， 然后才能有的放矢，进行下一步工作。 这个分析过程需要文件分析工具的辅助。 常见的文件分析工具有 PEiD，Exeinfo PE等。此类工具可以检测大多数编译语言、病毒和加密软件。

★考核知识点：反汇编引擎

参见讲稿章节： 3.2

单选题：以下对 x86 架构指令集支持最全的反汇编引擎是（）

A. ODDisasm B. BeaEngine C. Udis86 D. Capstone

多选题：下列是反汇编引擎的有（）

A.ODDisasm B.BeaEngine C.Udis86 D.Keystone

判断题： OllyDbg 自带的反汇编引擎 ODDisasm，优点是具有汇编接口（即文本解析，将文本字符串解析并编码成二进制值 ），这个特性曾经独树一帜，且支持 64 位指令的汇编和反汇编。（）

判断题：Keystone 和 Capstone 是同一系列的引擎， 由同一维护者主导开发。

Capstone 主要负责跨平台多指令集的反汇编工作，　而 Keystone 主要负责跨平台多指令集的汇编工作。 与 O11yDbg的汇编器一样， Keystone 也只支持文本汇编，不支持像 AsmJit 那样的函数式汇编。 （）

附（考核知识点解释）：

反汇编引擎概述

在安全软件和保护软件的开发过程中经常会用到汇编引擎和反汇编引擎，　例如 OllyDbg 、IDA、VMProtect、加壳软件和反编译器等。反汇编引擎的作用是把机器码解析成汇编指令。

常用的反汇编引擎有　ODDisasm、BeaEngine、Udis86、Capstone，汇编引擎有 ODAssemhler、Keystone、AsmJit 。

★考核知识点：IDA Pro　加载可执行文件

参见讲稿章节： 3.3

单选题： IDA 是按（）装载　PE文件的

A. 交叉参考　B. 参考重命名　C. 格式化指令操作数　D. 代码和数据转换

多选题：下列关于　IDA 有关说法正确的是（）

A. IDA　最主要的特性是交互和多处理器。用户可以通过对　　IDA 的交互来指导 IDA 更好地进行反汇编。

B. IDA　是按区块装载　　PE 文件的，例如　.text(　代码块　)、 .data(　数据块)、 .rsrc(　资源块)、.idata(　输入表)和.edata(　输出表)等

C. IDA　反汇编所消耗的时间与程序大小及复杂程度有关，通常需要等待一段时间才能完成。

D. IDA 可以格式化指令使用的常量，　因此应尽可能使用符号名称而非数字，从而使反汇编代码更具可读性。　IDA 根据被反汇编指令的上下文、所使用的数据作出格式化决定。 对其他情况， IDA 一般会将相关常量格式化成一个十进制常量。

★考核知识点：IDA Pro　导航

参见讲稿章节： 3.5

单选题： IDA PRO简称 IDA，是一个交互式（）工具

A. 调试 B. 汇编 C. 编译 D. 反汇编

★考核知识点：IDA Pro　交叉引用

参见讲稿章节： 3.6

单选题：通过（）可以知道指令代码的相互调用关系

A.交叉参考　B. 参考重命名　C. 格式化指令操作数　D. 代码和数据转换

★考核知识点：IDA Pro　函数分析

参见讲稿章节： 3.7

多选题： IDA 反汇编代码可以输出（）格式文件

A.MAP B.ASM C.EXE D.DIF

★考核知识点：IDA Pro　识别数组、结构体

参见讲稿章节： 3.8

判断题： IDA 有着较强的数组聚合能力。它可以将一串数据声明变成一个反汇编行，按数组的形式显示，从而简化反汇编代码清单。　（）

判断题： IDA 在进行反汇编的时候能正确区分数据和代码。　（）

★考核知识点：IDA Pro　增强反汇编

参见讲稿章节： 3.9

判断题：进人指令汇编修改状态，　jne 指令共 2 字节，因此用 2 个 nop 指令代替，这种跳过算法分析直接修改关键跳转指令使程序注册成功的方法，　通常被解密者称为“ 爆破法 ”。 （）

★考核知识点： 十六进制工具与静态分析技术应用实例

参见讲稿章节： 3.10

单选题：（）十六进制工具提供了文件比较功能

A.HexWorkshop B.WinHex C.Hiew D.ApateDNS

多选题：常用的十六进制工具有（）

A.HexWorkshop B.WinHex C.Hiew D.ApateDNS

参见讲稿章节： 4.1

多选题：在以下的传递方式中，　（）是函数传递参数的方式

A栈方式

B队列方式

C寄存器方式

D通过全局变量进行隐含参数传递

单选题：虚函数的地址是在（）时候确定的

A程序编写时

B编译程序时

C调用即将进行时

D程序执行后

判断题：局部变量是函数内部定义的一个变量，　其作用域和生命周期作用于整个程序。（）

判断题：调用虚函数时，程序先取出虚函数表指针，得到虚函数表的地址，再根据这个地址到虚函数表中取出该函数的地址，最后调用该函数。　（）

参见讲稿章节： 4.5

单选题：以下先执行语句块，再进行表达式判断的循环语句是。　（）

A do 循环

B while　循环

C for　循环

D 都不是

判断题：程序在运行时，先调用　main 函数执行用户编写的代码，再执行初始化函数代码。（）

参见讲稿章节： 4.7

判断题： C++的三大核心机制是封装、继承、多态，虚函数就是多态的一种体现。 VC++实现虚函数的方式是虚表。 （）

附（考核知识点解释）：

C++的三大核心机制是封装、继承、多态，虚函数是多态的一种体现，对于

面向对象思想设计的软件， 虚函数是软件逆向分析还原面向对象代码的重要手段。

VC++实现虚函数的方式是虚表， 如果一个类中有虚函数， 编译器就会为这个类生成一个虚表，不同的类，虚表是不相同的，相同的类对象，共享一个虚表。

★考核知识点： 序列号保护方式

参见讲稿章节： 5.1

判断题：软件验证序列号， 其实就是验证用户名和序列号之间的数学映射关系（）

★考核知识点： 警告窗口

参见讲稿章节： 5.2

判断题：若要完全去除警告窗口，只需找到创建该窗口的代码并将其跳过。（）

判断题：在另外一些情况下， 对话框不是以资源形式存在的， 通过常用断点就可以拦截不下来。（）

多选题：去除警告窗口常用的 3 种方法是（）

A. 修改程序的资源

B. 静态分析

C. 动态分析

D. 放置不管

★考核知识点： 时间限制

参见讲稿章节： 5.3

判断题：演示版软件一般都有使用时间的限制，例如试用 30 天，超过试用期也能运行。（）

多选题：用于获取时间的 API 函数有（）

A. GetSystemTime

B. GetLocalTime

C. GetFileTime

D. timeGetTime

★考核知识点： 菜单功能限制

参见讲稿章节： 5.4

选择题：允许或禁止指定的菜单条目的 API 函数是（）

A. EnableMenuItem() 函数

B. Enablewindow() 函数

C. GetTickCount() 函数

D. timeGetTime() 函数

★考核知识点：KeyFile 保护

参见讲稿章节： 5.5

选择题：确定文件是否存在的 API 函数是（ ）

A. FindFirstFileA 函数

B. CreateFileA 函数

C. GetFileAttributesA 函数

D. ReadFile 函数

选择题：打开文件以获得其句柄的 API 函数是（）

A. FindFirstFileA 函数

B. CreateFileA 函数

C. GetFileAttributesA 函数

D. ReadFile 函数

选择题：允许或禁止指定窗口的 API 函数是（）

A. EnableMenuItem() 函数

B. Enablewindow() 函数

C. GetTickCount() 函数

D. timeGetTime() 函数

★考核知识点： 内核空间、权限空间布局

参见讲稿章节： 6.1

单选题：CPU设计者将 CPU的运行级别从内向外分为 4个, 依次为 R0, R1, R2, R3, （）拥有最高执行权限

A. R0 B .R1 C.R2 D.R3

单选题：用户的应用程序 (就是用 Visual C++ 等工具开发的应用程序) 也是运行在 ( ) 级上的

A. R0 B .R1 C.R2 D.R3

★考核知识点：Windows 与内核启动过程

参见讲稿章节： 6.2

单选题：请对 Windows的启动过程包括的以下几个阶段的顺序进行排列（）

（1）初始化启动阶段

（2）启动自检阶段

（3）Boot 加载阶段

（4）检测和配置硬件阶段

A 3412 B 2341 C 2134 D 3214

判断题： Windows与内核启动过程中在 Boot 加载阶段，先对 ntldr 进行设置，然后从启动分区加载 ntldr 。（）

附（考核知识点解释）：

Windows的启动过程包括以下几个阶段。

（1）启动自检阶段

在打开电源时，计算机开始自检过程，从 BIOS中载入必要的指令，然后进行一系列的自检操作，进行硬件的初始化检查 (包括内存、硬盘、键盘等)，同时在屏幕上显示信息。

（2）初始化启动阶段

自检完成后，根据 CMOS的设置， BIOS 加载启动盘，将主引导记录 (MBR)中的引导代码载入内存。 接着，启动过程由 MBR来执行。 启动代码搜索 MBR中的分区表，找出活动分区， 将第 1 个扇区中的引导代码载入内存。 引导代码检测当前使用的文件系统，查找 ntldr 文件，找到之后将启动它。 BIOS 将控制权转交给

ntldr ，由 ntldr 完成操作系统的启动工作 (注意:Windows 7 与此不同，使用的是 Bootmgr)。

（3） Boot 加载阶段

在这个阶段，先从启动分区加载 ntldr ，然后对 ntldr 进行如下设置。

①设置内存模式。如果是 x86 处理器，并且是 32 位操作系统，则设置为“ 32- bit flat memorymode”; 如果是 64 位操作系统，并且是 64 位处理器，则设置为 64 位内存模式。

②启动一个简单的文件系统， 以定位 boot.ini 、ntoskrnl 、Hal 等启动文件。

③读取 boot.ini 文件。

（4）检测和配置硬件阶段

在这个阶段会检查和配置一些硬件设备， 例如系统固件、 总线和适配器、 显示适配器、键盘、通信端口、磁盘、软盘、输入设备 (例如鼠标)、并口、 ISA 总线上运行的设备等。

（5）内核加载阶段

ntldr 将首先加载 Windows内核 Ntoskrnl.exe 和硬件抽象层 (HAL)。 HAL 会对硬件底层的特性进行隔离，为操作系统提供统一的调用接口。接下来 ntldr 从注册表的 HKEY_LOCAL_MACHINE\System\CurrentControlSet 键下读取这台机器安装的驱动程序，然后加载驱动程序。初始化底层设备驱动，在注册表的 HKEY_LACAL_MACHINE\System\CurrentControlSet\Services 键下查 找“ Start ”键的值为 0 和 1 的设备驱动。

（6）Windows的会话管理启动

驱动程序加载完成， 内核会启动会话管理器。 这是一个名为 smss.exe 的程序，是 Windows系统中第 1 个创建的用户模式进程，其作用如下。

. 创建系统环境变量。

. 加载 win32k.sys ， 它是 Windows子系统的内核模式部分。

. 启动 csrss.exe ， 它是 Windows子系统的用户模式部分。

. 启动 winlogon.exe 。

. 创建虚拟内存页面文件。

. 执行上次系统重启前未完成的重命名工作 (PendingFileRename) 。

（7）登录阶段

Windows子系统启动的 winlogon.exe 系统服务提供对 Windows用户的登录和注销的支持，可以完成如下工作。

. 启动服务子系统 (services.exe) ，也称服务控制管理器 (SCM)。

. 启动本地安全授权 (LSA)过程 (lsass.exe) 。

. 显示登录界面。

登录组件将用户的账号和密码安全地传送给 LSA进行认证处理。如果用户提供的信息是正确的，能够通过认证，就允许用户对系统进行访问。

（8）Windows 7 和 Windows XP启动过程的区别

.BIOS 通过自检后，将 MBR载入内存并执行，引导代码找到启动管理器 Bootmgr。

.Bootmgr 寻找活动分区 boot 文件夹中的启动配置数据 BCD文件，读取并组成相应语言的启动菜单，然后在屏幕上显示多操作系统选择画面。

. 选择 Windows 7 系统后，Bootmgr 就会读取系统文件 windows\system32\winload.exe ，并将控制权交给 winload.exe 。

.Winload.exe 加载 Windows 7 的内核、硬件、服务等，然后加载桌面等信息，从而启动整个 Windows 7 系统。

★考核知识点：Windows R3 与 R0通信

参见讲稿章节： 6.3

单选题：Windows内核部分会调用一些内核层的函数。这些函数都以固定的前缀开始，分别属于内核中不同的管理模块，其中“ Ps ”属于哪个模块（）

A 管理层 B 核心层 C 进程管理 D 安全管理

判断题：Windows内核驱动模块是内核的重要组成部分，虽然有微软自己开发的内核驱动，但是没有第三方开发的内核驱动。 （）

附（考核知识点解释）：

Windows分为应用层与内核层。当应用程序调用一个有关 I/O 的 API(例如 WriteFile) 时，实际上这个 API 被封装在应用层的某个 DLL 库 (例如 kernel32.dll 和 user32.dll) 文件中。而 DLL动态库中的函数的更底层的函数包含在 ntdll.dll 文件中。

ntdll.dll 中的 Native API 是成对出现的，分别以“ Nt ”和“ Zw”作为前缀。即 ZwCreateFile 函数和 NtCreateFile 函数，只是名字不一样。 WindowsAPI 函数调用 ntdll.dll 中的 Native API 函数。 Native API 函数调用中断 int 2E 或者 SysEnter 指令，从 R3进入 R0。内核 ntoskrnl.exe 中的 SSDT，有与 ntdll.dll 中 Native API 一一对应的系统处理服务函数，即内核态的 Nt* 系列函数。

★考核知识点： 内核函数与内核驱动模块

参见讲稿章节： 6.4

判断题： SSDT的全称是系统服务描述符表， SSDT 用于处理应用层通过 kernel32.dll 下发的各个 API 操作请求。（）

★考核知识点： 内核对象

参见讲稿章节： 6.5

判断题：TEB与 PEB不一样， 在系统内核空间中， 而不是在应用层中的结构。（）

判断题：PEB 存在于用户地址空间中， 记录了进程的相关信息。 每个进程都有自己的 PEB信息。（）

★考核知识点：SSDT

参见讲稿章节： 6.6

判断题： WinDbg 有限制地支持本地内核调试，只能查看一些重要的系统数据结构，不能通过下断点的方式进行调试。 （）

判断题：利用 WinDbg调试内核有多种方法。 例如，WinDbg通过 USB、 1394 火线、 COM及网络把两台机器连接起来。 （）

★考核知识点：TEB 与 PEB

参见讲稿章节： 6.7

判断题： x64 系统内核驱动需要验证数字签名，但是直接运行没有数字签名

的驱动的操作也能成功。（）

判断题：加载内核驱动的方法很多，可以使用 instdrv.exe 工具来加载。（）

★考核知识点： 异常处理的基本概念

参见讲稿章节： 7.1

选择题：CPU设计者将 CPU的运行级别从内向外分为 4 个, 依次为 R0, R1, R2, R3, （）拥有最高执行权限

A.R0

B.R1

C.R2

D.R3

选择题：用户的应用程序 (就是用 Visual C++ 等工具开发的应用程序) 也是运行在 ( ) 级上的

A. R0

B.R1

C.R2

D.R3

选择题：除了 CPU能够捕获一个事件并引发一个硬件异常外，在代码中可以主动引发一个软件异常，这只需调用（ ）函数

A. RaiseExeeption()

B. nt!RtlDispatchException

C. KeBugCheckEx

D. KiDispatchException

★考核知识点：SEH 的概念及基本知识

参见讲稿章节： 7.2

判断题： SEH机制只能在用户模式下使用。（）

判断题： VEH机制支持用户模式和内核模式。（）

★考核知识点： 向量化异常处理

参见讲稿章节： 7.4

判断题：在 Windows的任何版本中都有向量化异常处理。 （）

★考核知识点： x64 平台上的异常处理

参见讲稿章节： 7.5

判断题：x64 平台上原生 x64 程序的异常分发流程与 x86 平台上不是完全一致的。（）

★考核知识点：PE 文件格式

参见讲稿章节： 8.1

判断题： PE文件是作为单一内存映射文件被载人内存的。 （）

判断题：每个 PE文件都是以一个 DOS程序开始的， 有了它，一旦程序在 DOS下执行， DOS就能识别出这是一个有效的执行体。 （）

参见讲稿章节： 8.2

单选题：用十六进制工具查看 IMAGE_ FILE_HEADER结构的情况时，以下字段中哪个代表可执行文件的目标 CPU类型。

A NumberOfSections

B Machine

C TimeDateStamp

D Characteristics

参见讲稿章节： 8.3

判断题：区块的大小是要对齐的。有两种对齐值，一种用于磁盘文件内，另一种用于内存中。（）

判断题：链接器的并不能够合并区块。 （）

参见讲稿章节： 8.5

选择题：数据目录表 (DataDirectory) 的第（）个成员指向绑定输人。绑定

输入以一个 IMAGE_ BOUND_IMPORT _DESCRIPTOR 结构的数组开始。

A 10　　B 11　　C 12　　D 13

选择题：输出表 (Export Table) 的主要内容是一个表格，其中包括函数名称、输出序数等。序数是指定 DLL中某个函数的（）位数字，在所指向的 DLL里是独一无二的。

A 13　　B 14　　C 15　　D 16

二、主观部分：

★考核知识点： 基础知识

参见讲稿章节： 1.2

简答题：阐述一下字节存储顺序。

参见讲稿章节： 1.3

简答题：简单地说，虚拟内存的实现方法和过程。

★考核知识点： OllyDdg 调试器

参见讲稿章节： 2.1

填空题：内存数据窗口输入（　），即可查看 rdx 指向的字符串。

填空题：在反汇编窗口选中任意的寄存器、地址、函数，按（）　，也可以跳转到相应的目标地址处。

填空题：通过（ ）菜单可以打开其他子窗口，例如反汇编窗口、寄存器窗口、内存窗口等。

简答题：如何快速回到当前程序领空？

简答题： OllyDbg 如何修改 EIP？

填空题： OllyDbg 的设置项在（）菜单里，有（）选项和（）选项等。这些选项配置都保存在（）文件里。

填空题： UDD文件是 OllyDbg 的工程文件，用于保存当前调试的一些状态，

例如（ ）、（ ）等，以便下次调试时继续使用。

填空题： OllyDbg 调用（ ）函数创建可执行文件的进程。

填空题：附加到一个正在运行的进程， 此刻 OllyDbg 会立即暂停这个程序以及它所有的（ ）；被附加的程序会暂停在 Ntdll.dll 的（ ）处。

填空题：在 OllyDbg 中可以使用 （ ）命令或者 （ ）快捷键来设置 / 消断点。

★考核知识点： OllyDdg 内存断点

参见讲稿章节： 2.5

简答题：内存断点的原理？

★考核知识点：反汇编引擎

参见讲稿章节： 3.2

简答题：比较各种反汇编引擎的优缺点。

简答题： Capstone 介绍。

★考核知识点：反汇编引擎

参见讲稿章节： 3.2

简答题：交叉引用的概念。

附（考核知识点解释）：

交叉引用，在 IDA Pro 中被称为 XREF，可以告诉你函数在何处被调用，或者一个字符串在何处被使用。

★考核知识点：IDA Pro 增强反汇编

参见讲稿章节： 3.9

简答题： FLIRT 介绍。

★考核知识点： 逆向分析技术

参见讲稿章节 :4.1

简答题：什么是逆向分析技术？

★考核知识点： 序列号保护方式

参见讲稿章节： 5.1

简答题：序列号保护机制都有哪些种？

简答题：序列号保护机制的逆向分析方法。

★考核知识点： 时间限制

参见讲稿章节： 5.3

简答题：时间限制程序大概分为哪几类，都是什么？

简答题：时间限制的软件， 如果考虑得比较周全， 软件最少要保存哪些时间值？

★考核知识点： 菜单功能限制

参见讲稿章节： 5.4

简答题：菜单功能受限的程序， 其菜单或窗口中的部分选项是灰色的， 无法使用。这种功能受限的程序一般分哪几种？

★考核知识点： 内核空间、权限空间布局

参见讲稿章节： 6.1

填空题：现代操作系统一般分为（）和（）两部分。

简答题：简要谈谈现代操作系统的应用层和内核层。

★考核知识点：Windows 与内核启动过程

参见讲稿章节： 6.2

简答题： Windows 7 和 Windows XP启动过程的区别。

★考核知识点：Windows R3 与 R0通信

参见讲稿章节： 6.3

填空题：如果想控制系统， 就必须取得 () 特权级， 例如大多数驱动程序就是工作在 () 特权级上的。

简答题：编译好的驱动在系统中如何被加载并执行？

★考核知识点： 内核函数与内核驱动模块

参见讲稿章节： 6.4

简答题：什么是内核函数？

简答题：什么是内核驱动模块？

★考核知识点： 内核对象

参见讲稿章节： 6.5

填空题： TEB的中文全称是。（）

简答题：什么是 TEB?

★考核知识点：SSDT

参见讲稿章节： 6.6

简答题：搭建 WinDbg结合虚拟机的内核调试环境主要哪几步？

★考核知识点：TEB 与 PEB

参见讲稿章节： 6.7

填空题： FS 为段寄存器，当代码运行在（）级时，基地址即为当前线程的线程环境块。 ()

简答题：如果想在调试内核模块的同时调试 R3级的程序，请简述可行方案。

★考核知识点：SEH 的概念及基本知识

参见讲稿章节： 7.3

简答题：回调函数的返回值都有哪些？

★考核知识点： 向量化异常处理

参见讲稿章节： 7.4

简答题： VEH与 SEH的异同？

★考核知识点： 异常处理的实际应用

参见讲稿章节： 7.6

简答题：异常处理程序设计中的注意事项？

★考核知识点：PE 文件格式

参见讲稿章节： 8.1

简答题：什么是基地址？

填空题：在 Windows系统中， PE 文件被系统加载器映射到内存中。每个程序都有自己的虚拟空间，这个虚拟空间的内存地址称为。 （）

填空题：当 PE文件储存在磁盘中时，某个数据的位置相对于文件头的偏移量称为。（）

参见讲稿章节： 8.2

简答题：列举 IMAGE_FILE_HEADER结构都有哪些字段？

填空题：紧跟着 DOSstub 的是（）。它是 PE 相关结构 NT 映像头(IMAGE_NT_HEADERS)的简称。

参见讲稿章节： 8.3

简答题：区块表有什么作用？

填空题：在 PE文件头与原始数据之间存在一个 （）。区块表中包含每个块在映像中的信息，分别指向不同的区块实体。

填空题：区块中的数据逻辑通常是关联的。 PE 文件一般至少有两个区块，一个是代码块，另一个是。 （）

参见讲稿章节： 8.5

填空题：当 PE装载器载入 PE文件时，会检查（）并将相关 DLL映射到进程

地址空间。

填空题： Windows 目录里的应用程序就是典型的绑定输人结构程序，其（）已指向相关 DLL 的函数。