1. windows平台的各种典型调试工具及应用的场景；

1、二进制查看工具  
Winhex/Ultraedit/exeScope（文件结构解析）

2、二进制查壳工具  
PEID/LordPE/DIE（**Detect It Easy ，linux也能用**）/IDA手动分析  
识别加密壳 zprotect、识别压缩壳 UPX。

LordPE官方版是一款集PE文件分析、修改、脱壳功能于一体的PE文件工具

3、静态分析工具  
Windows：IDA（R3）and plungins（插件）

与Vmware联动，调试程序。

与Qemu联动，调试固件。

4、动态分析工具  
Windows：OllyDbg（R3）、WinDbg（R0）  
Linux：gdb（R3）、kgdb（R0）  
VM image（虚拟机）：Vmware、VirtualBox、Qemu。  
注：应用调试为 R3，内核调试为 R0。

5、非主流工具  
符号执行：Angr、Z3  
污点跟踪：Pin、Valgrind、TraintDroid  
模糊测试：FileFuzz、AFL、Trinity、Peach  
Android专用：apktool、Dex2JAR、JD-GUI、SMALI/BAKSMALI  
固件专用：Qemu（动态模拟环境）、KVM（底层虚拟化环境）

**固件调试方案**

IDA+Qemu

KVM只模拟CPU和内存，因此一个客户机操作系统可以在宿主机上跑起来，但是你看不到它，无法和它沟通。于是，有人修改了QEMU代码，把他模拟CPU、内存的代码换成KVM，而网卡、显示器等留着，因此QEMU+KVM就成了一个完整的虚拟化平台。

KVM只是内核模块，用户并没法直接跟内核模块交互，需要借助用户空间的管理工具，而这个工具就是QEMU。KVM和QEMU相辅相成，QEMU通过KVM达到了硬件虚拟化的速度，而KVM则通过QEMU来模拟设备。

1. **各种典型的查壳工具名称及对应的用途；**

ExeInfoPE：查壳程序.它至今依然被更新，可以查看32/64位

PEiD(PE Identifier)是一款著名的查壳工具，只能查看32位应用程序

LoadPE/DIE（**Detect It Easy ，linux也能用**）/IDA手动分析

**3.虚拟化环境的搭建过程中涉及的步骤、原理、工具名称；**

内核调试需要使用一台计算机用于被调试，另一台计算机用于使用调试器来调试前面所述的计算机。为了能在同一台计算机上进行内核调试，通常使用虚拟机来运行被调试的计算机。另外，使用虚拟机运行被调试的机器比较安全，因为虚拟机可以随便的使用，无需担心调试过程中导致的系统损坏。而调试器所在的计算机通常使用物理机。

双机：一台调试机（主机）+一台被调试机（虚拟机）+一个串口线。　　注：Windbg安装在调试机上

1. 调试机中（物理机）

1. 启动 WinDbg2. 选择 File- - >Kernel Debug 3. 设置 COM 标签

2.为虚拟机添加串行端口（注意查看是否已有串口存在，如果有需要移除，否在会导致windebug与虚拟机连接不上）， 串行端口类型选择 “输出到命名管道”，命名管道就是在windbg里的kernel debugging，COM标签里设置的port

3.打开虚拟机中Windows的系统盘，在文件夹选项中设置为“显示所有文件”，“不隐藏系统保护的文件”，然后可以在系统盘下看到一个boot.ini文件。因为在boot.ini中设置的debug输出端口为com1，所以在虚拟操作系统的设备管理器中设置COM1端口的波特率为115200。 （与kernel debugging，COM标签里设置的port里的设置相一致）

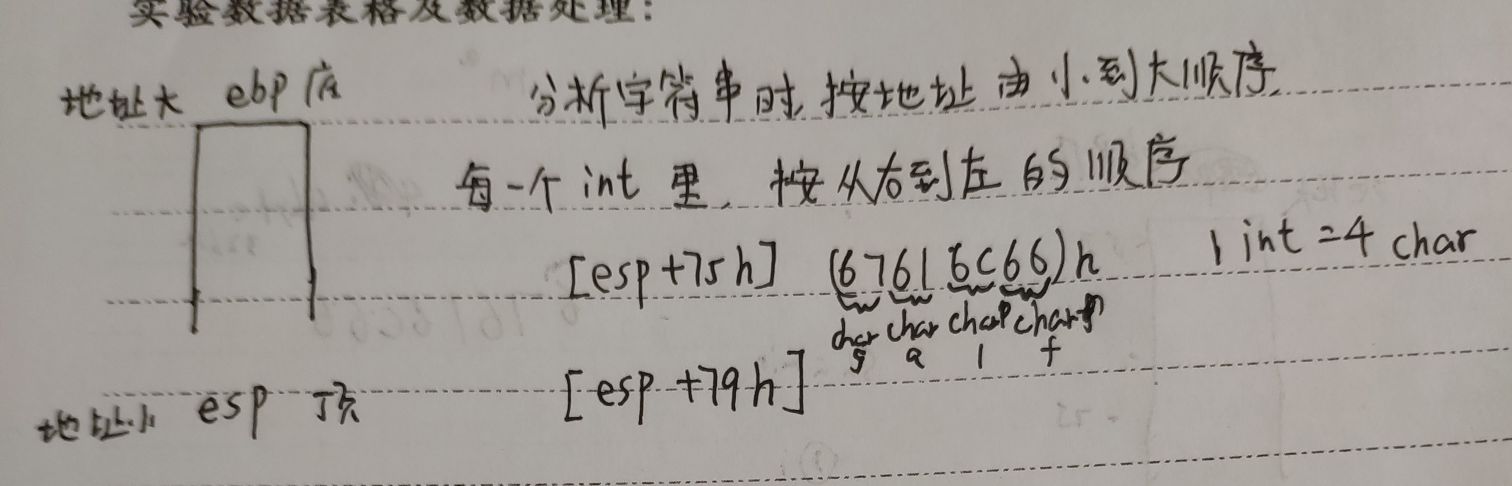
至此虚拟机设置完毕，重启时即可使用windebug进行连接。

4.重启虚拟机，进入调试模式，修改管理员权限CMD，把debug权限打开，这是，就可用windbg对虚拟机进行调试了。

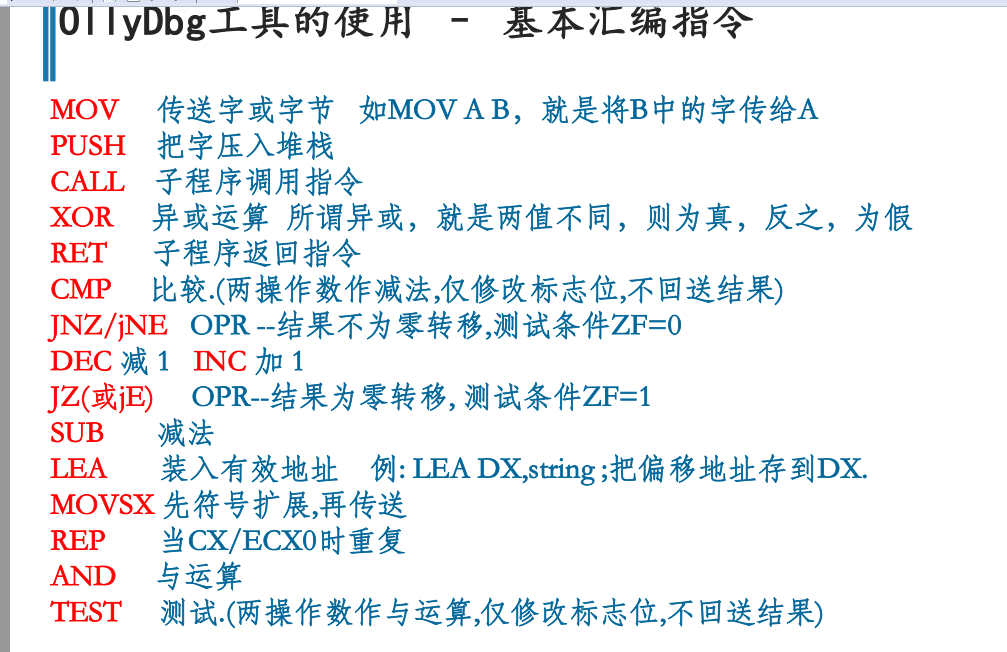


**4.crackme分析的入手方法；**

**每一位对应什么值**

****

**翻译汇编语意**

****

****

**crackme入口，有哪些常见的入口函数**

GetDlgItemText是C++中的函数，调用这个函数以获得与对话框中的控件相关的标题或文本。

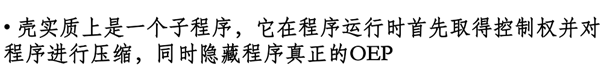
MessageBox 函数用于创建、显示并操作一个消息对话框。

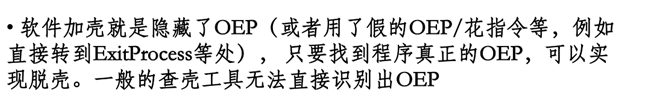
GetWindowTextA

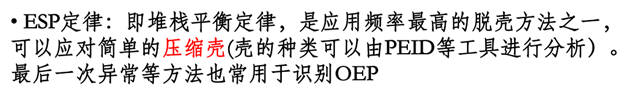
ollydbg找到的，不一定是真的OEP

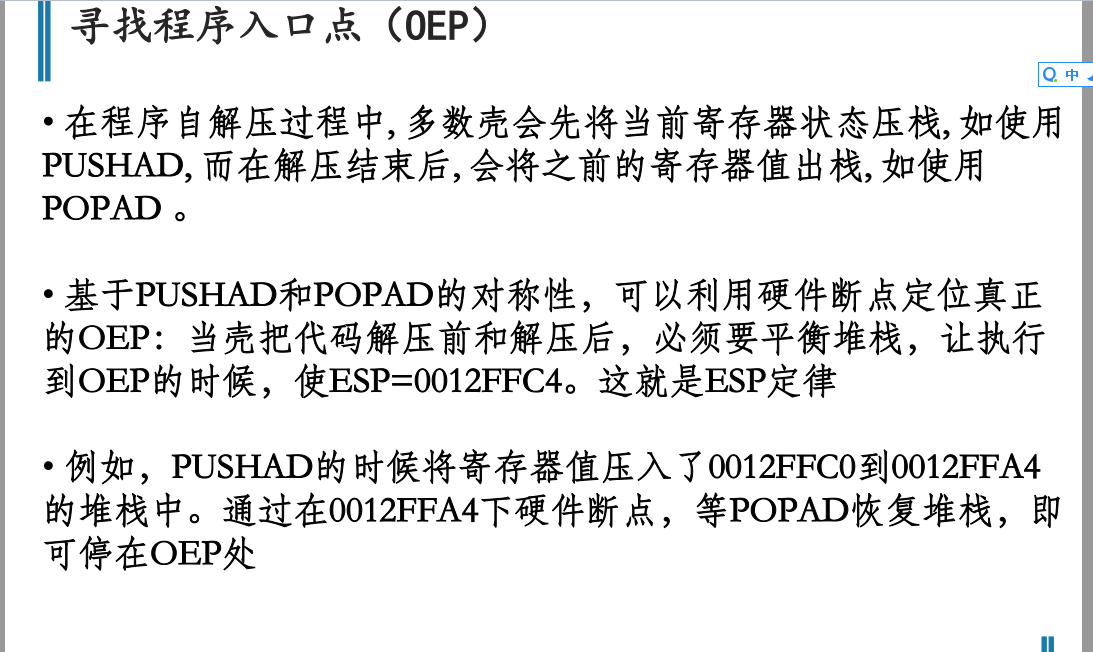
**5.压缩壳的脱壳步骤及对应的原理；**

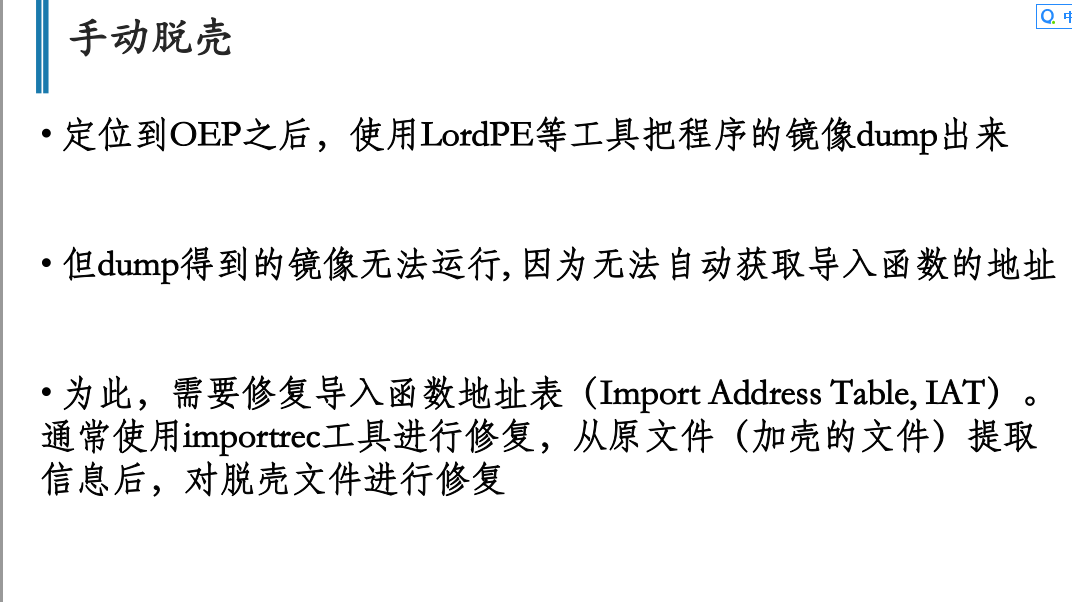
**脱壳过程中，找到oep（**程序的入口点**），还要进行恢复导入地址表IAT**

****

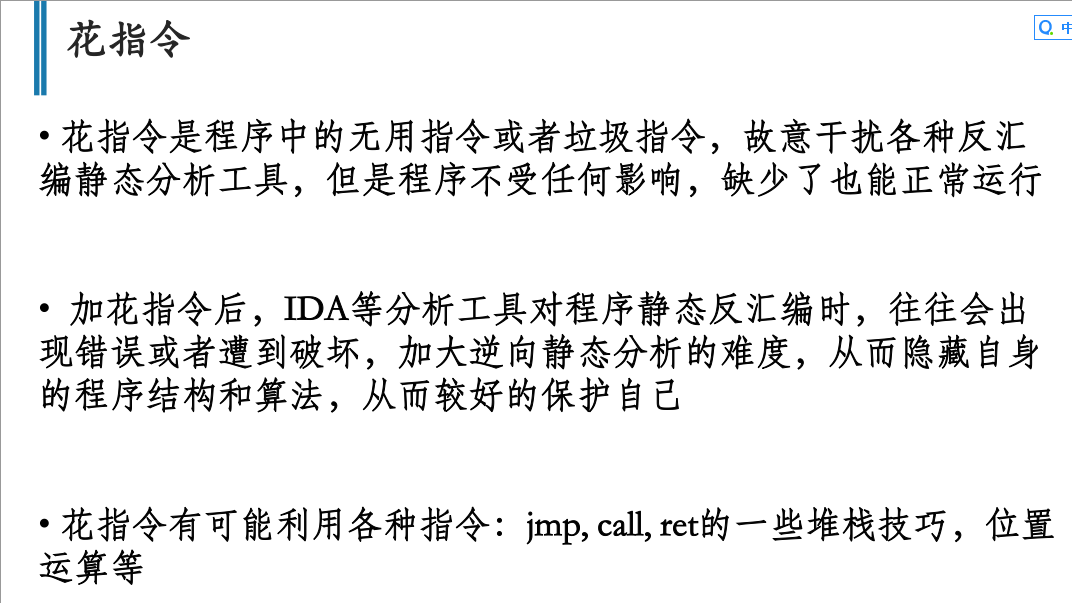
****

****

****

****

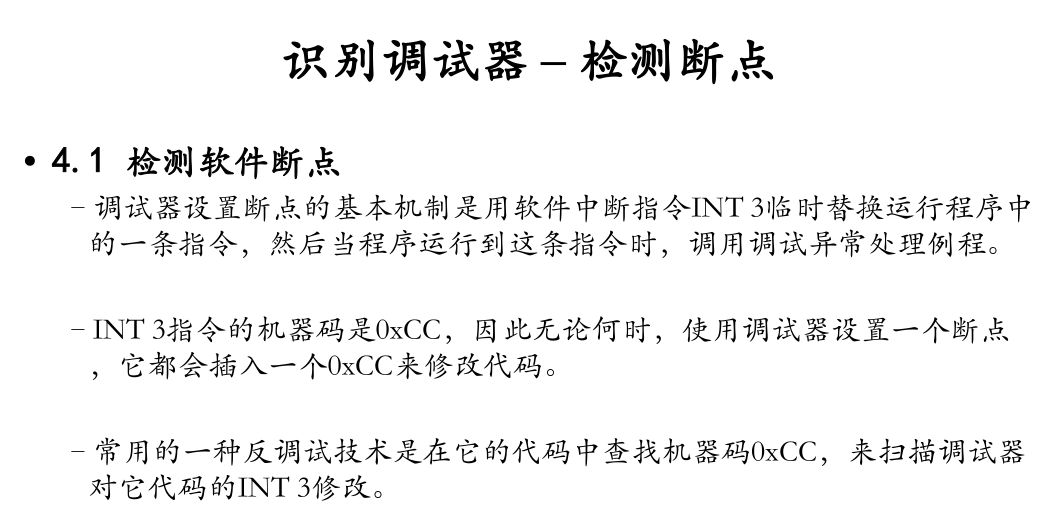
**6.花指令的原理及意义，目的；**

****

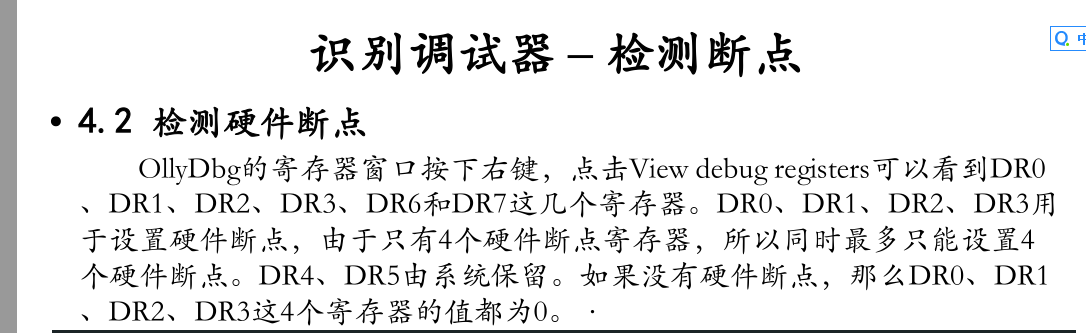
**7.olldbg典型快捷键的名称、功能；**

**填空题，IDA如何识别目标程序的API(导入表)，exe采用什么方式进行upx脱壳（ESP定律法，**单步跟踪法、ESP定律法、内存镜像法、一步到达OEP法、最后一次异常法、模拟跟踪法、“SFX”法。**）**

**8.调试中不同断点的使用原理及使用条件；**

****

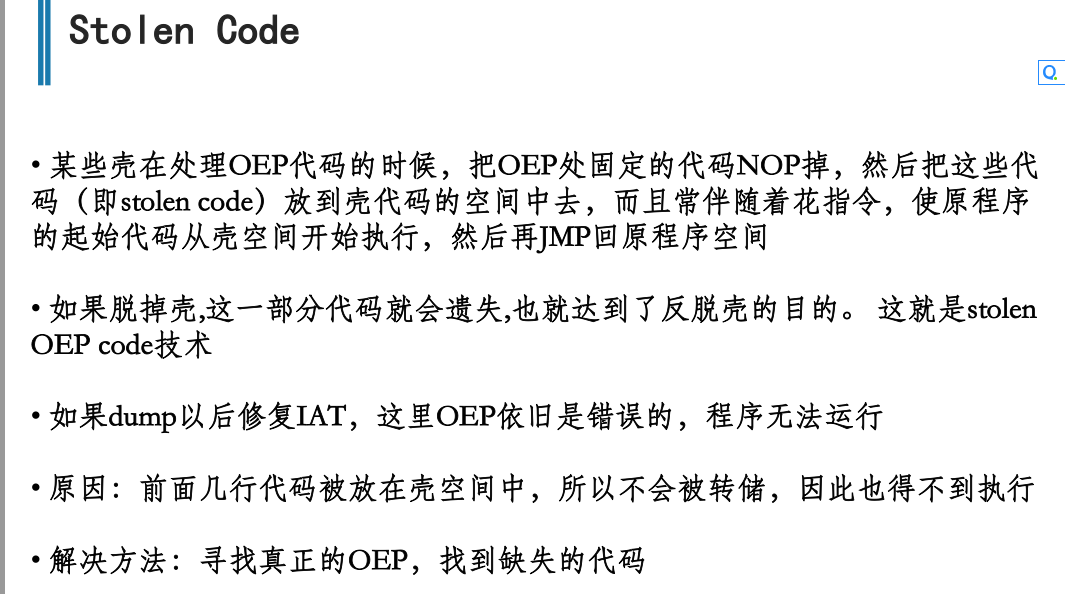
**条件：通常只能设在 RAM 运行的代码上。**

****

调试寄存器组,Dr0 ~ Dr7。Dr0,Dr1,Dr2,Dr3是用于设置硬件断点的,由于只有4个硬件断点寄存器,所以同时最多只能设置4个硬件断点。产生的异常是STATUS\_SINGLE\_STEP(单步异常)。Dr4,Dr5是系统保留的。Dr7是一些控制位,用于控制断点的方式,Dr6是用于显示哪个硬件调试寄存器引发的断点,如果是Dr0 ~ Dr3的话,相应位会被置1。即如果是Dr0引发的断点,则Dr6的第0位被置1,如果是Dr1引发的断点,则Dr6的第1位被置1,依次类推。

条件：需要目标 CPU 的硬件支持

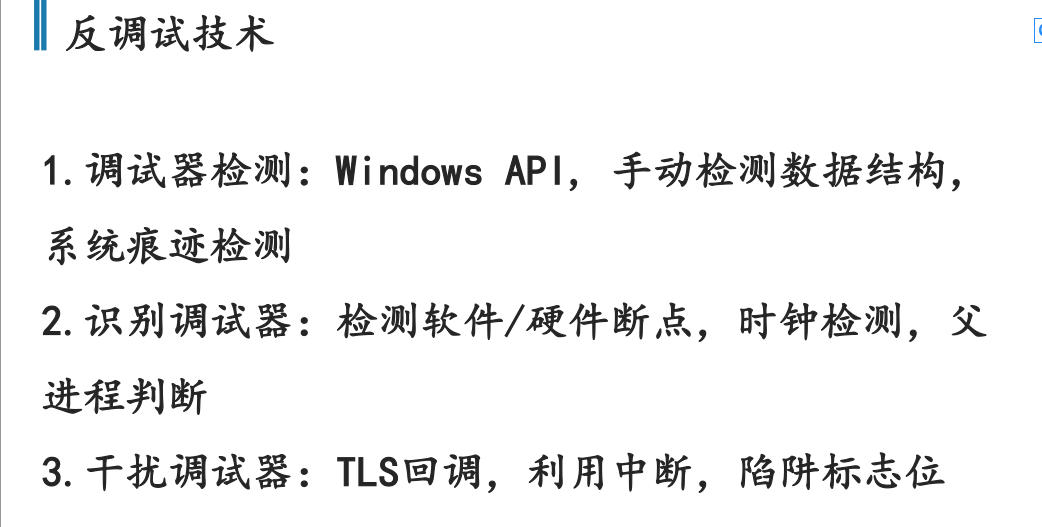
**9.简单了解一下壳偷代码的基本原理。**

****

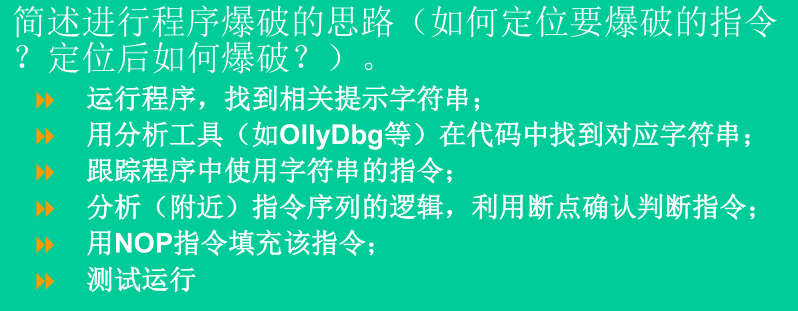
**10.一个简单的crackme实例计算题目；**

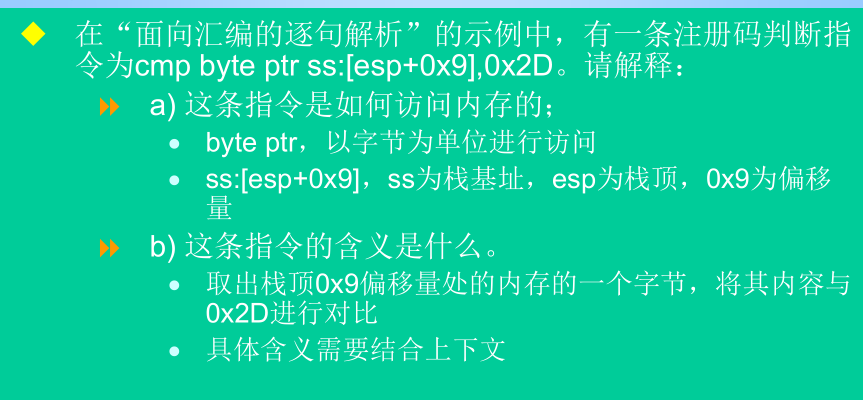
**11.目标程序呈现出各种典型特点，给出对应的分析及破解思路；**

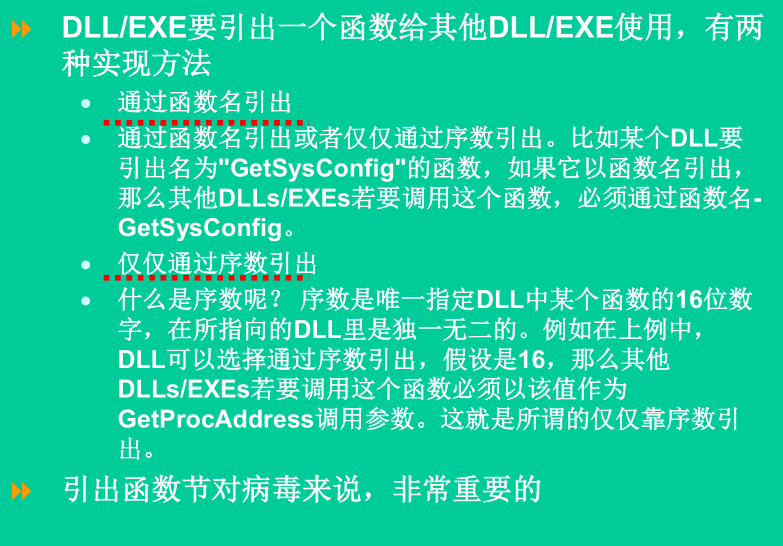
**12.反调试技术的分类及各类的具体技术名称及原理。**

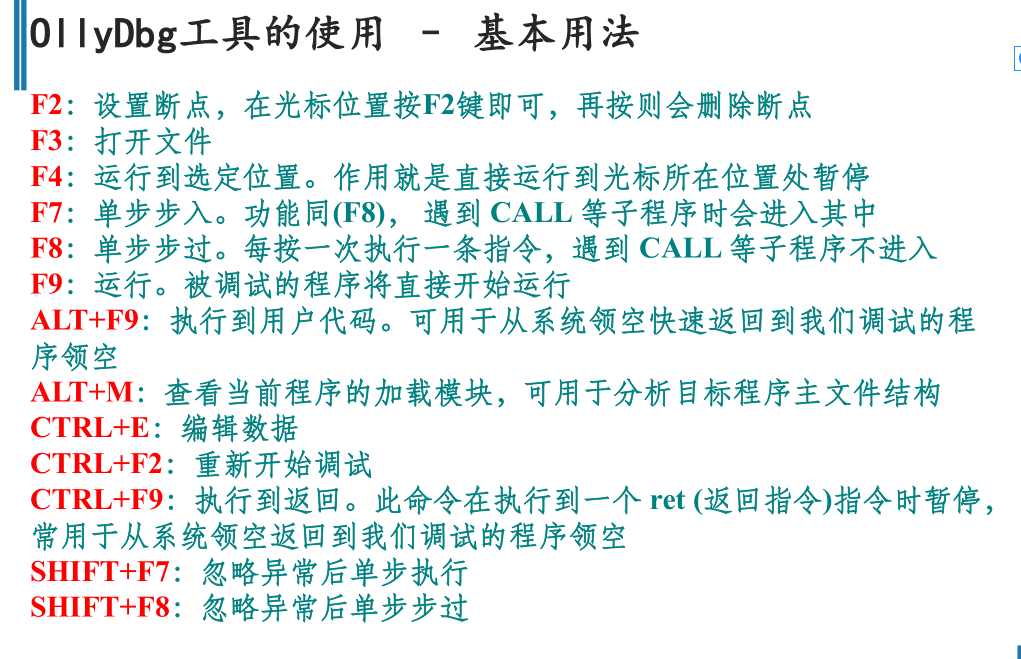
****

**windows api:**

****

****

****

****