软件安全
Software Security

国家网络安全学院 傅建明

Jmfu_whu@126.com

教学大纲

- 软件安全的安全属性和安全威胁
- 计算机（软件）的基础知识
- 恶意代码的原理（木马和Rootkit）
- 恶意代码的对抗
- 软件缺陷（栈、堆、Web等）
- 补丁分析与漏洞挖掘

第四讲 计算机木马和Rootkit

1、计算机木马

2、Rootkit

4.1 特洛伊木马

4.1.1 木马概述

4.1.2 木马的原理及其实现技术

4.1.3 远程控制型木马

4.1.4 木马的预防和清除

4.1.5 木马示例分析

4.1.1 木马概述

- 木马：全称为特洛伊木马，来源于古希腊神话

4.1.1 木马概述

- 通过**欺骗或诱骗的方式安装**，并在用户的计算机中隐藏以实现控制用户计算机的目的。
- 具有远程控制、信息窃取、信息传输等功能的恶意代码；
- 特点
  - **伪装性、隐藏性**、破坏性、窃密性

4.1.1 木马概述

- 木马的分类
  - 远程控制型木马
    - 流行的远程控制型木马有冰河、网络神偷、广外女生、网络公牛、黑洞、上兴、彩虹桥、PCShare、灰鸽子等。

- 木马的分类
  - 密码发送型木马
    - 搜索敏感信息：内存、临时文件夹、Cache等
    - 密码发送至三方空间，文件服务器、制定邮箱等
  - 键盘记录型木马

4.1.1 木马概述

- 木马的分类
  - 破坏型木马
    - 破坏删除系统中重要文件，导致电脑无法正常运行
  - DoS型木马
    - 肉鸡；邮件爆炸木马
  - FTP型木马
    - 木马程序开启FTP服务，攻击者使用FTP客户端访问

4.1.2 木马的原理及其实现技术

1.木马结构

2.植入方式

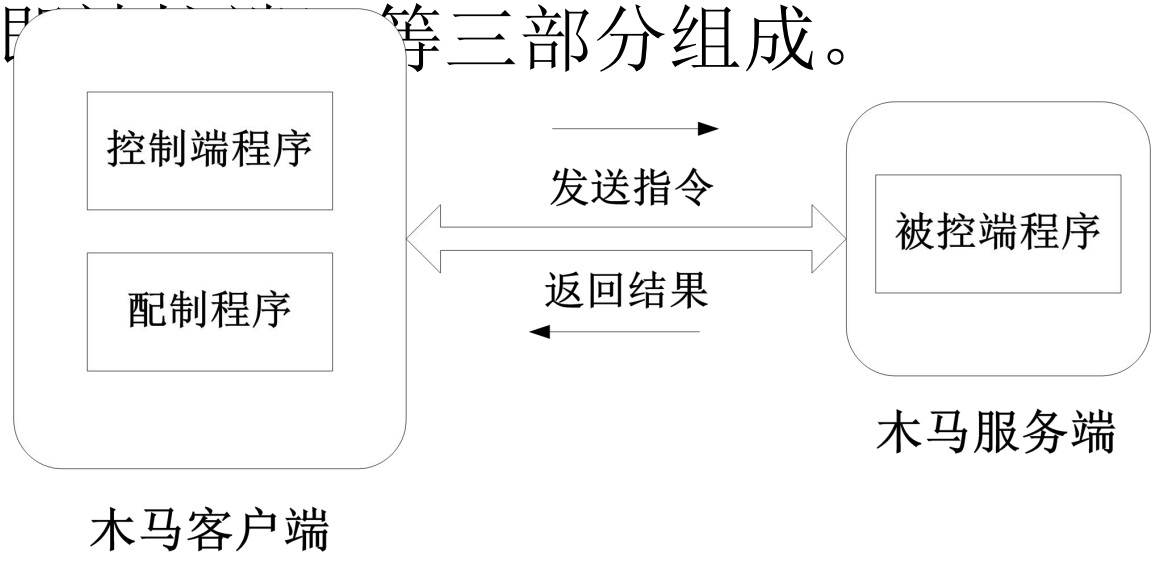
3.连接通信方式

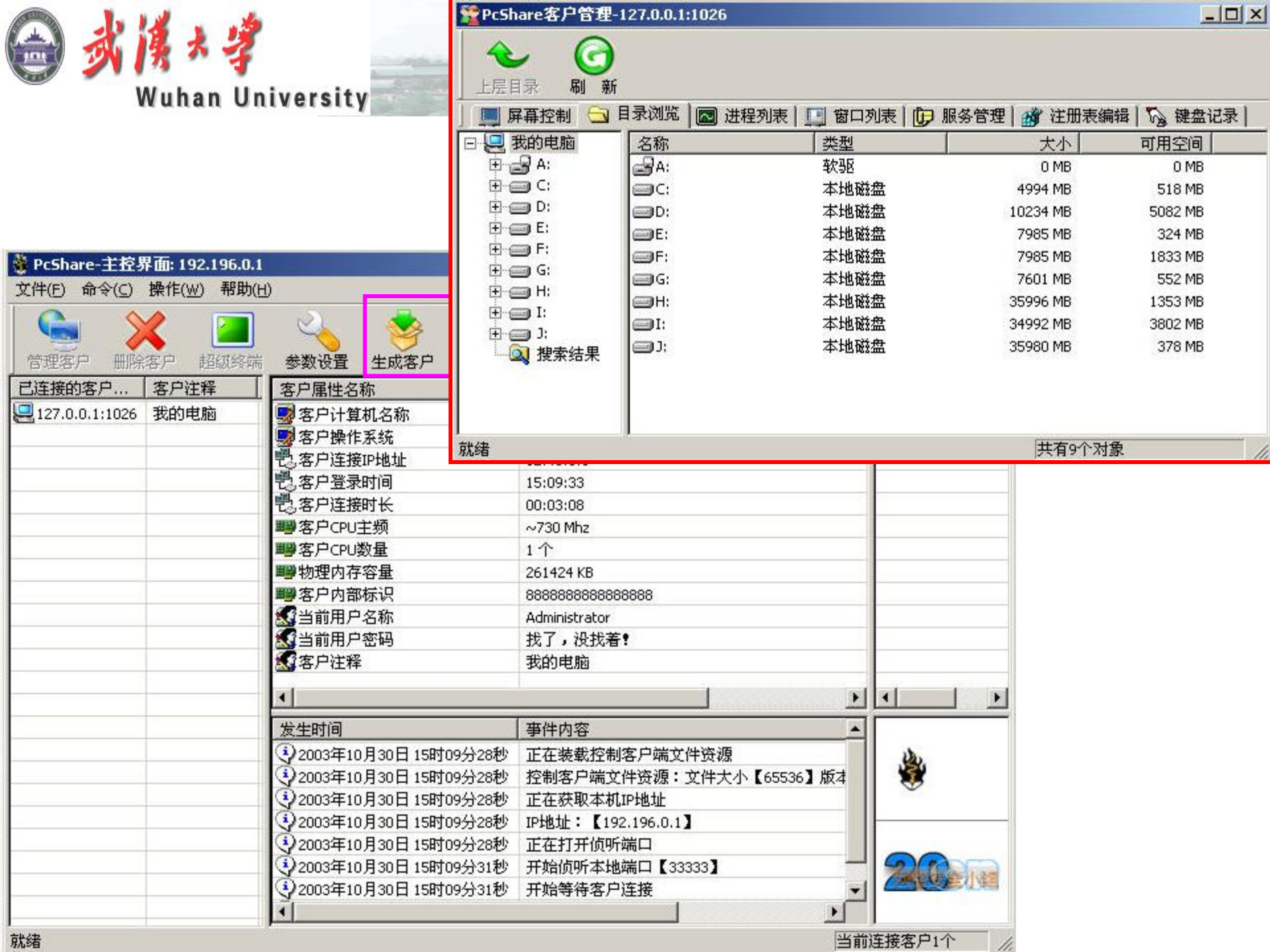
4.自启动技术

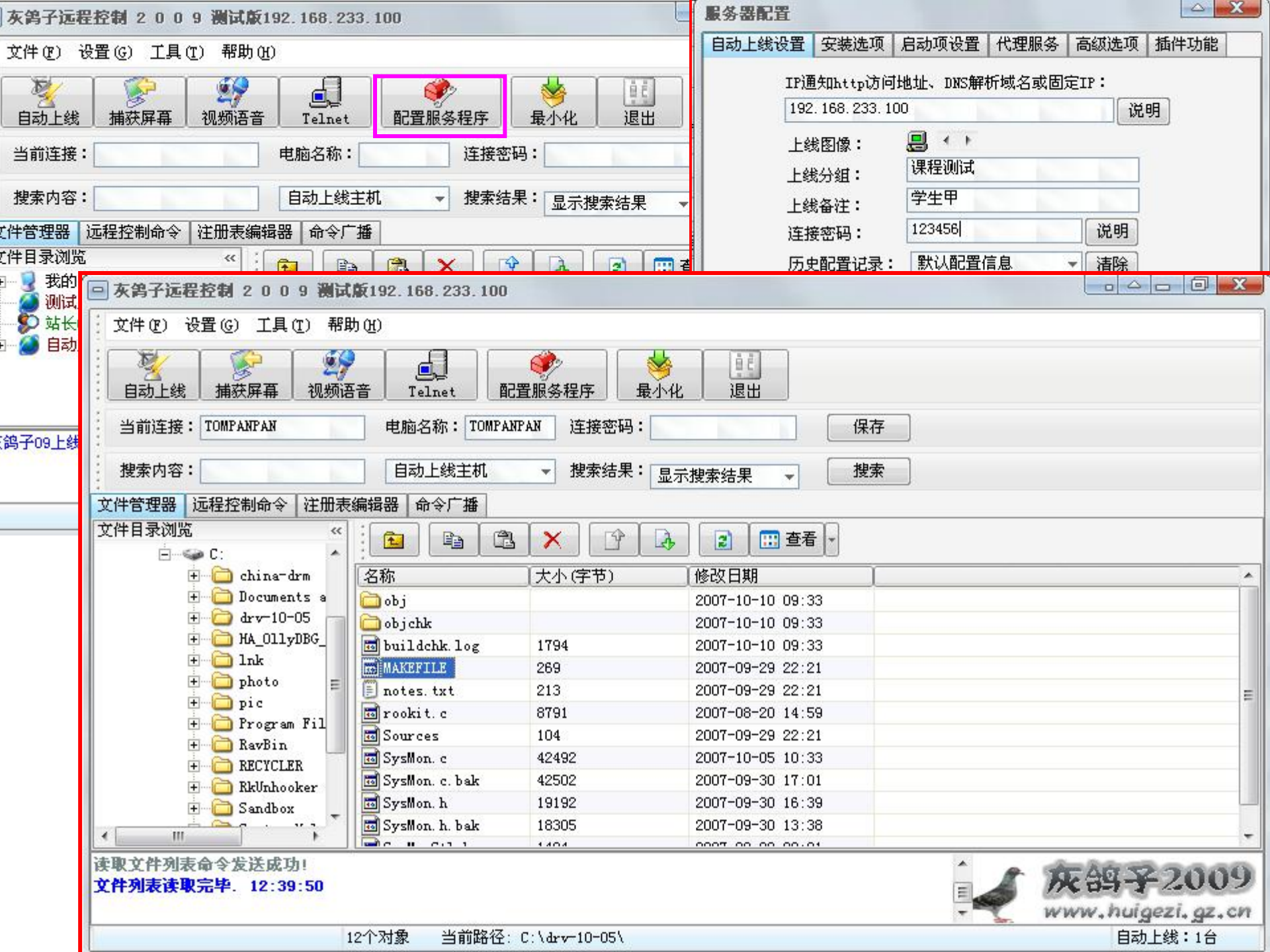
5.隐藏技术

4.1.2 木马的原理及其实现技术

- 1.木马结构
  - 完整的木马一般由木马配置程序、控制端程序（客户端）和木马程序（服务端程序，[illegible]等三部分组成。




武漢大學
Wuhan University
PcShare客户管理-127.0.0.1:1026
上层目录
刷 新
屏幕控制
目录浏览
进程列表
窗口列表
服务管理
注册表编辑
键盘记录
我的电脑
A:
C:
D:
E:
F:
G:
H:
I:
J:
搜索结果
名称 类型 大小 可用空间
A: 软驱 0 MB 0 MB
C: 本地磁盘 4994 MB 518 MB
D: 本地磁盘 10234 MB 5082 MB
E: 本地磁盘 7985 MB 324 MB
F: 本地磁盘 7985 MB 1833 MB
G: 本地磁盘 7601 MB 552 MB
H: 本地磁盘 35996 MB 1353 MB
I: 本地磁盘 34992 MB 3802 MB
J: 本地磁盘 35980 MB 378 MB
就绪
共有9个对象
PcShare-主控界面: 192.196.0.1
文件(F) 命令(C) 操作(W) 帮助(H)
管理客户
删除客户
超级终端
参数设置
生成客户
已连接的客户...
客户注释
127.0.0.1:1026
我的电脑
客户属性名称
客户计算机名称
客户操作系统
客户连接IP地址
客户登录时间 15:09:33
客户连接时长 00:03:08
客户CPU主频 ~730 Mhz
客户CPU数量 1 个
物理内存容量 261424 KB
客户内部标识 8888888888888888
当前用户名称 Administrator
当前用户密码 找了，没找着！
客户注释 我的电脑
发生时间
事件内容
2003年10月30日 15时09分28秒 正在装载控制客户端文件资源
2003年10月30日 15时09分28秒 控制客户端文件资源：文件大小【65536】版本
2003年10月30日 15时09分28秒 正在获取本机IP地址
2003年10月30日 15时09分28秒 IP地址：【192.196.0.1】
2003年10月30日 15时09分28秒 正在打开侦听端口
2003年10月30日 15时09分31秒 开始侦听本地端口【33333】
2003年10月30日 15时09分31秒 开始等待客户连接
就绪
当前连接客户1个


灰鸽子远程控制 2 0 0 9 测试版192.168.233.100
文件(F) 设置(G) 工具(T) 帮助(H)
自动上线
捕获屏幕
视频语音
Telnet
配置服务程序
最小化
退出
当前连接：
电脑名称：
连接密码：
搜索内容：
自动上线主机
搜索结果：
显示搜索结果
文件管理器
远程控制命令
注册表编辑器
命令广播
文件目录浏览
服务器配置
自动上线设置
安装选项
启动项设置
代理服务
高级选项
插件功能
IP通知http访问地址、DNS解析域名或固定IP：
192.168.233.100
说明
上线图像：
上线分组：
课程测试
上线备注：
学生甲
连接密码：
123456
历史配置记录：
默认配置信息
清除
灰鸽子远程控制 2 0 0 9 测试版192.168.233.100
文件(F) 设置(G) 工具(T) 帮助(H)
自动上线
捕获屏幕
视频语音
Telnet
配置服务程序
最小化
退出
当前连接：TOMPANPAN
电脑名称：TOMPANPAN
连接密码：
保存
搜索内容：
自动上线主机
搜索结果：显示搜索结果
搜索
文件管理器
远程控制命令
注册表编辑器
命令广播
文件目录浏览
查看
C:
china-drm
Documents a
drv-10-05
HA_OllyDBG_
lnk
photo
pic
Program Fil
RavBin
RECYCLER
RkUnhooker
Sandbox
名称
大小(字节)
修改日期
obj 2007-10-10 09:33
objchk 2007-10-10 09:33
buildchk.log 1794 2007-10-10 09:33
MAKEFILE 269 2007-09-29 22:21
notes.txt 213 2007-09-29 22:21
rookit.c 8791 2007-08-20 14:59
Sources 104 2007-09-29 22:21
SysMon.c 42492 2007-10-05 10:33
SysMon.c.bak 42502 2007-09-30 17:01
SysMon.h 19192 2007-09-30 16:39
SysMon.h.bak 18305 2007-09-30 13:38
读取文件列表命令发送成功!
文件列表读取完毕. 12:39:50
灰鸽子2009
www.huigezi.gz.cn
12个对象
当前路径: C:\drv-10-05\
自动上线：1台

4.1.2 木马的原理及其实现技术

- 2.木马的植入方式
  - 网页脚本植入
    - 网页挂马，自动下载安装（MS06014,MS10002）
  - 电子邮件植入
    - 附件形式，打开附件被植入
    - 电子邮件与恶意网页相结合，即使不打开附件，选中就会被植入（以HTML格式发送，eg:求职者）
  - 系统漏洞植入(通过其他病毒或蠕虫植入)
    - MS03026 MS05051 MS08067

4.1.2 木马的原理及其实现技术

- 2.木马的植入方式
  - 伪装欺骗植入
    - 更改后缀名、图标伪装成合法程序
  - 捆绑植入
    - Exe捆绑、文档嵌入、多媒体文件、电子书植入
  - 其他
    - 社会工程，用户习惯

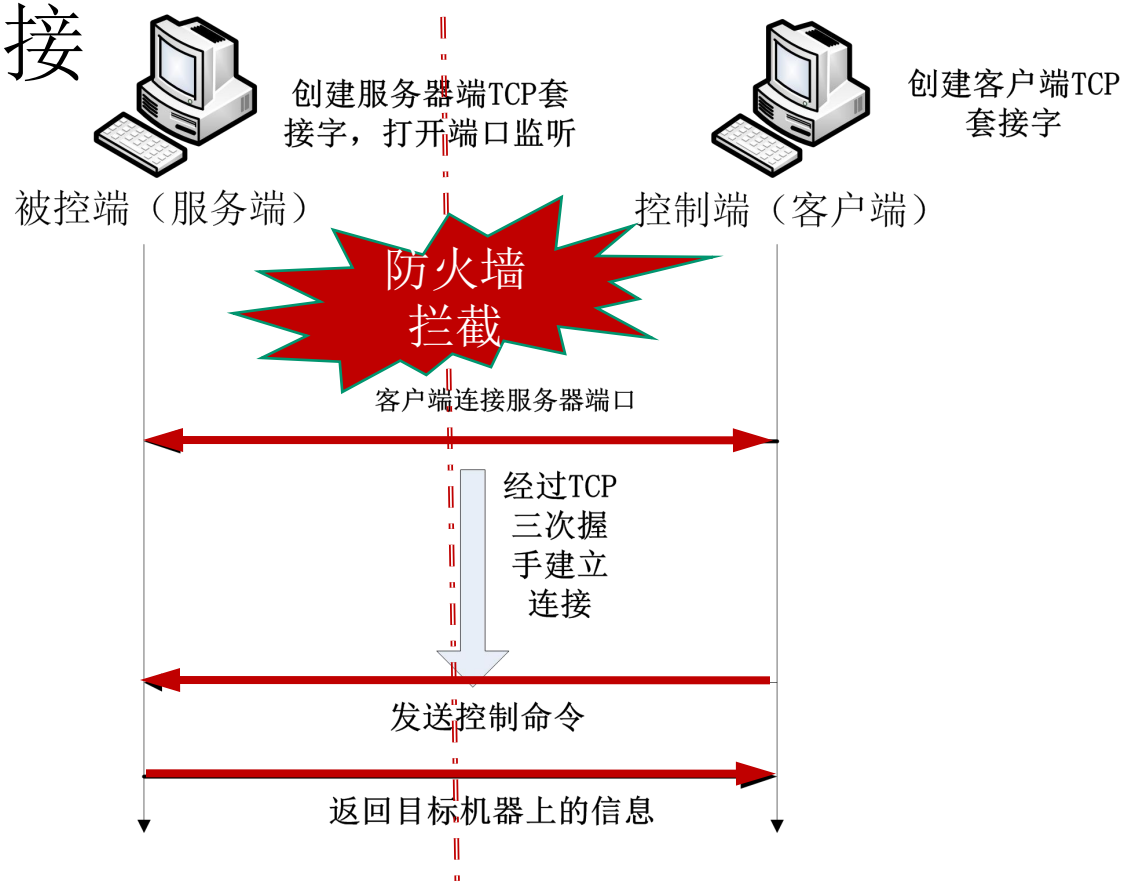
4.1.2 木马的原理及其实现技术

- 3.木马连接方式
  - 获取连接信息
    - 服务端通知客户端自身的IP地址、端口等信息、第三方控件信息
    - 服务端不主动通知，则客户端主动扫描
  - 建立通信连接
    - 选择可靠的、面向连接的传输层通信协议TCP
      - 正向连接
      - 反向连接

4.1.2 木马的原理及其实现技术

- 正向连接

控制端
主动连接
被控端



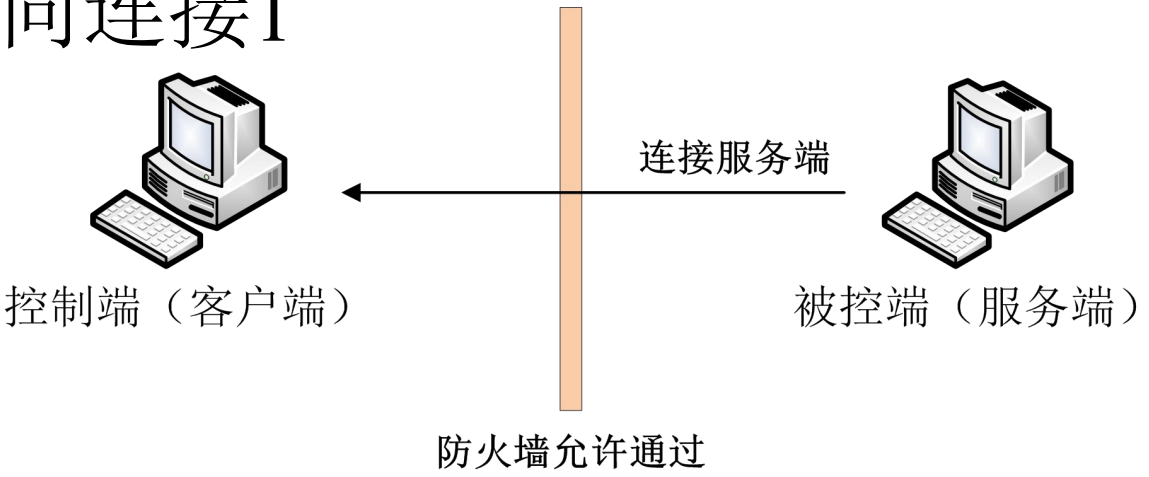
正向连接的优缺点

- 优点：
  - 攻击者无需外部IP地址
  - 木马样本不会泄露攻击者IP地址

- 缺点
  - 可能被防火墙阻挡
  - 被攻击者必须具备外部IP地址
  - 定位被攻击者相对困难
    - 目标IP经常变化
    - 目标主机何时上线？

4.1.2 木马的原理及其实现技术

- 反向连接1



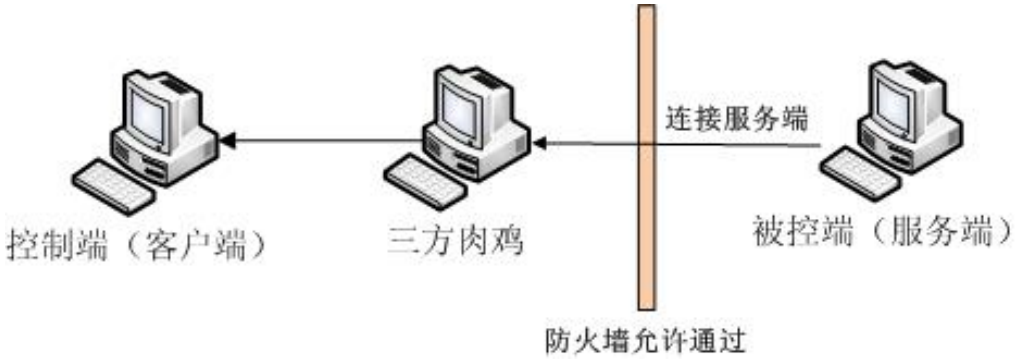
反向连接1的优缺点

- 优点：
  - 通过防火墙相对容易
  - 攻击目标随时上线随时控制
  - 可以控制局域网内的目标

- 缺点：
  - 样本会暴露控制服务器信息
  - 攻击者通常应当具备外部IP的服务器

4.1.2 木马的原理及其实现技术

- 反向连接2



  - 被控端和控制端都和第三方通信
    - 肉鸡、Web服务器
  - 优点
    - 绕过防火墙，自动连接上线，不易被发现（代理）

4.1.2 木马的原理及其实现技术

- UDP通信
  - 和TCP一样也有正向连接反向连接两种方式
  - 负载比TCP少，但是可靠性低

- ICMP通信
  - 监听ICMP报文，以感知
  - ICMP报文是由系统内核或进程直接处理而不是通过端口
  - ICMP_ECHOREPLY报文一般不会被防火墙过滤
  - Ping of Death、ICMP风暴、ICMP碎片攻击

4.1.2 木马的原理及其实现技术

POST /objects/ocget.dll HTTP/1.1
Accept: application/x-cabinet-win32-x86, application/x-pe-win32-x86, application/octet-stream, application/x-setupscript, */*
Content-Type: application/x-www-form-urlencoded Accept-Language: zh-tw Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 compatible; MSIE 6.0; Windows NT 5.1; SV1) Host: activex.microsoft.com Content-Length: 44
Connection: Keep-Alive
Cache-Control: no-cache

GET /objects/index.asp HTTP/1.1
Host: activex.microsoft.com Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Connection: Keep-Alive

4.1.2 木马的原理及其实现技术

- 4.木马自启动
  - 利用注册表
    - 注册表项
    - 注册文件关联
    - 利用映像劫持启动
  - 与其他文件捆绑
  - 利用特定系统文件
    - Autostart、Win.ini、System.ini……
  - IE开始页启动

[HKLM\Software\Microsoft\Windows\CurrentVersion\Run]
[HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce]
[HKLM\Softw
[HKLM\Softw

HKEY_CURRENT_USER\Software\Microsoft\Command Processor
AutoRun REG_SZ “xxx.exe”

4.1.2 木马的原理及其实现技术

- 5.木马隐藏技术
  - 运行形式的隐藏
    - 进程、线程
  - 通信形式的隐藏
    - 端口
    - 通信连接
  - 存在形式的隐藏
    - 木马文件
    - 注册表

4.1.2 木马的原理及其实现技术

- 木马程序的操纵者基本上都可以远程实施计算机用户自己能够实施的操作
- 并且还可以实施一些计算机用户意想不到的功能。

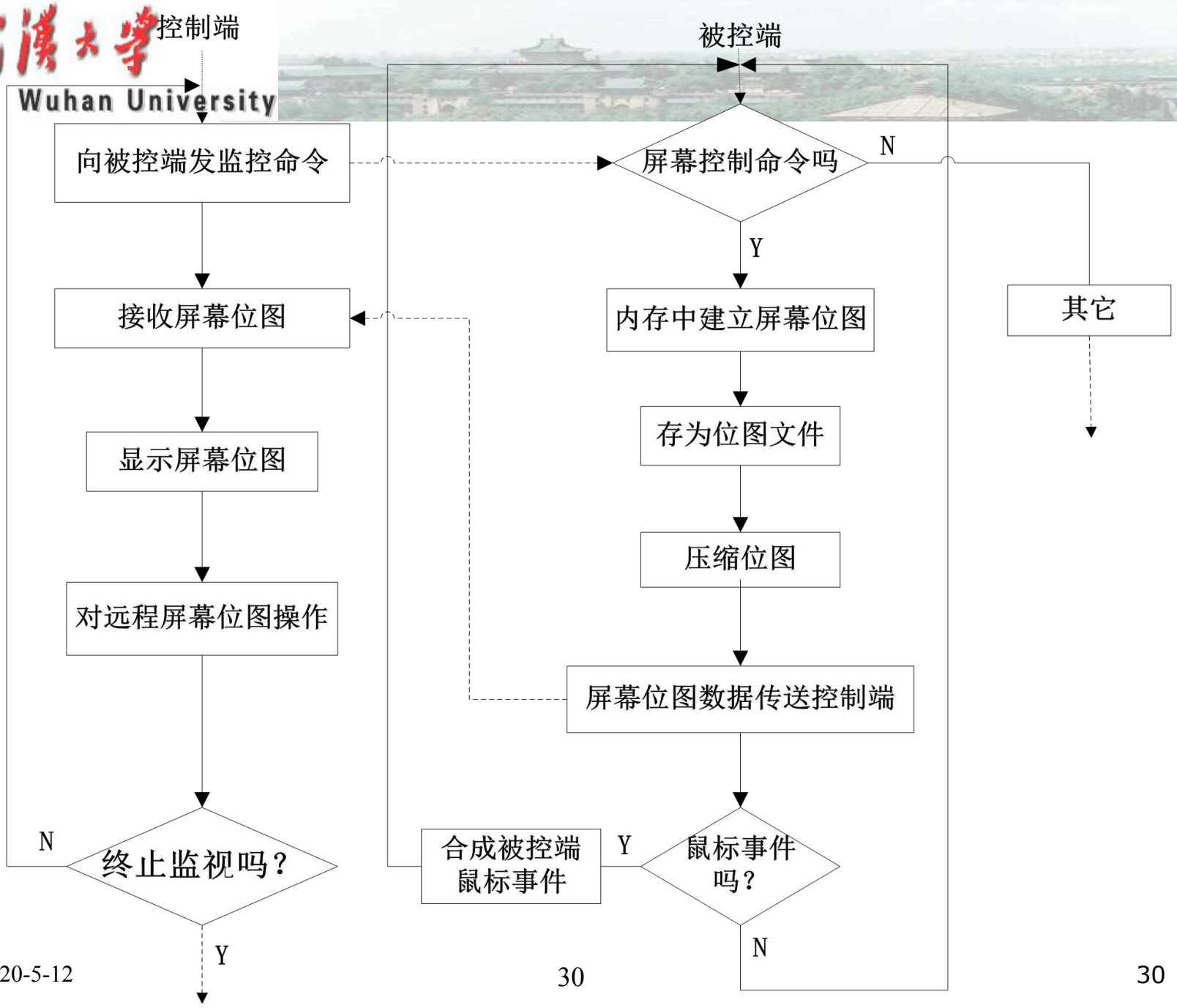
4.1.3 远程控制型木马

- 目的：远程控制、获取有价值信息
- 特点
  - 数量最多
  - 功能强大
  - 隐私窃取

4.1.3 远程控制型木马

- 部分功能
  - 屏幕监控
  - 文件管理
  - 进程管理
  - 服务管理
  - 屏幕截取
  - 语音视频截获
  - 键盘记录

远程屏幕控制功能


控制端
被控端
向被控端发监控命令
屏幕控制命令吗
N
Y
接收屏幕位图
内存中建立屏幕位图
其它
显示屏幕位图
存为位图文件
压缩位图
对远程屏幕位图操作
屏幕位图数据传送控制端
N
终止监视吗？
合成被控端
鼠标事件
Y
鼠标事件
吗？
Y
N

文件管理

- 获取目标的文件系统信息，通常包括如下功能：
  - 浏览各磁盘文件
  - 上传、下载文件
  - 执行文件
  - 删除文件
  - 修改文件信息（如时间）

进程管理

- 查看、结束或者暂停目标系统进程

- 作用：
  - 查看目标系统的环境信息
    - 安装了哪些软件？目前对方正在做什么？
  - 停止或暂停目标系统的相关程序
    - 如反病毒程序

服务管理

- 查看并管理目标系统的服务
  - 创建服务
  - 启动/停止服务
  - 删除服务

- 作用：
  - 查看目标系统的环境信息
    - 安装了哪些软件？启动了哪些服务？
  - 停止或暂停目标系统的相关程序
    - 如反病毒程序

屏幕截取

- 抓取屏幕
  - 单张
  - 多张连续
- 作用：
  - 了解目标主机的当前操作情况

语音视频截获

- 录音
  - 窃取对方谈话信息
  - 窃取对方对外语音通话（如QQ、SKYPE等）
- 摄像头
  - 打开摄像头（了解对方现场环境）
  - 摄像录制（敲诈…）

键盘记录

- 获取目标电脑中的键盘击键信息
  - 用户名、密码信息
    - QQ、邮箱、网银、网上证券、网络游戏、支付宝…
  - 聊天信息
    - 部分木马支持中文汉字记录

窗口管理

- 查看目标主机目前开启了哪些窗口
  - 了解目标用户正在做什么？

木马的关键

- 功能强大（可不断扩充）
- 高效性
- 稳定性
- 木马的自更新问题
- 木马本身存在的问题（反制）
- 木马面临的免杀问题
  - 木马是否容易被检测？
  - 特征值、通用主机行为、异常的通信流量…

4.1.4 木马的预防和清除

- 木马防护目的
  - 避免木马进入系统
  - 如果进入系统，避免木马产生危害

木马样本分析

- 在什么环境中对木马进行分析
  - 虚拟机环境(VMWare/VPC/……)
  - 沙箱
- 木马对计算机进行了哪些修改？
  - 进程、文件、注册表、服务、通信…
    - Process Monitor、Process Explorer、IceSword…
    - FileMon/RegMon/RegSnap/TDIMon/netstat/…
    - SSM…

木马样本分析

- 分析目标：
  - 产生了哪些文件、修改了哪些注册表、创建了哪些进程（远程线程、服务）？
  - 有何自我运行保护措施？
  - 有何对抗反病毒软件的手段？
  - 技术上有何创新之处？
  - 该木马有哪些特殊功能？可以实现哪些特殊目的？控制者可能是什么人？
  - 木马的被控制端处于什么位置？
  - 这是一款什么类型的木马？（通用/小范围使用/完全私用）
  - 用什么语言编写的？作者具有哪些编程风格和特点？
  - 如何对该木马进行检测？…

4.1.4 木马的预防和清除

- 木马防护手段
  - 虚拟机、沙箱…
  - 反病毒软件、个人防火墙、主机行为监控软件（如SSM）。
  - 使用安全浏览器和电子邮件客户端工具。
  - 操作系统、应用软件经常打补丁。
  - 不随便打开陌生文件和下载、使用破解软件。
  - 不要浏览不正规网站，以免遭遇网页木马攻击。

4.1.4 木马的预防和清除

- 木马的清除
  - 检测木马对系统所做的修改
    - 注册表
    - 文件
  - 检测方法
    - 端口扫描
      - 哪些端口是非正常开放的
    - 检查网络连接
      - Iceword、Xuetr、netstat –a 远程IP

4.1.4 木马的预防和清除

- 检测方法：
  - 查看进程内存模块
    - 陌生进程
    - 正常进程是否被注入
  - 检查注册表
    - 启动项
  - 查找文件
    - 木马特定文件,Kernl32.exe、sysexpl.exe，创建时间
  - 杀毒软件查杀

4.1.4 木马的预防和清除

- 删除木马文件并恢复其对系统所做的修改
  - 全面清理文件
  - 专杀工具

一些木马防护方案

- 如何对木马进行防护？
- 如何让普通用户可以有效阻挡木马？
  - 安全意识
  - 好用的安全工具
    - 信安竞赛中的一些努力…

4.1.5 木马技术的发展

- 攻与防是道高一尺魔高一丈的相互竞争
- 木马新技术在不断探索中
  - 通信隐藏技术
  - 远程线程注入技术
  - 攻击杀毒软件
  - 穿透防火墙
  - 更加隐秘的加载方式
  - Rootkit技术

4.1.6 木马实例分析

- 当前比较流行的上兴、PCShare、灰鸽子、gh0st等
- 木马演示

4.2 Rootkit

4.2.1 Rootkit概述

4.2.2 Rootkit技术介绍

4.2.3 文件隐藏

4.2.4 进程隐藏

4.2.5 注册表隐藏

4.2.6 端口隐藏

4.2.7 Rootkit示例

4.2.8 Rootkit检测

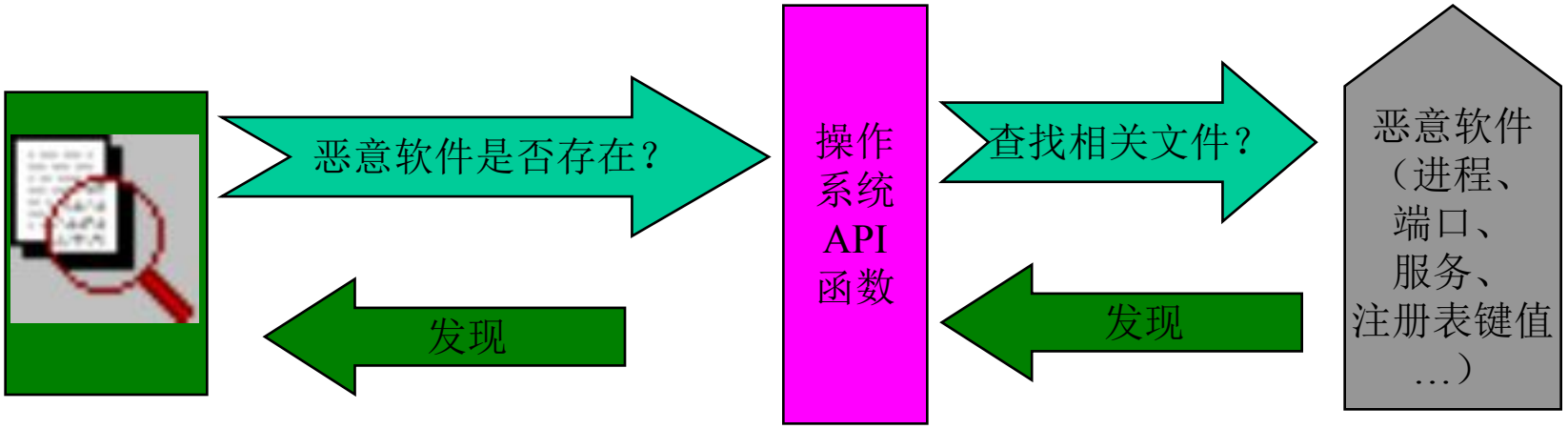
4.2.1 Rootkit概述

- Rootkit定义：
  - 它是由有用的小程序组成的工具包，使得攻击者能够保持访问计算机上具有最高权限的用户”root”。
  - 或者说，Rootkit是能够持久或可靠地、无法被检测地存在于计算机上的一组程序或代码。
  - 关键：无法被检测。
    - 主要用于在计算机上隐藏代码和数据。
    - 因此常被木马程序用来实现隐藏，包括文件隐藏、进程隐藏、注册表隐藏、端口隐藏等。

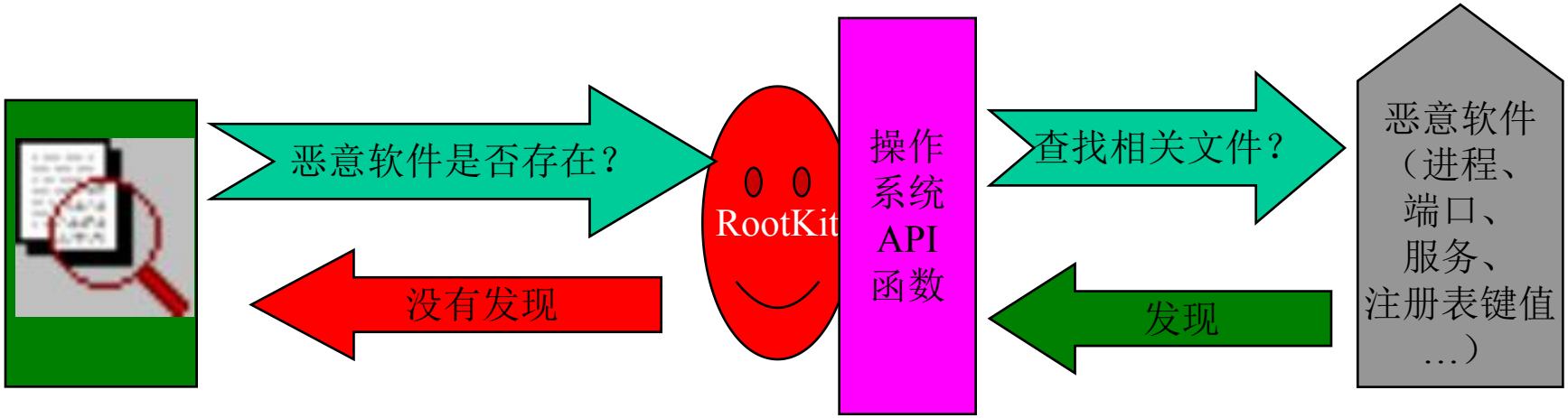
Rootkit概述

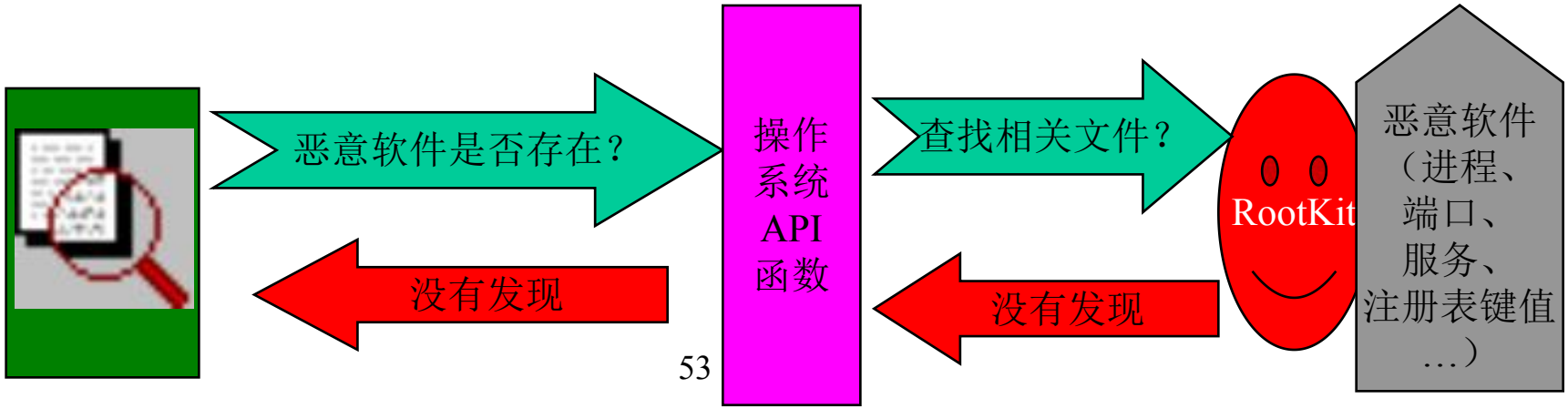
- Rootkit本质
  - Rootkit并不是真的使文件或进程消失，而是采用一定的方法使系统”看不见”。

正常的系统查询过程



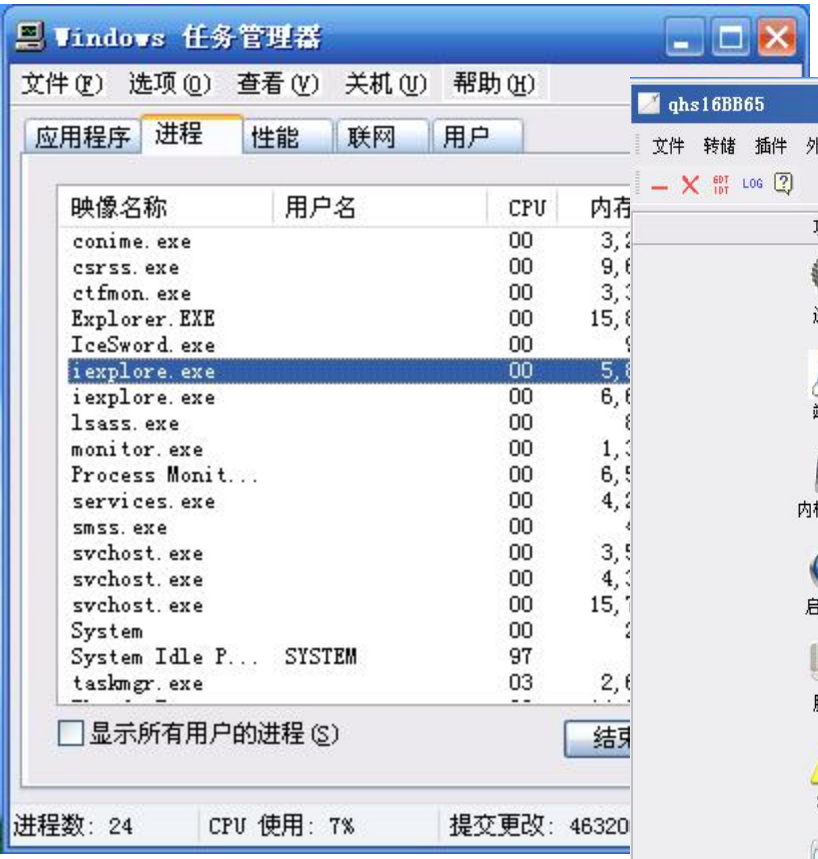
RootKit入侵之后的系统查询过程

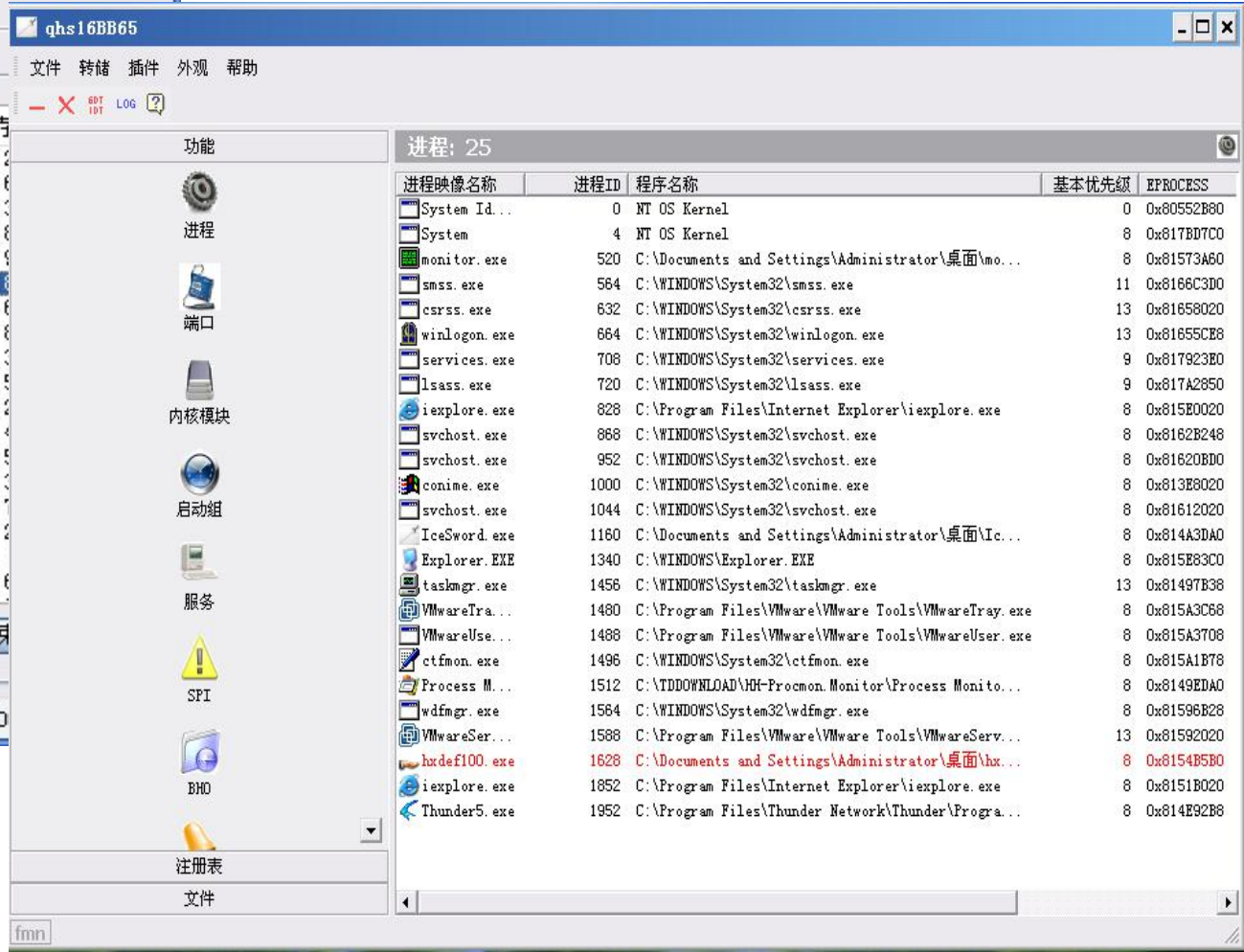




Rootkit隐藏进程示例：

Hacker Defender(hxdef)





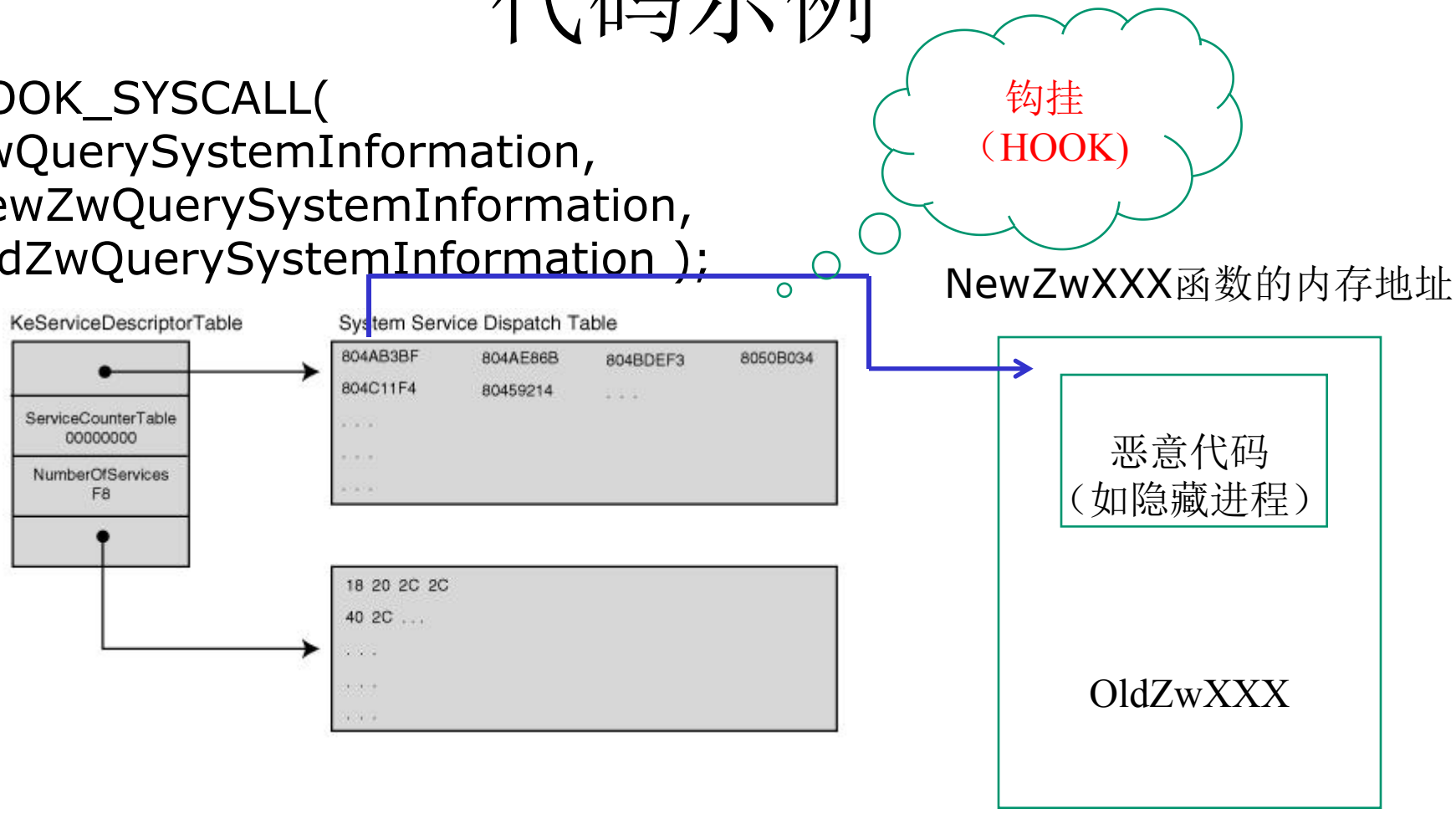
```
typedef
NTSTATUS
(*ZWQUERYSYSTEMINFORMATION)(
        IN    ULONG    SystemInformationClass,
        IN OUT PVOID    SystemInformation,
        IN    ULONG    SystemInformaitonLength,
        OUT   PULONG   ReturnLength OPTIONAL);

 ZWQUERYSYSTEMINFORMATION   OldZwQuerySystemInformation;

NTSTATUS NewZwQuerySystemInformation(
                  IN ULONG SystemInformationClass,
                  IN PVOID SystemInformation,
                  IN ULONG SystemInformationLength,
                  OUT PULONG ReturnLength)
{       NTSTATUS ntStatus;
       DbgPrint("Call ZwQuerySystemInformation\n");    //象征恶意代码
       ntStatus =((ZWQUERYSYSTEMINFORMATION)
                 (OldZwQuerySystemInformation)) (
                  SystemInformationClass,
                  SystemInformation,
                  SystemInformationLength,
                  ReturnLength );
         return ntStatus;
}
```

代码示例

```
HOOK_SYSCALL(
ZwQuerySystemInformation,
NewZwQuerySystemInformation,
OldZwQuerySystemInformation );
```



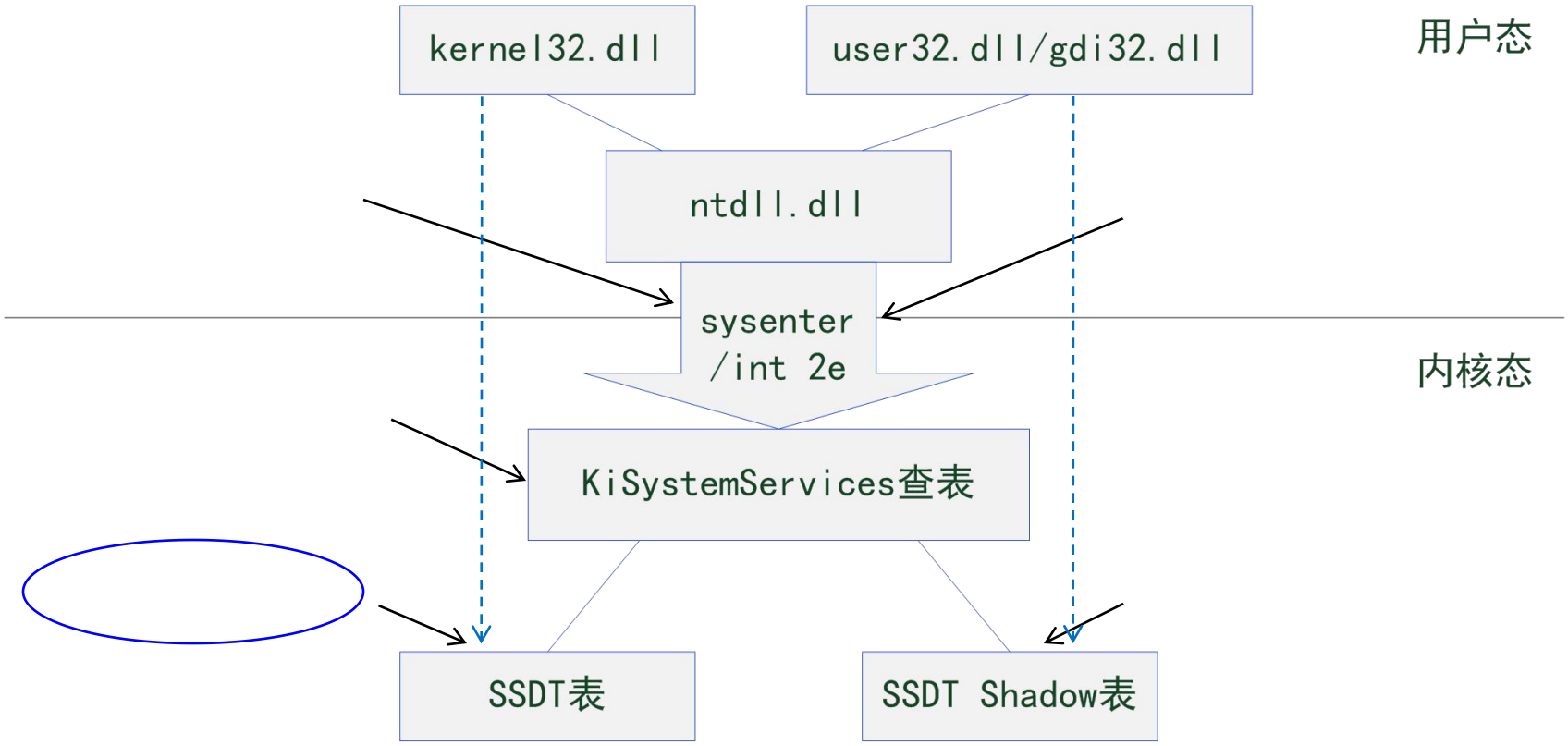
Rootkit概述

- Rootkit实现隐藏，只是采取一定的措施，使系统“看不到”相应的对象。
  - 系统API查询相应的对象，返回信息
  - 而Rootkit拦截系统API的执行过程或返回结果
  - 将返回结果中指定的对象清除，经过处理后的信息

Rootkit的演化

- 第一代Rootkit简单地替换或修改受害者系统上关键的系统文件。
- 第二代的Rootkit大体上基于挂钩技术——通过对已加载的应用程序和一些诸如系统调用表的操作系统部件打内存补丁而改变执行路径。
- 第三代Rootkit技术称为直接内核对象操作，其动态修改内核的数据结构，可逃过安全软件的检测。

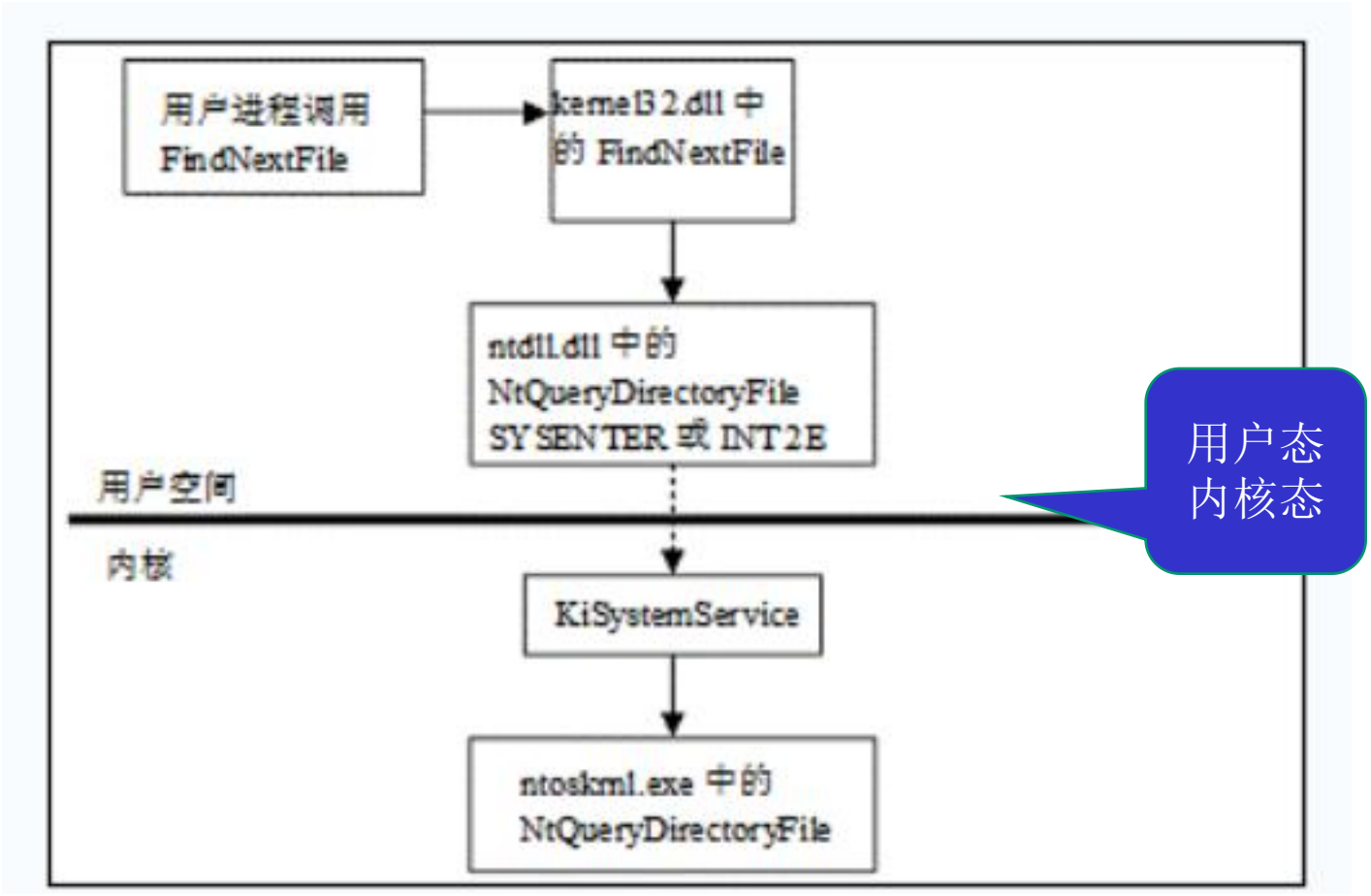
4.2.2 Rootkit技术介绍



Windows API 调用流程

59

4.2.2 Rootkit技术介绍



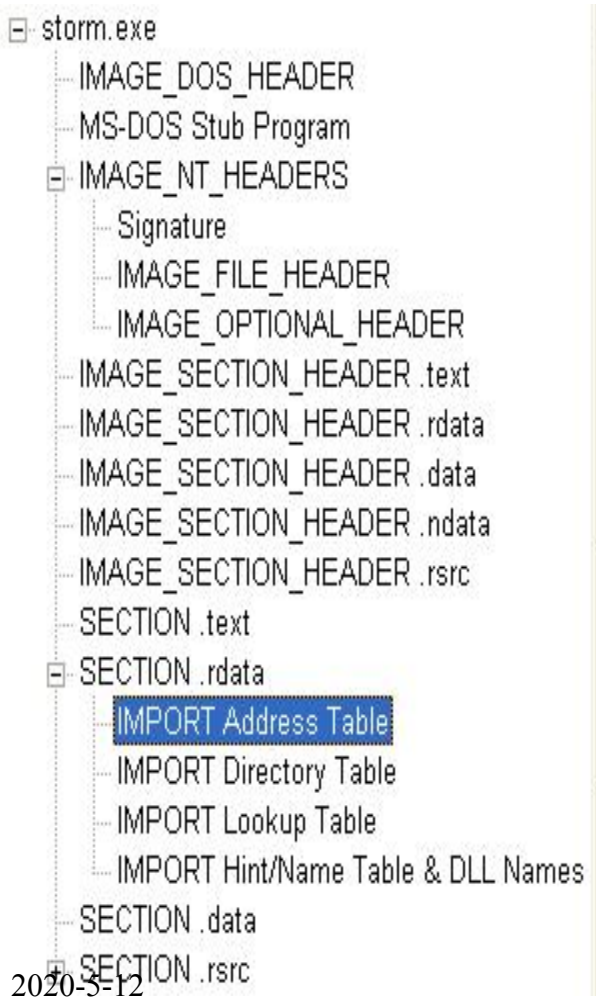
3.直接内核对象操作(DKOM)

1.用户态HOOK

- 用户态Hook是指在操作系统的用户态执行的钩挂，主要是钩挂用户态的API函数。

- 用户态Hook的种类
  - –IAT(Import Address Table, 导入地址表)钩子
  - –内联钩子(Inline HooK)

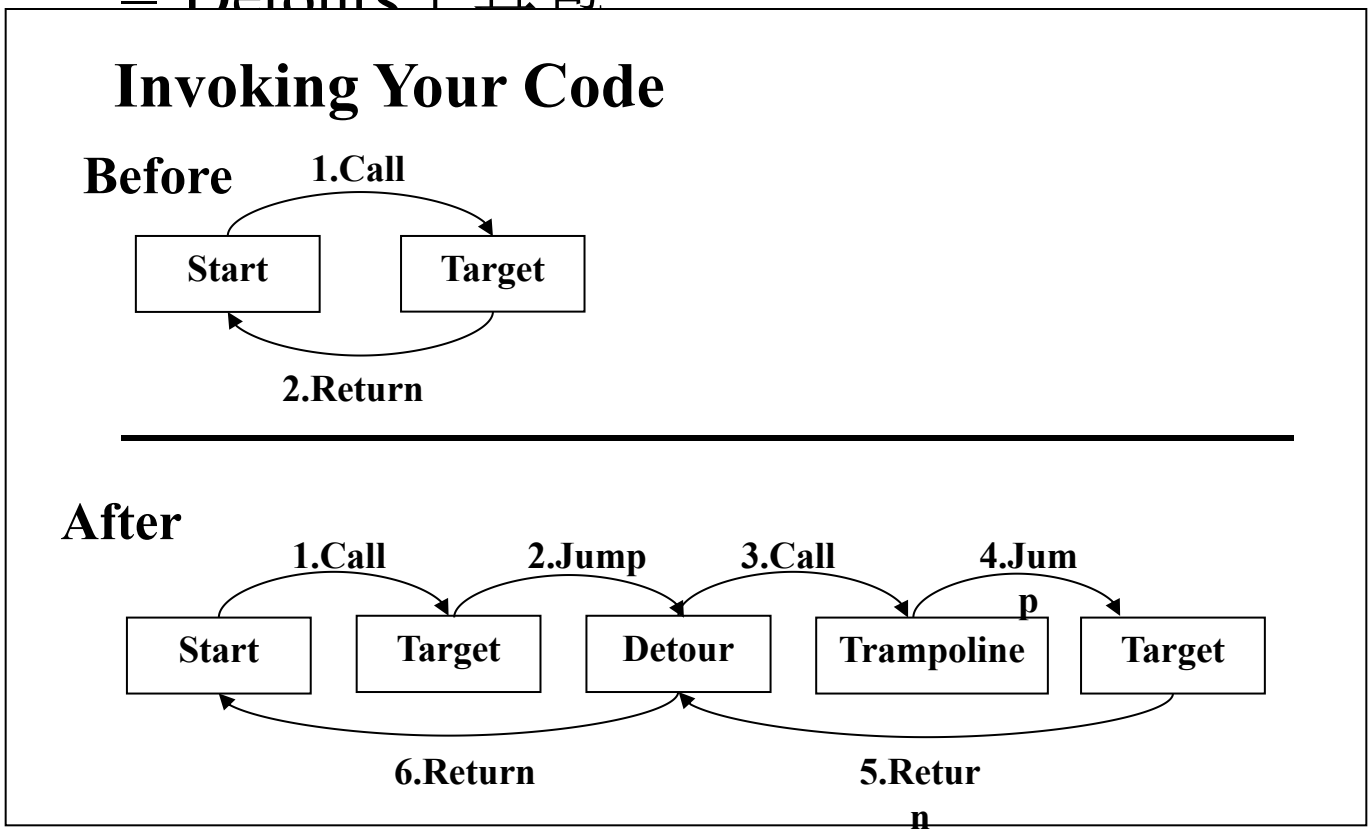
1) IAT钩子(IAT Hook)



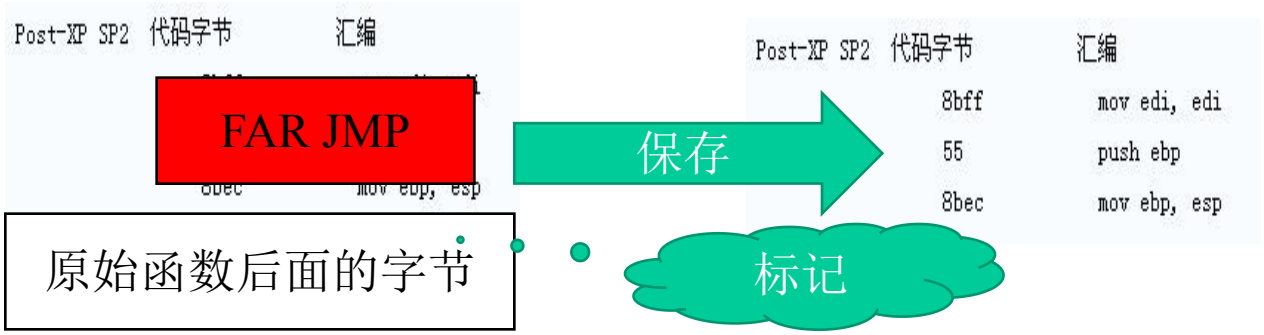
pFile	Data	Description	Value	
00005E00	0000800A	Hint/Name RVA	01EC	RegQueryValueExA
00005E04	0000801E	Hint/Name RVA	01F9	RegSetValueExA
00005E08	00007FFC	Hint/Name RVA	01D5	RegEnumKeyA
00005E0C	00007FEC	Hint/Name RVA	01D9	RegEnumValueA
00005E10	00008072	Hint/Name RVA	01E2	RegOpenKeyExA
00005E14	00008062	Hint/Name RVA	01D0	RegDeleteKeyA
00005E18	00008050	Hint/Name RVA	01D2	RegDeleteValueA
00005E1C	00008042	Hint/Name RVA	01C9	RegCloseKey
00005E20	00008030	Hint/Name RVA	01CD	RegCreateKeyExA
00005E24	00000000	End of Imports	ADVAPI32.dll	
00005E28	000080A4	Hint/Name RVA	0034	ImageList_AddMasked
00005E2C	00008090	Hint/Name RVA	0038	ImageList_Destroy
00005E30	80000011	Ordinal	0011	
00005E34	000080BA	Hint/Name RVA	0037	ImageList_Create
00005E38	00000000	End of Imports	COMCTL32.dll	
00005E3C	00007F46	Hint/Name RVA	0215	SetBkColor
00005E40	00007F36	Hint/Name RVA	016B	GetDeviceCaps
00005E44	00007F26	Hint/Name RVA	008F	DeleteObject
00005E48	00007F10	Hint/Name RVA	0029	CreateBrushIndirect

2)内联钩子(Inline Hook)

- 它重写了目标函数的代码字节.
  - Detours工具包



2)内联钩子(Inline Hook)



2)内联钩子(Inline Hook)

- 内联钩子并不仅局限于用户层，但其原理都是相似的。

2.内核态HOOK

- 与用户态HOOK相比，内核态HOOK有两个重要优势：
  - 内核钩子是全局的；
    - 因为所有进程共享内核地址区
  - 更难以检测
    - 因为若Rootkit和防护/检测软件都处理Ring0级时，Rootkit有一个平等竞赛域，可以在其上躲避或禁止保护/检测软件。

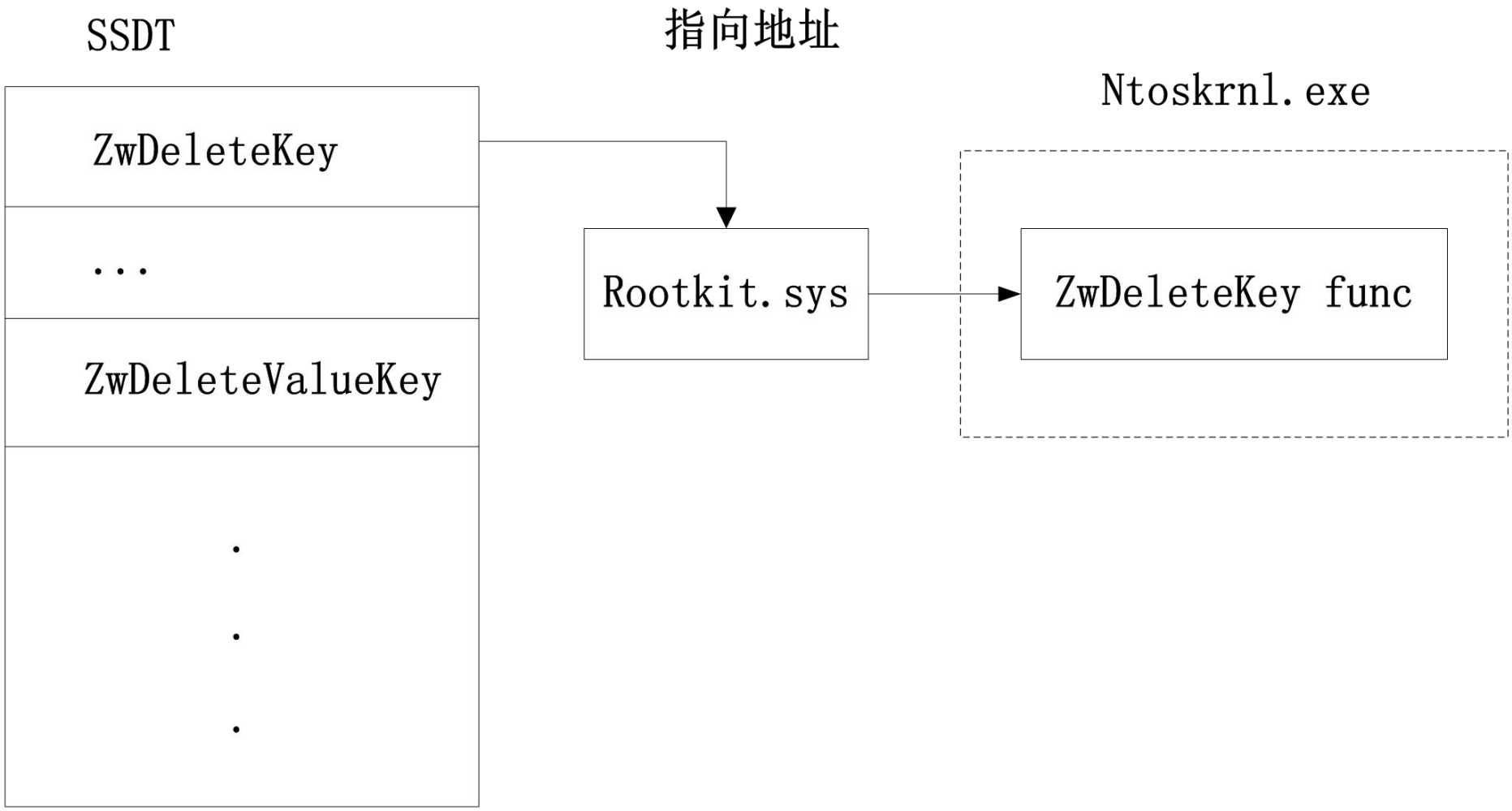
2.内核态Hook

- 内核态Hook种类
  - IDT钩子
  - SSDT钩子
  - 过滤驱动程序
  - 驱动程序钩子

1) IDT钩子

- IDT(Interrupt Descriptor Table,IDT)称为中断描述符表，其中指明了每个中断处理例程的地址。
- Rootkit通过修改这个表即可使发生中断调用时改变正常的执行路径。

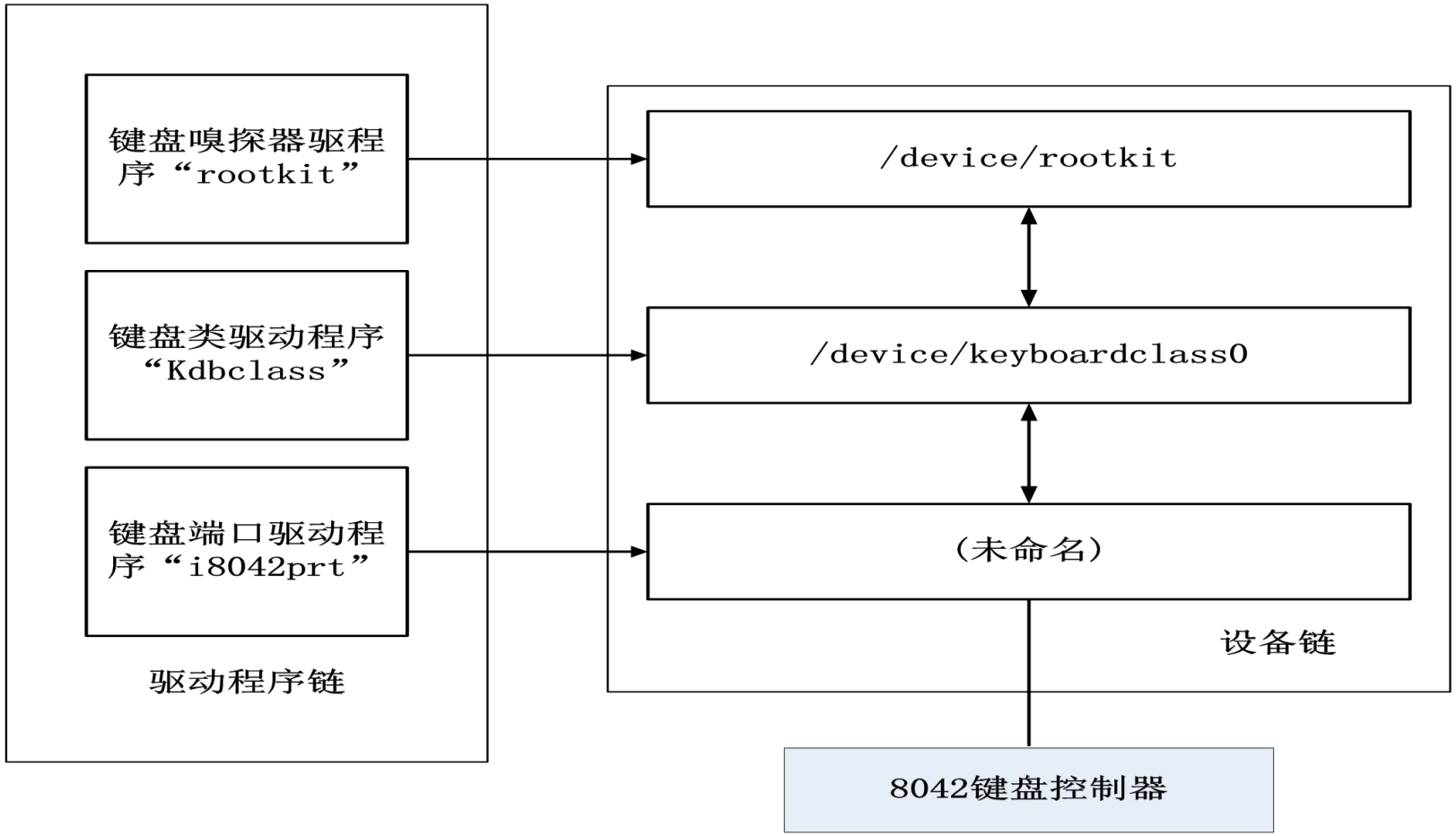
2) SSDT 钩子


SSDT
指向地址
Ntoskrnl.exe
ZwDeleteKey
...
ZwDeleteValueKey
Rootkit.sys
ZwDeleteKey func

3)过滤驱动程序

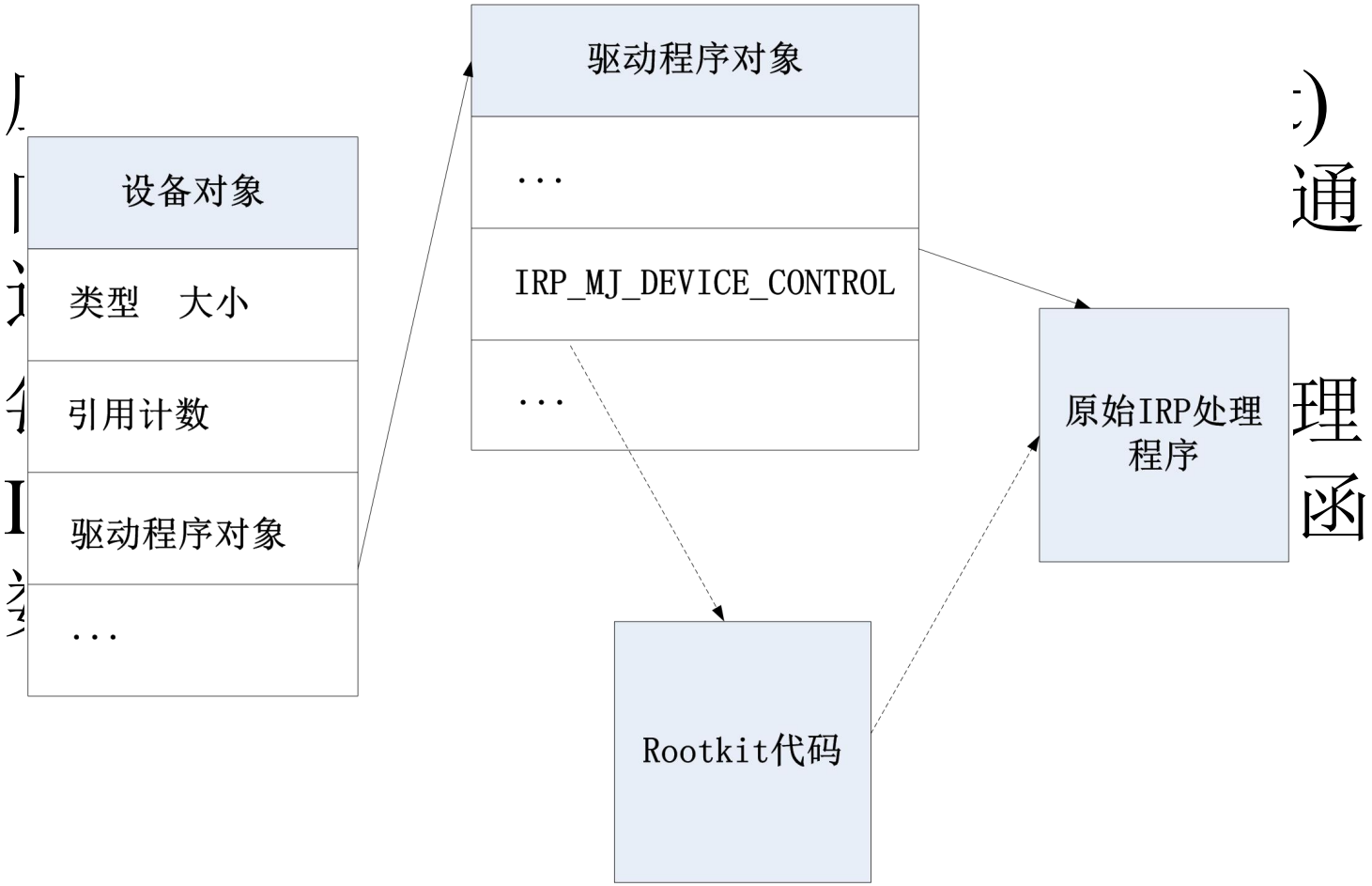
- Windows采用分层驱动程序的结构。
- 几乎所有的硬件都存在着驱动程序链。
  - –最低层的驱动程序处理对总线和硬件的直接访问。
  - –更高层驱动程序与应用程序打交道。

3)过滤驱动程序


键盘嗅探器驱程序“rootkit”
键盘类驱动程序“Kdbclass”
键盘端口驱动程序“i8042prt”
驱动程序链
/device/rootkit
/device/keyboardclass0
(未命名)
设备链
8042键盘控制器

4)驱动程序钩子

-
-


设备对象
类型　大小
引用计数
驱动程序对象
...
驱动程序对象
...
IRP_MJ_DEVICE_CONTROL
...
原始IRP处理程序
Rootkit代码

3.直接内核对象操作（DKOM）

- 内核对象都是由相应的数据结构表示的。
- 直接修改内核所使用的一些对象。
  - 用户空间进程请求操作系统信息。例如进程、线程或设备驱动程序列表。
  - 对象被报告给用户。
  - 修改对象。
- 优点：

  不必钩住API调用和过滤结果。

4.2.3 文件隐藏


用户态
FindFirstFileExw
API
NtQueryDirectoryFile
Native API
核心态文件处理
ZwQueryDirectoryFile
SSDT
IRP _MJ _DIRECTORY _CONTROL
文件过滤驱动
IRP _MJ _DIRECTORY _CONTROL
文件系统驱动
核心态磁盘处理
IRP _MJ _READ
磁盘过滤驱动
IRP _MJ _READ
磁盘驱动

文件隐藏

- 用户态文件隐藏

  IAT Hook FindFirstFile和FindNextFile
- 内核态文件隐藏

  SSDT Hook ZwQueryDirectoryFile

1).用户态文件隐藏

```
FindFirstFileA(
    __in LPCSTR lpFileName,
    __out LPWIN32_FIND_DATAA
     lpFindFileData );  //找到的文件或目录属性
FindNextFileA(
    __in HANDLE hFindFile,
    __out LPWIN32_FIND_DATAA
     lpFindFileData);
```

2).内核态文件隐藏

- 通过SSDT表钩挂ZwQueryDirectoryFile函数。

msdn

ZwQueryDirectoryFile Routine

The **ZwQueryDirectoryFile** routine returns various kinds of information about files in the directory specified by a given file handle.

Syntax

复制

```
NTSTATUS ZwQueryDirectoryFile(
  __in      HANDLE FileHandle,
  __in_opt  HANDLE Event,
  __in_opt  PIO_APC_ROUTINE ApcRoutine,
  __in_opt  PVOID ApcContext,
  __out     PIO_STATUS_BLOCK IoStatusBlock,
  __out     PVOID FileInformation,
  __in      ULONG Length,
  __in      FILE_INFORMATION_CLASS FileInformationClass,
  __in      BOOLEAN ReturnSingleEntry,
  __in_opt  PUNICODE_STRING FileName,
  __in      BOOLEAN RestartScan
);
```

FileInformationClass [in]

The type of information to be returned about files in the directory. One of the following.

Value	Meaning
FileBothDirectoryInformation	Return a **FILE_BOTH_DIR_INFORMATION** structure for each file.
FileDirectoryInformation	Return a **FILE_DIRECTORY_INFORMATION** structure for each file.
FileFullDirectoryInformation	Return a **FILE_FULL_DIR_INFORMATION** structure for each file.
FileIdBothDirectoryInformation	Return a **FILE_ID_BOTH_DIR_INFORMATION** structure for each file.
FileIdFullDirectoryInformation	Return a **FILE_ID_FULL_DIR_INFORMATION** structure for each file.
FileNamesInformation	Return a **FILE_NAMES_INFORMATION** structure for each file.
FileObjectIdInformation	Return a **FILE_OBJECTID_INFORMATION** structure for each file. This information class is valid only for NTFS volumes on Windows 2000 and later versions of Windows.
FileReparsePointInformation	Return a single **FILE_REPARSE_POINT_INFORMATION**

- ZwQueryDirctoryFile将为查询结果返回一个_FILE_BOTH_DIRCTORY_INFORMATIO的链表。
- 每个_FILE_BOTH_DIRCTORY_INFORMATION代表一个相应的文件。

FILE_ID_BOTH_DIR_INFORMATION Structure

The FILE_ID_BOTH_DIR_INFORMATION structure is used to query file reference number information for the files in a directory.

Syntax

复制

```
typedef struct _FILE_ID_BOTH_DIR_INFORMATION {
  ULONG          NextEntryOffset;
  ULONG          FileIndex;
  LARGE_INTEGER CreationTime;
  LARGE_INTEGER LastAccessTime;
  LARGE_INTEGER LastWriteTime;
  LARGE_INTEGER ChangeTime;
  LARGE_INTEGER EndOfFile;
  LARGE_INTEGER AllocationSize;
  ULONG          FileAttributes;
  ULONG          FileNameLength;
  ULONG          EaSize;
  CCHAR          ShortNameLength;
  WCHAR          ShortName[12];
  LARGE_INTEGER FileId;
  WCHAR          FileName[1];
} FILE_ID_BOTH_DIR_INFORMATION, *PFILE_ID_BOTH_DIR_INFORMATION;
```

Members

NextEntryOffset

Byte offset of the next FILE_ID_BOTH_DIR_INFORMATION entry, if multiple entries are present in a buffer. This member is zero if no other entries follow this one.

4.2.4 进程隐藏

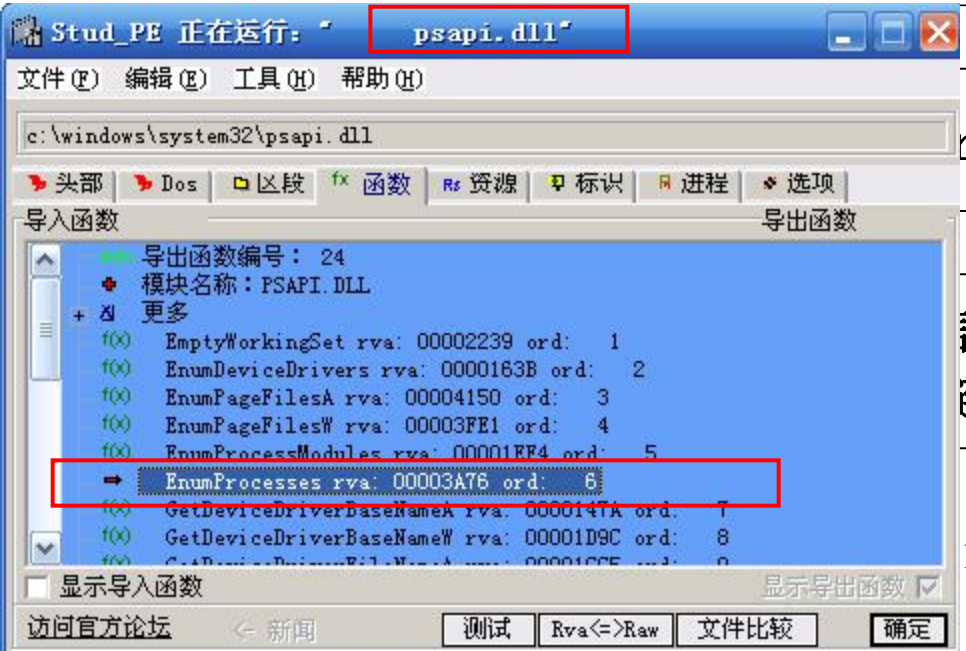
Psapi.EnumProcess
ToolHelp32/Process32First
ToolHelp32/Process32Next

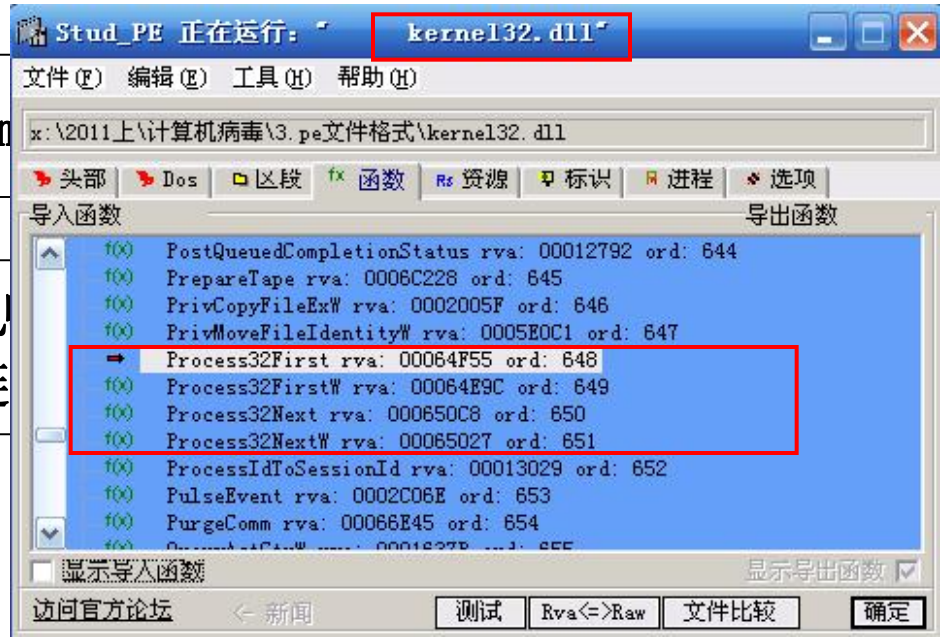
API

用户态

NtQuerySystemInformation

Native API





进程隐藏

- SSDT

  SSDT Hook ZwQuerySystemInformation
- DKOM

  EPROCESS结构断链

ZwQuerySystemInformation

- Windows操作系统通过ZwQuerySystemInformation函数查询进程信息

msdn

ZwQuerySystemInformation Function

[**ZwQuerySystemInformation** may be altered or unavailable in subsequent versions of Windows. Applications should use the alternate functions listed in this topic.]

Retrieves the specified system information.

Syntax

复制

```
NTSTATUS WINAPI ZwQuerySystemInformation(
  __in       SYSTEM_INFORMATION_CLASS SystemInformationClass,
  __inout    PVOID SystemInformation,
  __in       ULONG SystemInformationLength,
  __out_opt  PULONG ReturnLength
);
```

SystemInformationClass [in]

The type of system information to be retrieved. This parameter can be one of the following values from the **SYSTEM_INFORMATION_CLASS** enumeration type.

SystemBasicInformation

The number of processors in the system in a **SYSTEM_BASIC_INFORMATION** structure. Use the **GetSystemInfo** function instead.

SystemPerformanceInformation

An opaque **SYSTEM_PERFORMANCE_INFORMATION** structure that can be used to generate an unpredictable seed for a random number generator. Use the **CryptGenRandom** function instead.

SystemTimeOfDayInformation

An opaque **SYSTEM_TIMEOFDAY_INFORMATION** structure that can be used to generate an unpredictable seed for a random number generator. Use the **CryptGenRandom** function instead.

SystemProcessInformation

An array of **SYSTEM_PROCESS_INFORMATION** structures, one for each process running in the system.

These structures contain information about the resource usage of each process, including the number of handles used by the process, the peak page-file usage, and the number of memory pages that the process has allocated.

SystemProcessorPerformanceInformation

An array of **SYSTEM_PROCESSOR_PERFORMANCE_INFORMATION** structures, one for each processor installed in the system.

SystemInterruptInformation

An opaque **SYSTEM_INTERRUPT_INFORMATION** structure that can be used to generate an unpredictable seed for a random number generator. Use the **CryptGenRandom** function

SystemInformation

- 进程信息以链表的形式组织

SYSTEM_PROCESS_INFORMATION

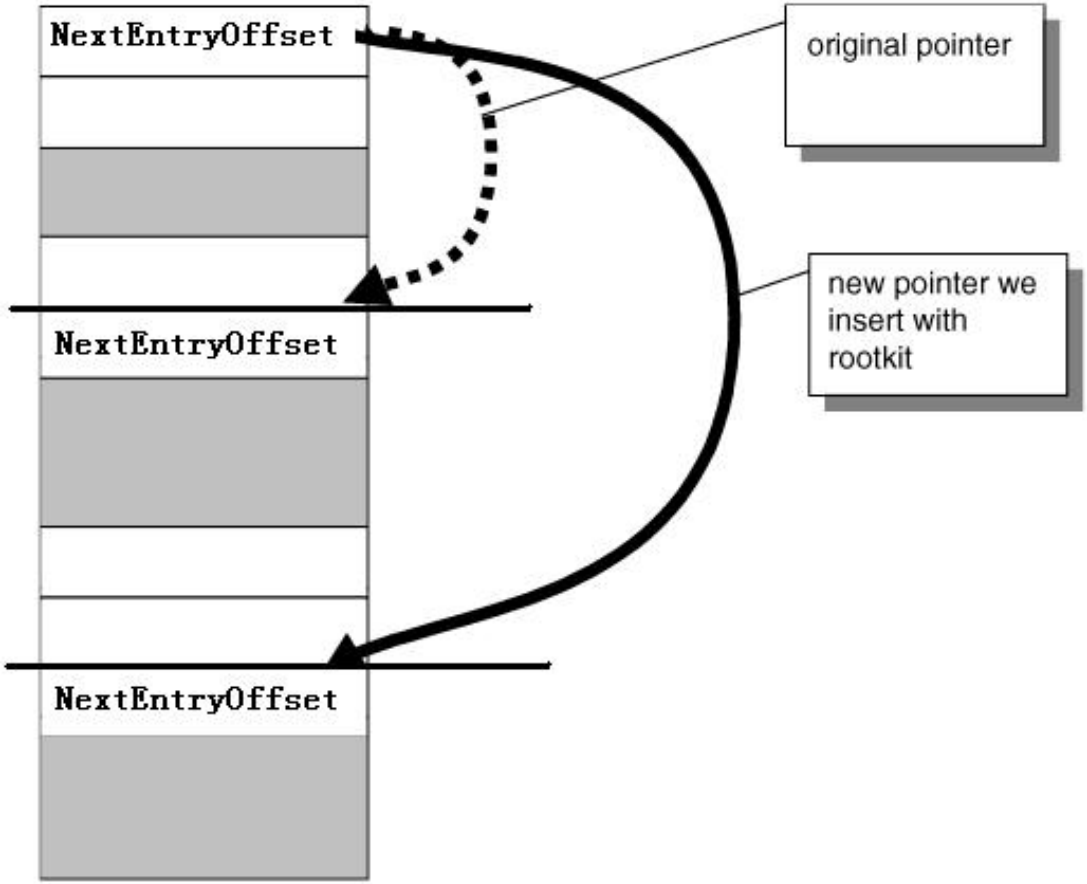
When the *SystemInformationClass* parameter is SystemProcessInformation, the buffer pointed to by the *SystemInformation* parameter should be large enough to hold an array that contains as many **SYSTEM_PROCESS_INFORMATION** structures as there are processes running in the system. Each structure has the following layout:

复制

```
typedef struct _SYSTEM_PROCESS_INFORMATION {
    ULONG NextEntryOffset;
    ULONG NumberOfThreads;
    BYTE Reserved1[48];
    PVOID Reserved2[3];
    HANDLE UniqueProcessId;
    PVOID Reserved3;
    ULONG HandleCount;
    BYTE Reserved4[4];
    PVOID Reserved5[11];
    SIZE_T PeakPagefileUsage;
    SIZE_T PrivatePageCount;
    LARGE_INTEGER Reserved6[6];
} SYSTEM_PROCESS_INFORMATION;
```

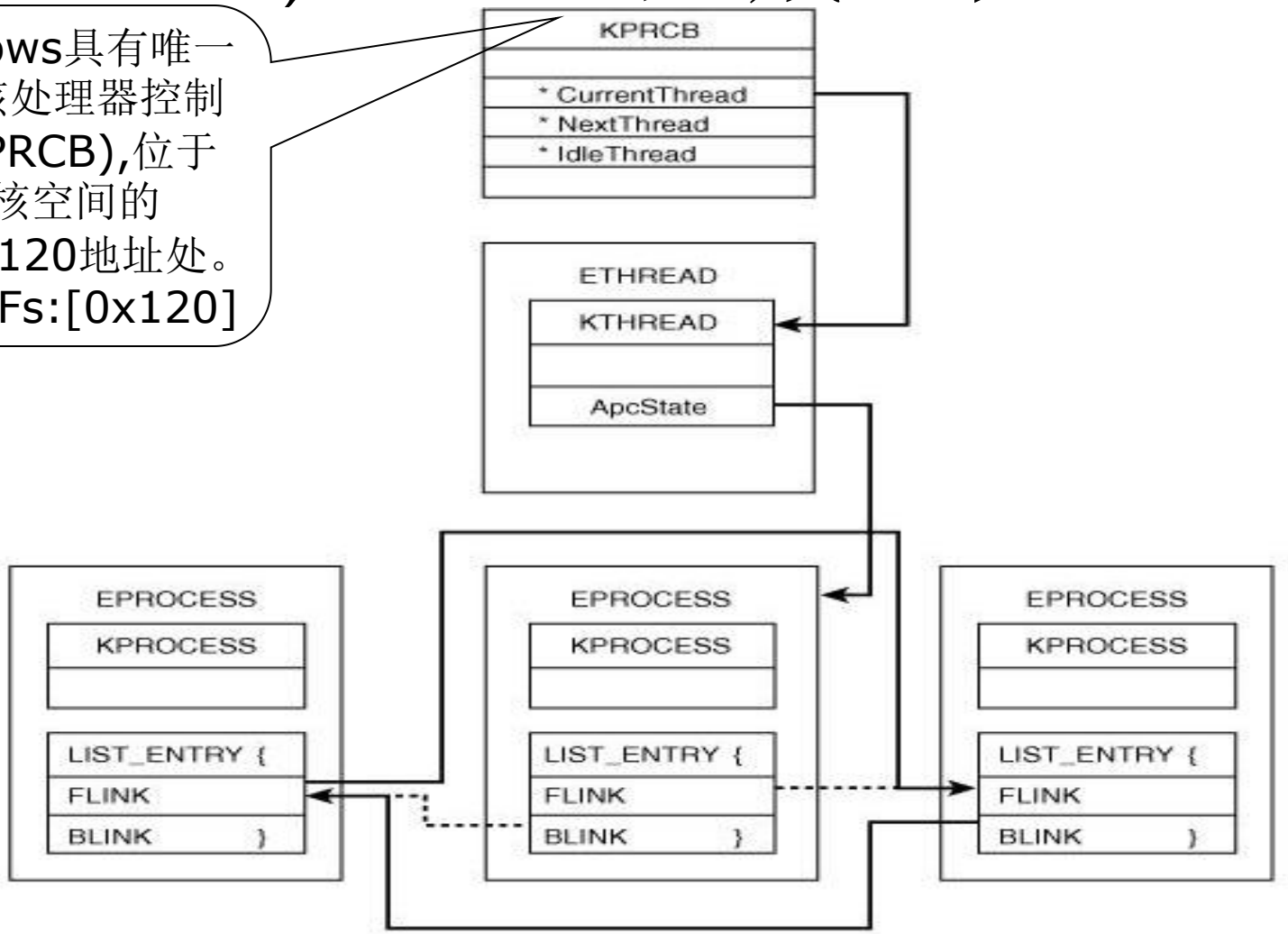
通过删除指定结构信息来隐藏进程

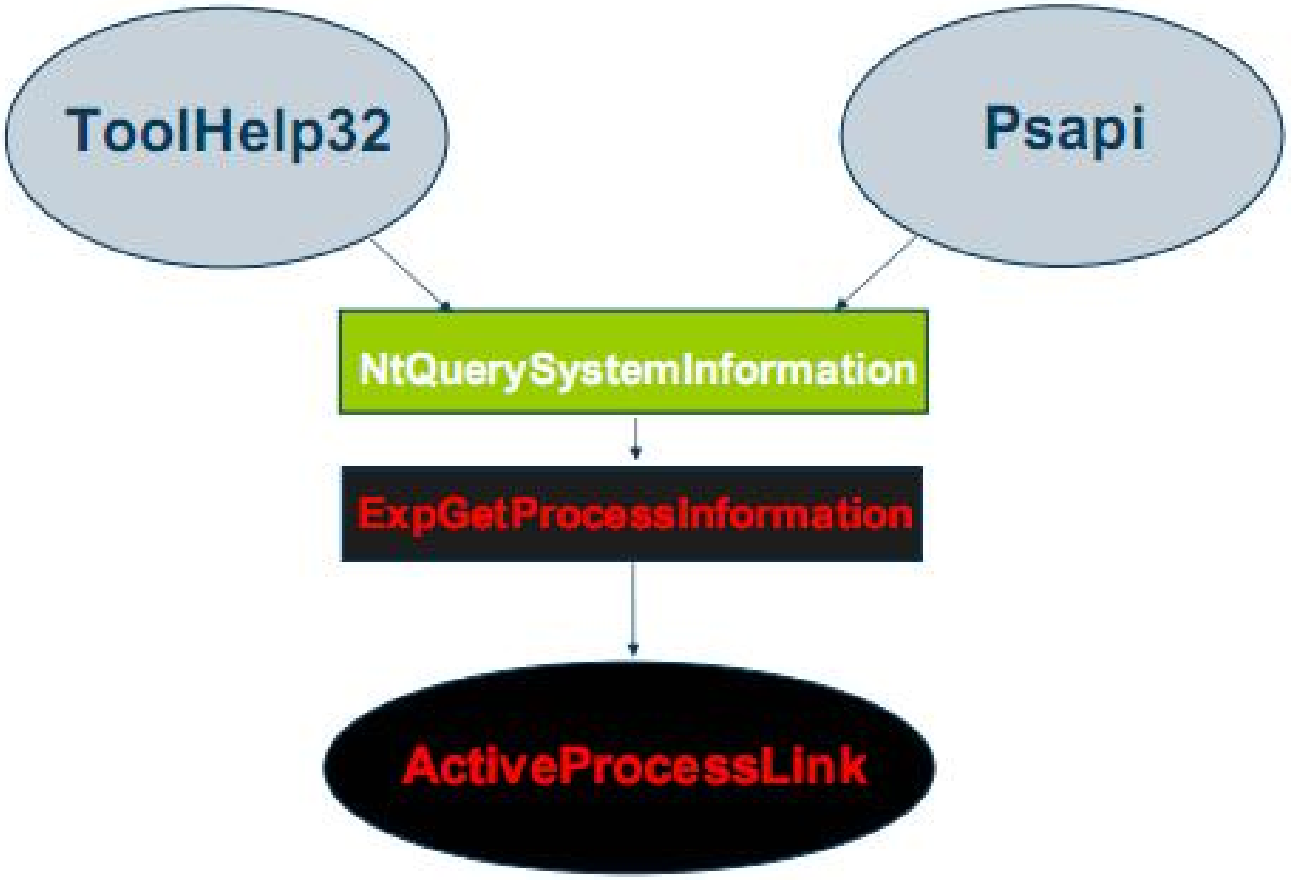
- 进程信息以链表的形式组织
  - 前驱结点
  - 后驱结点



2)DKOM隐藏进程

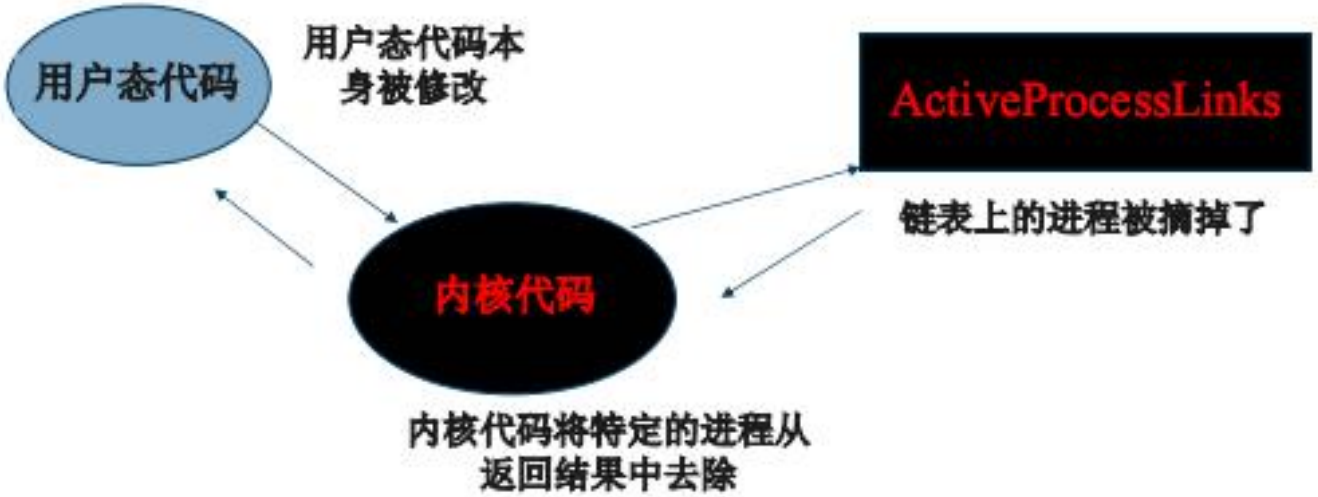
Windows具有唯一的内核处理器控制块(KPRCB),位于内核空间的0xffdff120地址处。
内核：Fs:[0x120]





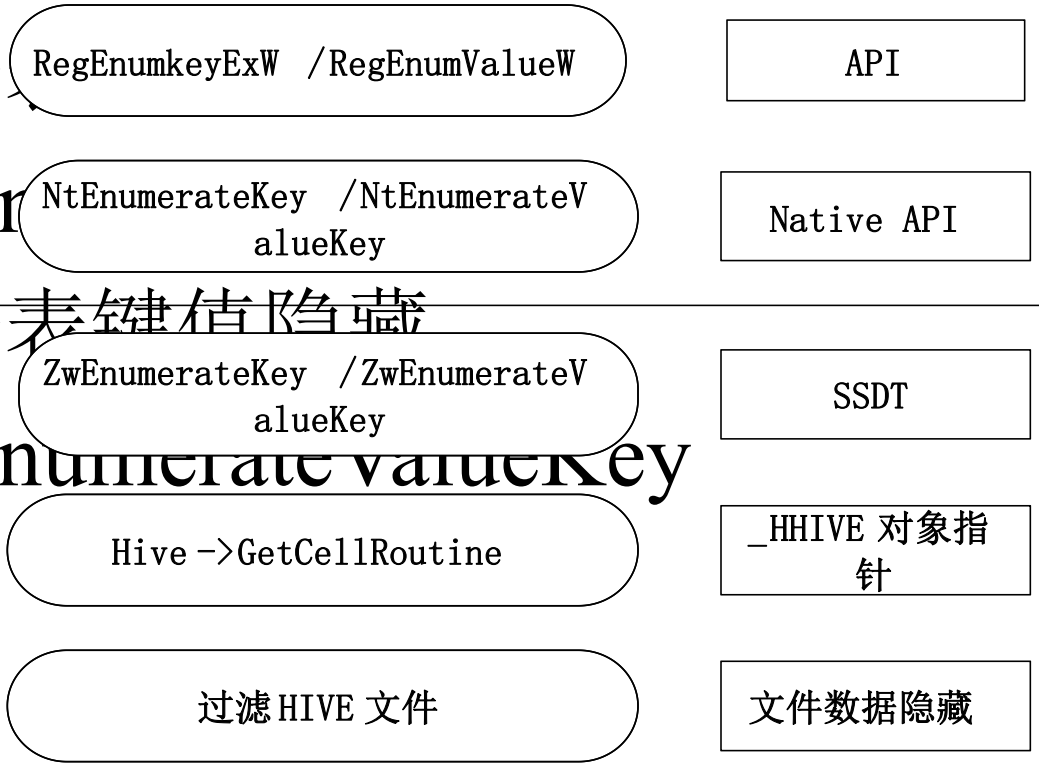
From Rootkit技术（Windows系统）， by 于旸 (yuyang@nsfocus.com)

- 控制整个过程的任意一个环节都可以隐藏进程
  - HOOK NtQuerySystemInformation()
    - 可以通过HOOK用户态的代码实现
  - 从ActiveProcessLinks摘除



4.2.5注册表隐藏

- 注册表键值隐藏
  - 用户态
    - ZwEnumerateValueKey
  - 核心态



ZwEnumerateKey Routine

The **ZwEnumerateKey** routine returns information about a subkey of an open registry key.

Syntax

复制

```
NTSTATUS ZwEnumerateKey(
  __in       HANDLE KeyHandle,
  __in       ULONG Index,
  __in       KEY_INFORMATION_CLASS KeyInformationClass,
  __out_opt  PVOID KeyInformation,
  __in       ULONG Length,
  __out      PULONG ResultLength
);
```

通过索引
NTSTAT
ZwEnum

KeyInforn
typedef
{
LAR
ULO
ULO
WCH
}KEY_I
Name和N

Parameters

KeyHandle [in]

Handle to the registry key that contains the subkeys to be enumerated. The handle is created by a successful call to **ZwCreateKey** or **ZwOpenKey**.

Index [in]

The index of the subkey that you want information for. If the key has *n* subkeys, the subkeys are numbered from 0 to *n*-1.

KeyInformationClass [in]

Specifies a **KEY_INFORMATION_CLASS** enumeration value that determines the type of information to be received by the *KeyInformation* buffer. Set *KeyInformationClass* to one of the following values:

- KeyBasicInformation
- KeyFullInformation
- KeyNodeInformation

If any value not in this list is specified, the routine returns error code

2)注册表键值隐藏

获取一个注册表键值信息的系统函数是**ZwEnumerateValueKey**。其函数原型是：

```
NTSTATUS
NtEnumerateValueKey(
    IN      HANDLE                      KeyHandle,
    IN      ULONG                       Index,
    IN      KEY_VALUE_INFORMATION_CLASS KeyValueInformationClass,
    OUT     PVOID                       KeyValueInformation,
    IN      ULONG                       KeyValueInformationLength,
    OUT     PULONG                      ResultLength );
```

KeyValueInformation缓冲区的结构是KEY_VALUE_BASIC_INFORMATION。其定义如下：

```
typedef struct _KEY_VALUE_BASIC_INFORMATION
{
        ULONG           TitelIndex;
        ULONG           Type;
        ULONG           NameLength;
        WCHAR           Name[1];
}KEY_VALUE_BASIC_INFORMATION;
```

Name和NameLength分别表示键值名和键值长度。

注册表隐藏

键名	索引	实际索引
A	0	0
B	1	1
C	2	2
D		3
E	3	4
F	4	5

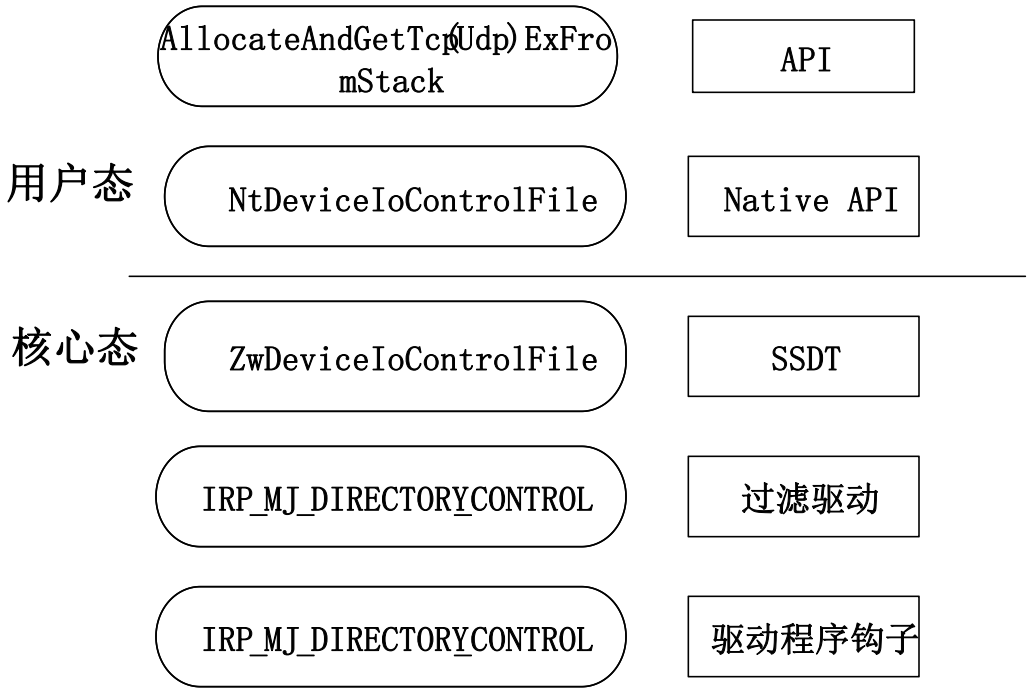
要隐藏的目标注册表项

```
NTSTATUS
    NewNtEnumerateKey(
    IN    ULONG     Index,
    OUT   PVOI D    KeyValueInformation,);
```

- ◆变量newIndex
- ◆注册表信息过滤算法：
  - 查询序号为Index的键时，返回Index+n键值的信息，其中n为Index内要过滤的值的数量。
  - Index以内的键中有几个过滤项，newIndex比Index就大几。
- ◆ 调用NtEnumerateKey(**NewIndex**,…)

4.2.6 端口隐藏

系统中可以进行端口隐藏的位置的概括图



端口隐藏

- SSDT

  ZwDeviceIoControlFile
- 驱动程序

  IRP_MJ_DIRECTORY_CONTROL

1)SSDT钩挂隐藏端口

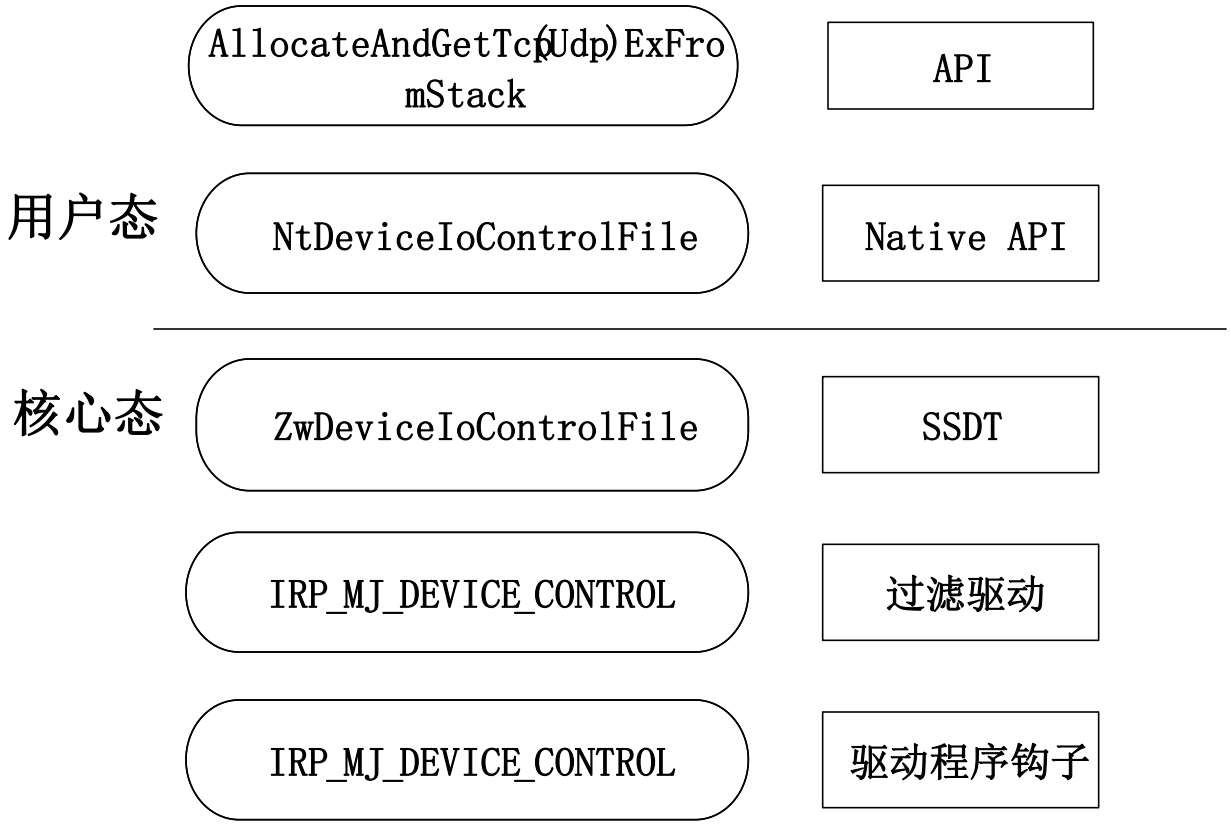
- 查询端口的API是调用ZwDeviceIoControlFile来获得系统中所有打开端口的列表。
- 进行TCP端口查询时，在其输入参数InputBuffer中指定。
- 返回的端口列表信息保存在该API参数的OutBuffer中。

1)SSDT钩挂隐藏端口

- 查询TCP端口信息时，返回结构如下：

```
typedef struct TCPAddrEntry
{
        ULONG           tae_ConnState;
        ULONG           tae_ConnLocalAddress;
        ULONG           tae_ConnLocalPort;
        ULONG           tae_ConnRemAddress;
        ULONG           tae_ConnRemPort;
}TCPAddrEntry;
```

2)驱动程序钩子隐藏端口



2)驱动程序钩子隐藏端口

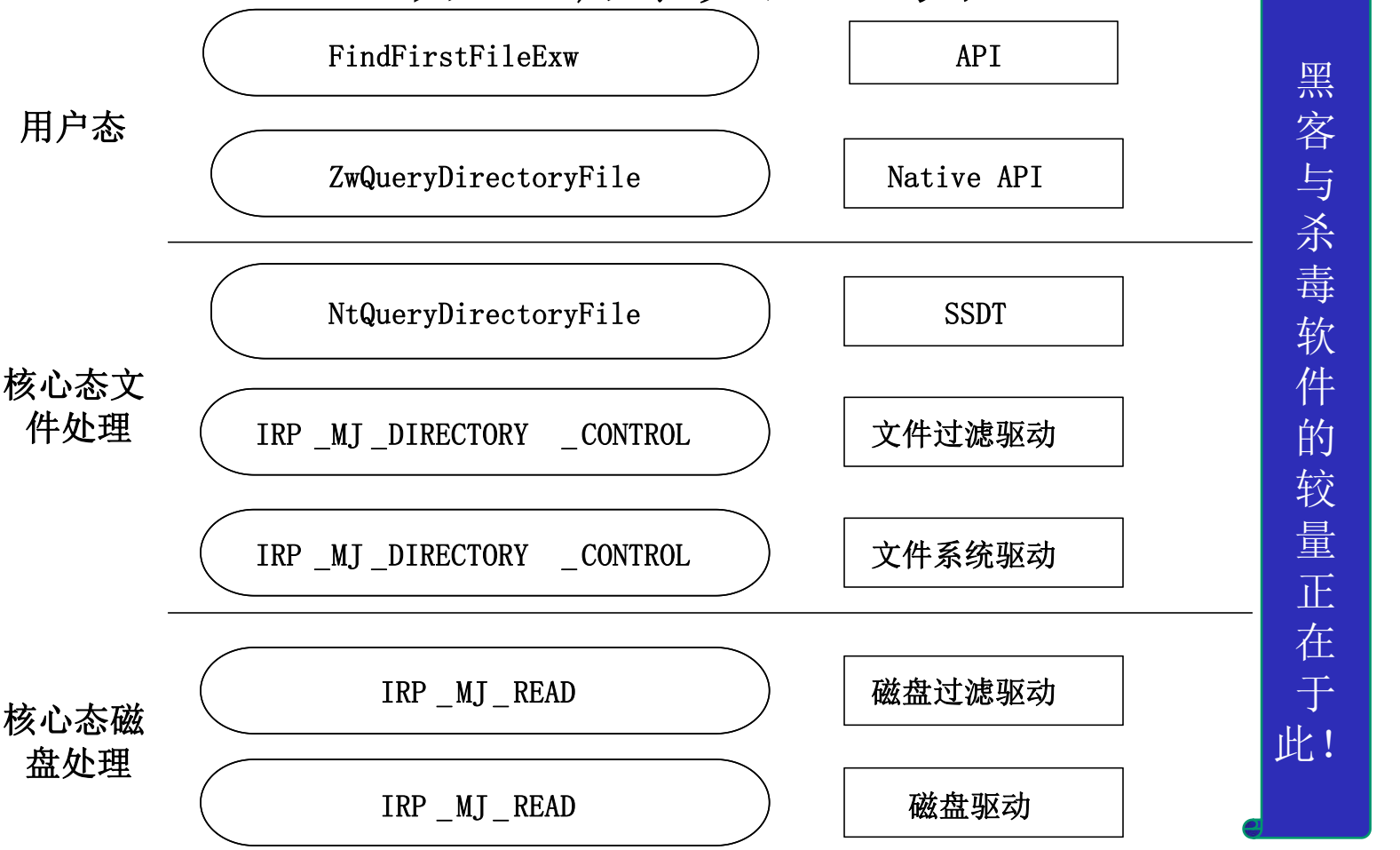
- 获取TCPIP.SYS指针
- 获取驱动派遣函数地址表(IRP_MJ_DEVICE_CONTROL)
- 钩挂该表项地址的值，改为钩子函数的地址
- 在钩子函数中获取返回的内容，并进行处理

2)驱动程序钩子隐藏端口

- TCPIP.SYS返回包含CONNINFO102结构的缓冲区

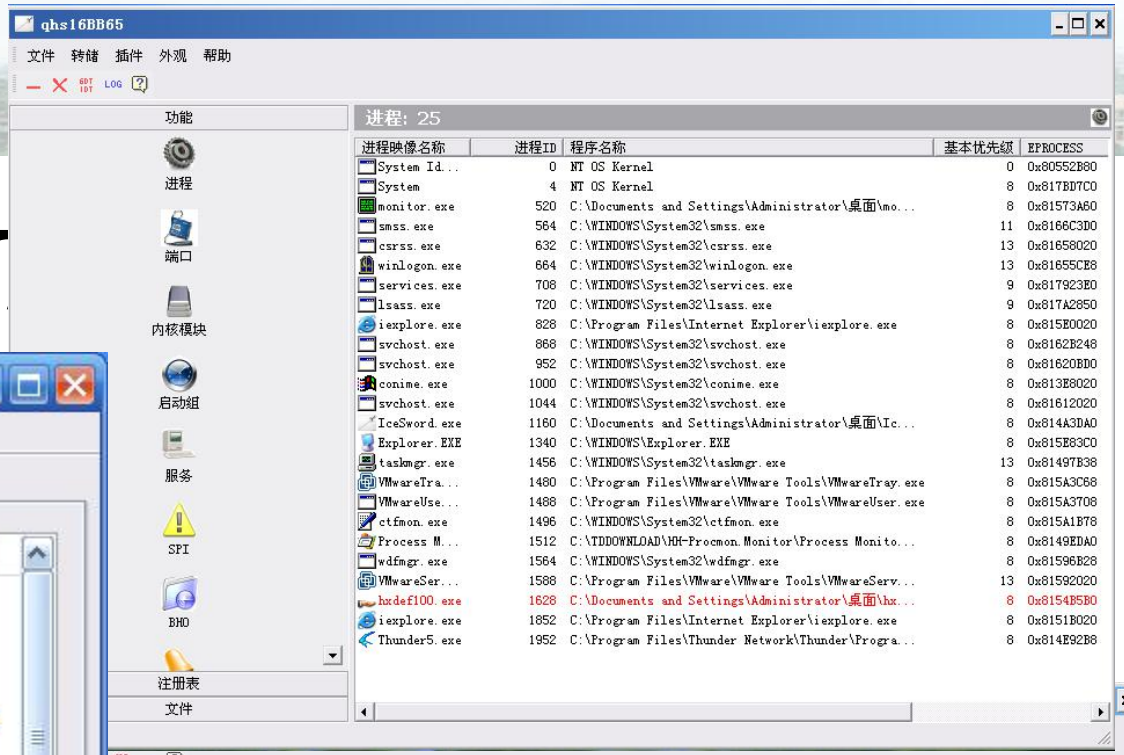
```
typedef struct _CONNINFO102 {
   unsigned long status;
   unsigned long src_addr;
   unsigned short src_port;
   unsigned short unk1;
   unsigned long dst_addr;
   unsigned short dst_port;
   unsigned short unk2;
   unsigned long pid;
} CONNINFO102, *PCONNINFO102;
```

总结（层次+钩挂）


用户态
FindFirstFileExw
API
ZwQueryDirectoryFile
Native API
核心态文件处理
NtQueryDirectoryFile
SSDT
IRP _MJ _DIRECTORY _CONTROL
文件过滤驱动
IRP _MJ _DIRECTORY _CONTROL
文件系统驱动
核心态磁盘处理
IRP _MJ _READ
磁盘过滤驱动
IRP _MJ _READ
磁盘驱动
黑客与杀毒软件的较量正在于此！

4.2.1


Windows 任务管理器
文件(F) 选项(O) 查看(V) 关机(U) 帮助(H)
应用程序 进程 性能 联网 用户
映像名称 用户名 CPU 内存使用
conime.exe 00 3,224 K
csrss.exe 00 9,684 K
ctfmon.exe 00 3,392 K
Explorer.EXE 00 15,812 K
IceSword.exe 00 944 K
iexplore.exe 00 5,820 K
iexplore.exe 00 6,648 K
lsass.exe 00 844 K
monitor.exe 00 1,348 K
Process Monit... 00 6,552 K
services.exe 00 4,260 K
smss.exe 00 464 K
svchost.exe 00 3,552 K
svchost.exe 00 4,300 K
svchost.exe 00 15,776 K
System 00 296 K
System Idle P... SYSTEM 97 28 K
taskmgr.exe 03 2,668 K
显示所有用户的进程(S)
结束进程(E)
进程数: 24
CPU 使用: 7%
提交更改: 463200K / 631940


qhs16BB65
文件 转储 插件 外观 帮助
功能
进程
端口
内核模块
启动组
服务
SPI
BHO
注册表
文件
进程: 25
进程映像名称 进程ID 程序名称 基本优先级 EPROCESS
System Id... 0 NT OS Kernel 0 0x80552B80
System 4 NT OS Kernel 8 0x817BD7C0
monitor.exe 520 C:\Documents and Settings\Administrator\桌面\mo... 8 0x81573A60
smss.exe 564 C:\WINDOWS\System32\smss.exe 11 0x8166C3D0
csrss.exe 632 C:\WINDOWS\System32\csrss.exe 13 0x81658020
winlogon.exe 664 C:\WINDOWS\System32\winlogon.exe 13 0x81655CE8
services.exe 708 C:\WINDOWS\System32\services.exe 9 0x817923E0
lsass.exe 720 C:\WINDOWS\System32\lsass.exe 9 0x817A2850
iexplore.exe 828 C:\Program Files\Internet Explorer\iexplore.exe 8 0x815E0020
svchost.exe 868 C:\WINDOWS\System32\svchost.exe 8 0x8162B248
svchost.exe 952 C:\WINDOWS\System32\svchost.exe 8 0x81620BD0
conime.exe 1000 C:\WINDOWS\System32\conime.exe 8 0x813E8020
svchost.exe 1044 C:\WINDOWS\System32\svchost.exe 8 0x81612020
IceSword.exe 1160 C:\Documents and Settings\Administrator\桌面\Ic... 8 0x814A3DA0
Explorer.EXE 1340 C:\WINDOWS\Explorer.EXE 8 0x815E83C0
taskmgr.exe 1456 C:\WINDOWS\System32\taskmgr.exe 13 0x81497B38
VMwareTra... 1480 C:\Program Files\VMware\VMware Tools\VMwareTray.exe 8 0x815A3C68
VMwareUse... 1488 C:\Program Files\VMware\VMware Tools\VMwareUser.exe 8 0x815A3708
ctfmon.exe 1496 C:\WINDOWS\System32\ctfmon.exe 8 0x815A1B78
Process M... 1512 C:\TDDOWNLOAD\HH-Procmon.Monitor\Process Monito... 8 0x8149EDA0
wdfmgr.exe 1564 C:\WINDOWS\System32\wdfmgr.exe 8 0x81596B28
VMwareSer... 1588 C:\Program Files\VMware\VMware Tools\VMwareServ... 13 0x81592020
hxdef100.exe 1628 C:\Documents and Settings\Administrator\桌面\hx... 8 0x8154B5B0
iexplore.exe 1852 C:\Program Files\Internet Explorer\iexplore.exe 8 0x8151B020
Thunder5.exe 1952 C:\Program Files\Thunder Network\Thunder\Progra... 8 0x814E92B8
功能
注册表
文件
Computer
可移动磁盘(A:)
本地磁盘(C:)
Config.Msi
Documents and Settings
Administrator
「开始」菜单
Application Data
Cookies
Favorites
Local Settings
My Documents
NetHood
PrintHood
Recent
SendTo
Templates
桌面
hxdef100r
driver
units
All Users
「开始」菜单
Application Data
Documents
DRM
Favorites
Templates
桌面
IceSword
名称 大小 占用空间 创建时间 修改时间
rdrbs100.ini 63 4096 2009-08-28 15:51:44 2009-08-
bdcli100.dpr 10361 12288 2009-08-28 15:50:15 2009-08-
bdcli100.exe 28624 28672 2009-08-28 15:50:15 2009-08-
driver.res 3408 4096 2009-08-28 15:50:15 2009-08-
hxdef-OFdis.exe 70656 73728 2009-08-28 15:50:15 2009-08-
hxdef100.2.ini 3924 4096 2009-08-28 15:50:15 2009-08-
hxdef100.dpr 353366 356352 2009-08-28 15:50:15 2009-08-
hxdef100.exe 70656 73728 2009-08-28 15:50:15 2009-08-
hxdef100.ini 4119 8192 2009-08-28 15:50:15 2009-08-
rdrbs100.dpr 57931 61440 2009-08-28 15:50:15 2009-08-
rdrbs100.exe 49152 49152 2009-08-28 15:50:15 2009-08-
rdrbs100.res 1220 4096 2009-08-28 15:50:15 2009-08-
readmecz.txt 37407 40960 2009-08-28 15:50:15 2009-08-
readmeen.txt 37905 40960 2009-08-28 15:50:15 2009-08-
src.zip 93679 94208 2009-08-28 15:50:15 2009-08-
hxdefdrv.sys 3342 4096 2009-08-28 16:01:30 2009-08-

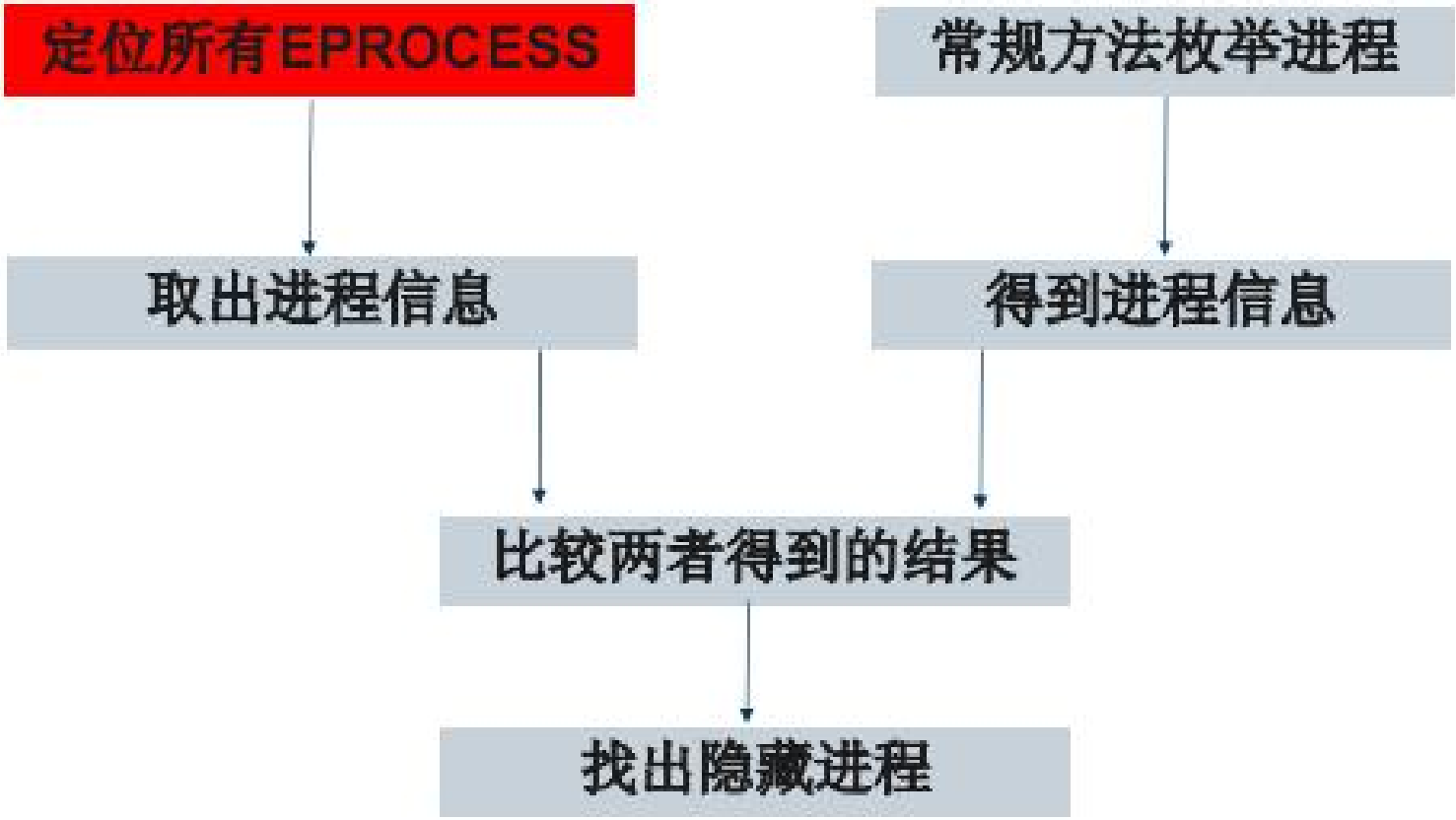
Hackdef hook的API函数

- File：
  - Kernel32.ReadFile
  - Ntdll.NtVdmControl
  - Ntdll.NtQueryDirectoryFile
  - Ntdll.NtCreateFile
  - Ntdll.NtQueryVolumeInformationFile
- Process：
  - Ntdll.NtQuerySystemInformation (class 5 & 16)
  - Ntdll.NtOpenProcess
  - Ntdll.NtResumeThread
- Regedit：
  - Ntdll.NtEnumerateKey
  - Ntdll.NtEnumerateValueKey
- NetWork：
  - WS2_32.recv
  - WS2_32.WSARecv
  - Ntdll.NtDeviceIoControlFile
- Service：
  - Advapi32.EnumServiceGroupW
  - Advapi32.EnumServicesStatusExW
  - Advapi32.EnumServicesStatusExA
  - Advapi32.EnumServicesStatusA
- Ntdll.NtLdrLoadDll
- Ntdll.NtLdrInitializeThunk
- Ntdll.NtReadVirtualMemory

4.2.8 Rootkit检测

- 比较两种检测结果
  - 采用系统默认的方式访问被检测对象
  - 采用更底层或特殊的方式访问被检测对象
    - 检测效果的关键所在

Windows的RootKit——检测隐藏的进程

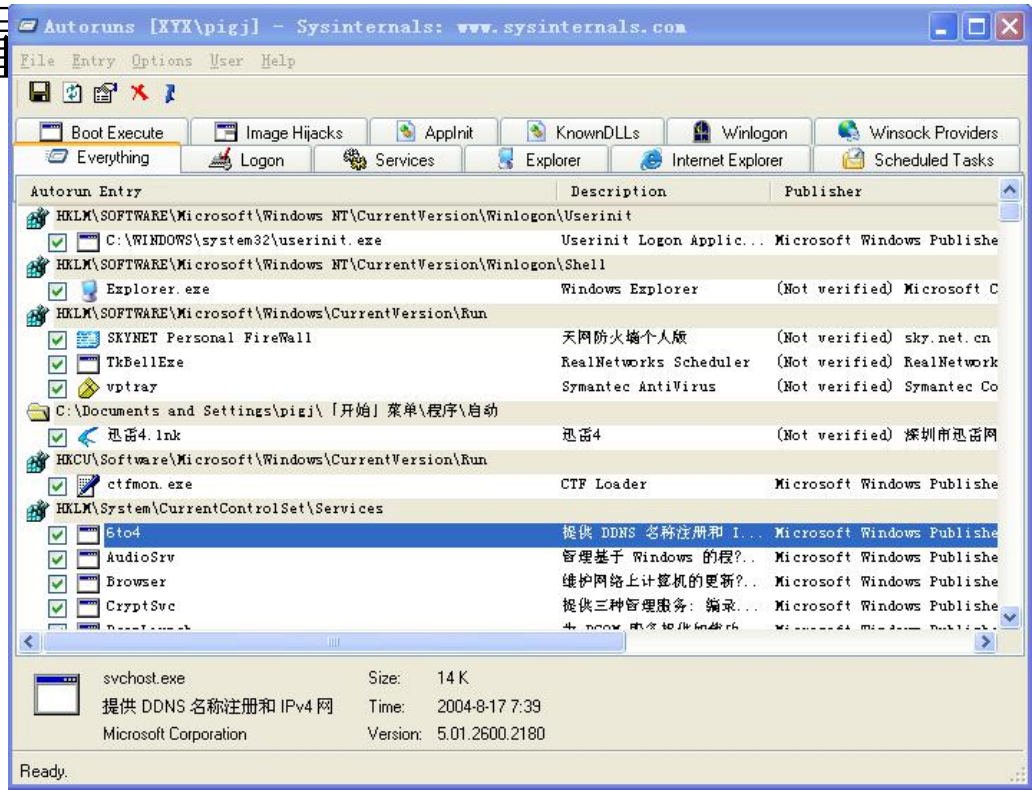


相关检测工具

- 部分Rootkit检测工具
  - IceSword
  - Darkspy
  - GMER
  - BlackLight
  - RootkitRevealer
  - Rootkit Unhooker
  - SanityCheck
  - XueTr
  - …

常见的自启动方式

- 注册表中的键值
- 系统中的特定位置
- 配置文件
- 修改特定的文件
  - 如Explorer.exe



作业

- 对Windows下的所有启动方式进行总结，并针对每一个启动方式进行启动演示。
- 结合木马和Rootkit的功能和特性，设计一种可以检测或者防御该恶意代码的工具。

- Question?