网络攻防技术与实践课程

课程9.恶意代码基础知识和分析方法

诸葛建伟

zhugejw@gmail.com

3Q战争

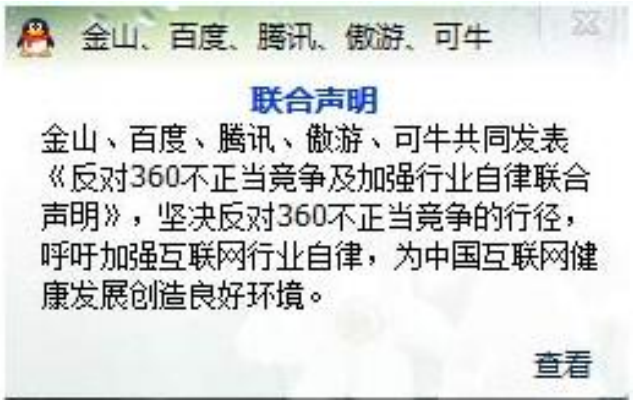
- 3Q战争: Qihoo 360 VS 腾讯QQ
  - 中国Internet上的一场”日俄战争”
- 事件回顾
  - 9.27导火索: 奇虎发布 “360隐私保护器”
    - 宣称QQ窥视用户隐私，剑指腾讯QQ
  - 9月下旬-10月下旬口水战
    - 10.14腾讯起诉奇虎，奇虎称将反诉
    - 10.27腾讯、百度、金山、傲游、可牛发表《反对360不正当竞争联合声明》
    - 10.28弹窗战争
  - 10月29日-11月上旬：软件攻防之战
    - 10.29奇虎推“扣扣保镖”死磕QQ
    - 11.3腾讯QQ“有他没我”不共戴机
    - 11.5金山、傲游、可牛、百度、搜狗五厂商宣布将不兼容360
  - 11月上/中旬：政府监管部门介入，暂时消停

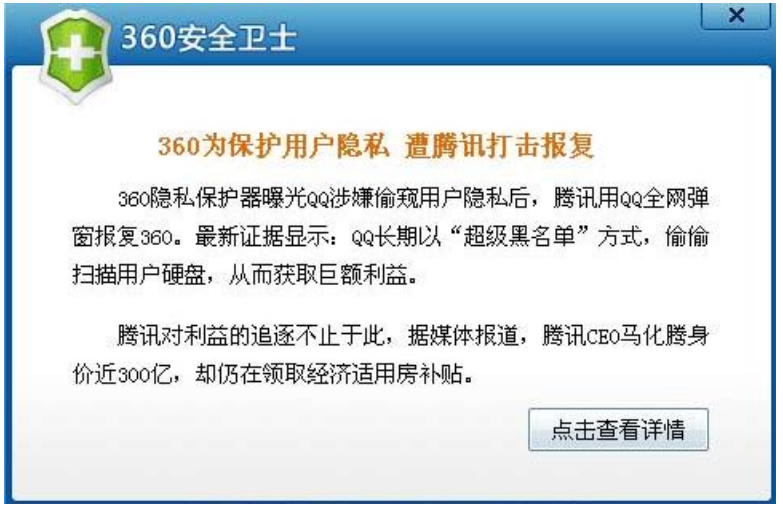
导火索 - “360隐私保护器”

- 仅针对QQ
  - 查看文件访问列表
  - 没有任何保护功能
- QQ回应
  - QQ账户安全模块对盗号木马的扫描:未提示
- 技术实现
  - DLL注入 & API Hooking
  - 记录CreateFile函数(文件访问与操作行为)调用
  - QQ.exe → 腾讯QQ

口水战 - 弹窗战争

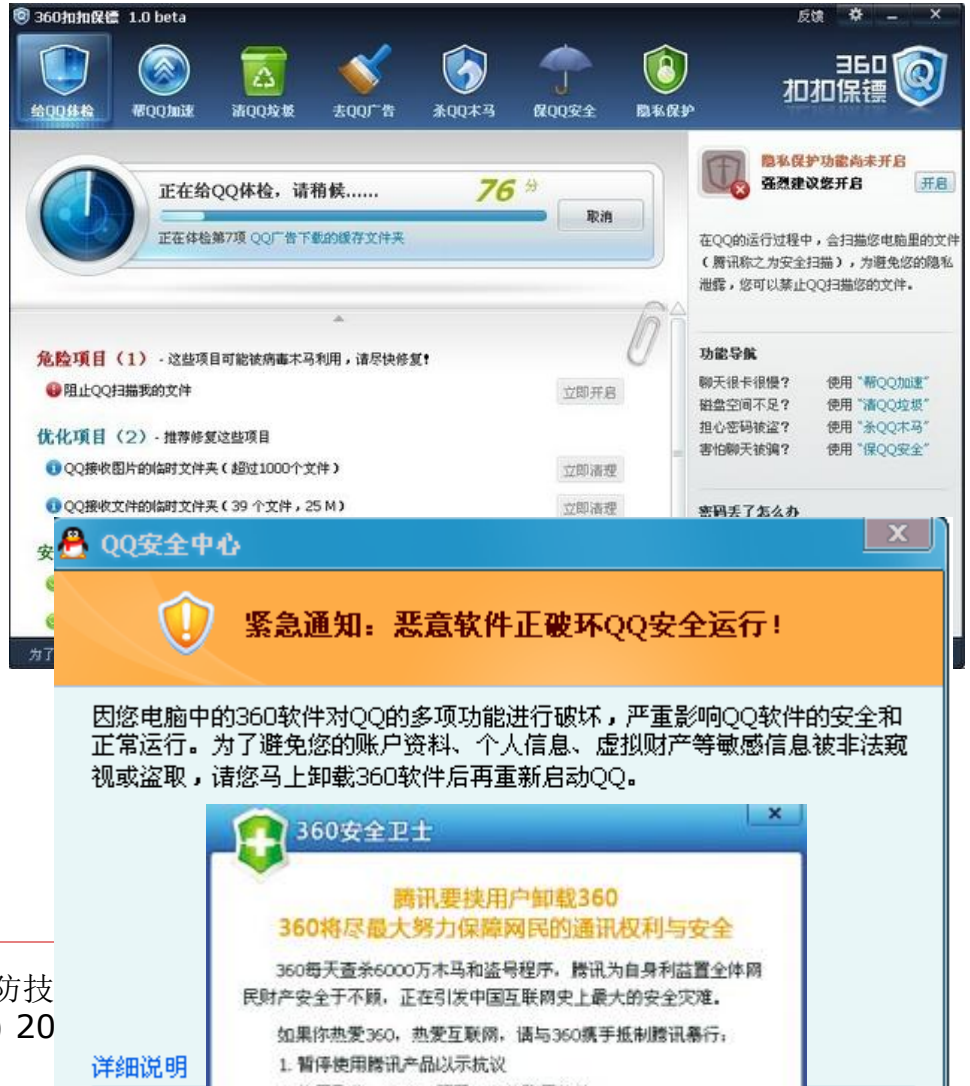
- 腾讯QQ全网弹窗
  - 五家联合声明
  - “行业正义战”
- 360弹窗应对
  - “迫不得已”
  - “以前只有在跟瑞星打仗的时候，动用过弹窗，但那时360用户不多，所以影响不大”
  - “这是一个很痛苦的决定”





软件攻防之战

- “扣扣保镖”
  - 用户提示?用户误导?–道德思考
  - QQ外挂?-法律问题
  - 捆绑360安全卫士强制推广
- QQ“不共戴机”
  - 在装有360软件的电脑上停止运行QQ软件
  - 被指“以退为进”，“绑架网友”
- 360召回“扣扣保镖”



当“小偷”碰上“流氓”

- **360历来有口水战和搅局传统**
  - 周鸿祎**3721**，雅虎助手—”中国流氓软件之父”
  - **360**安全卫士：“网民救世主”—”查杀流氓软件”, 反木马，“免费杀毒”
  - 搅局反病毒业界，口水战/”误报”门/“漏洞门”：**360 VS.** 金山、卡巴、瑞星
- **腾讯历来有“拿来主义”、“剽窃Idea”的传统**
  - **QQ(oicq) VS. ICQ, QQ**游戏 **VS.** 联众游戏, **QQ**农场 **VS.** 开心农场, **QQ**超级旋风 **VS.** 迅雷…
  - 充分利用**QQ**庞大的用户基础，将原先的创新者打趴下
  - **QQ**医生/**QQ**电脑管家 **VS. 360**安全卫士/软件管理: 与奇虎构成直接竞争关系

3Q事件的"结局"

- 天朝工部、刑部介入监管
  - 约谈双方负责人
  - "各打五十大板"：通报批评，向社会公开道歉
  - 两家公司采取不正当竞争行为，造成了恶劣的社会影响
  - 涉嫌违反相关法律规定的行为进行进一步调查处理
- 腾讯、360分别道歉
  - 腾讯《和你在一起》-"忽略了用户的感受"/"阳光下的竞争"
  - 360《再次致歉社会与网民的道歉信》-"用户利益至上"/"专业品质、免费安全"
- "结局"？
  - 腾讯起诉360不正当竞争案件，将于明日在北京开庭一审

- 卸载QQ/360，此事与我从此无关

内容

1. 恶意代码基础知识
2. 恶意代码分析技术
   - 课题实践：恶意代码静态分析
   - 课堂实践：分析Crackme程序
3. 课外作业：分析一个自制恶意代码样本

恶意代码(Malware)

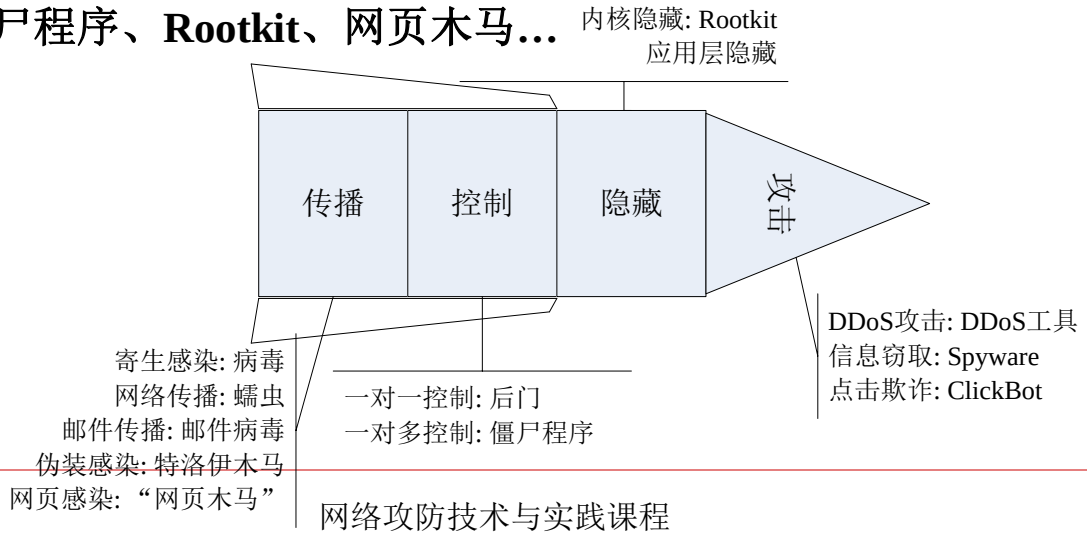
- 恶意代码定义
  - ***Malware is a set of instructions that run on your computer and make your system do something that an attacker wants it to do.***
  - 使计算机按照攻击者的意图运行以达到恶意目的的指令集合。
  - 指令集合: 二进制执行文件, 脚本语言代码, 宏代码, 寄生在文件、启动扇区的指令流
  - 恶意代码目的: 技术炫耀/恶作剧, 远程控制, 窃取私密信息, 盗用资源, 拒绝服务/破坏, ...
- 恶意代码类型
  - 计算机病毒, 蠕虫, 恶意移动代码, 后门, 特洛伊木马, 僵尸程序, **Rootkit**等...
  - 计算机病毒是最早出现的恶意代码，媒体/工业界的概念混淆，经常以计算机病毒**(Computer Virus)**等价于恶意代码

恶意代码的类型

恶意代码类型	定义特征	典型实例
计算机病毒（**Virus**）	通过感染文件**(**可执行文件、数据文件、电子邮件等**)**或磁盘引导扇区进行传播，一般需要宿主程序被执行或人为交互才能运行	**Brain, Concept, CIH**
蠕虫（**Worm**）	一般为不需要宿主的单独文件，通过网络传播，自动复制，通常无需人为交互便可感染传播	**Morris, Code Red, Slammer**
恶意移动代码（**Malicious mobile code**）	从远程主机下载到本地执行的轻量级恶意代码，不需要或仅需要极少的人为干预。代表性的开发工具有：**JavaScript**，**VBScript**，**Java**，以及**ActiveX**	**Santy Worm**
后门（**Backdoor**）	绕过正常的安全控制机制，从而为攻击者提供访问途径	**Netcat, BO,** 冰河
特洛伊木马 **(Trojan)**	伪装成有用软件，隐藏其恶意目标，欺骗用户安装执行	**Setiri**
僵尸程序 **(Bot)**	使用一对多的命令与控制机制组成僵尸网络	**Sdbot, Agobot**
内核套件**(Rootkit)**	通过替换或修改系统关键可执行文件（用户态），或者通过控制操作系统内核**(**内核态**)**，用以获取并保持最高控制权**(root access)**	**LRK, FU, hdef**
融合型恶意代码	融合上述多种恶意代码技术，构成更具破坏性的恶意代码形态	**Nimda**

恶意代码的命名规则与分类体系

- 恶意代码命名规则
  - [恶意代码类型.]恶意代码家族名称[.变种号]
- 恶意代码分类的混淆
  - 反病毒工业界并没有形成规范的定义，概念混淆
  - 各种恶意代码形态趋于融合
- 各种形态恶意代码在关键环节上具有其明确的定义特性
  - 传播、控制、隐藏、攻击
  - 针对明确定义特性对恶意代码进行分类研究
    - 僵尸程序、Rootkit、网页木马…



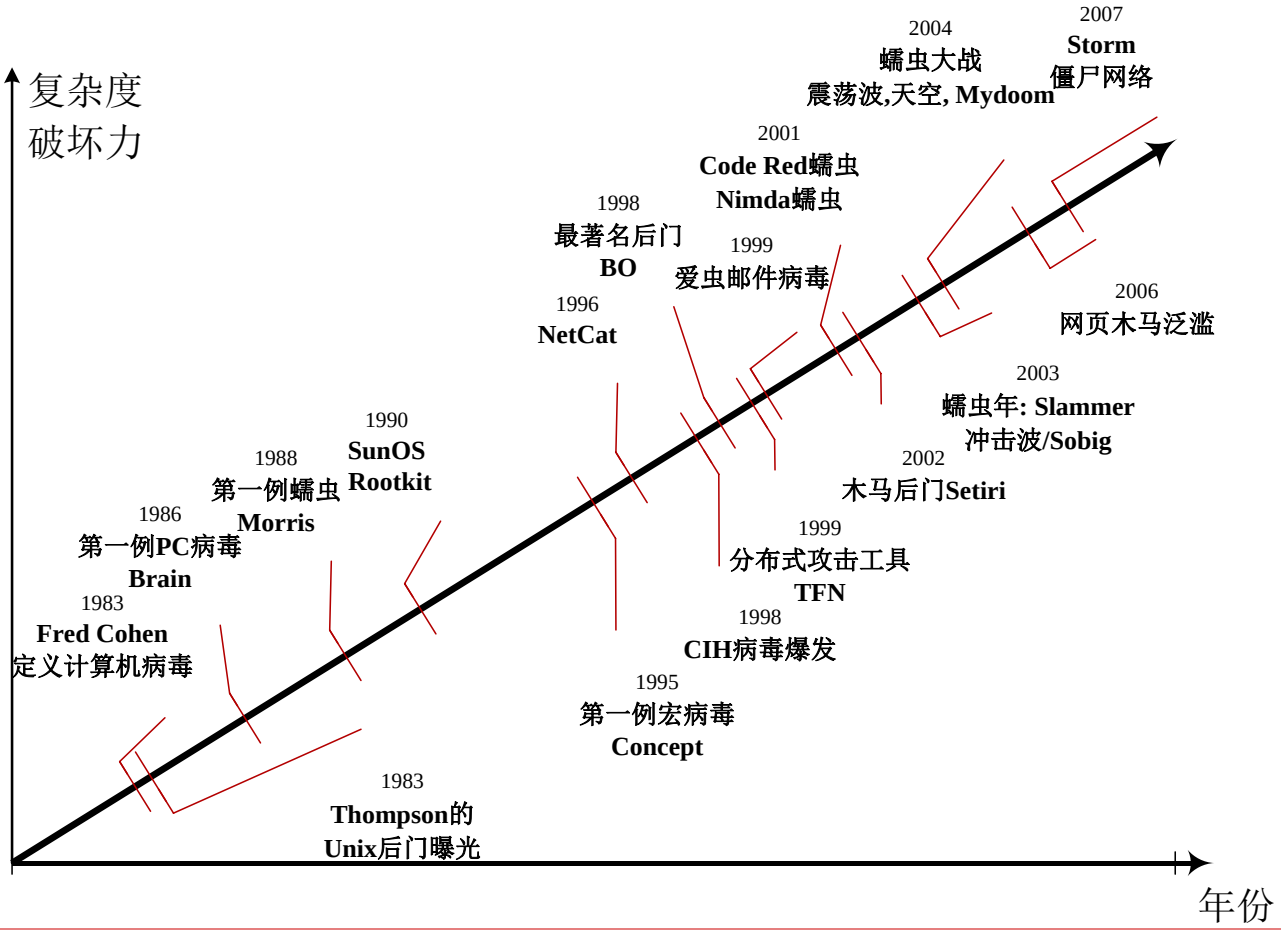
恶意代码的发展史

- 1949年: Von Neumann提出计算机程序自我复制概念
- 1960年: 康维编写出“生命游戏”, 1961年AT&T实验室程序员编写出“Darwin”游戏，通过复制自身来摆脱对方控制
- 1970s早期: 第一例病毒Creeper在APANET上传播
- 1983年: Fred Cohen给出计算机病毒定义
- 1983年: 最著名的Backdoor, Thompson Ken (October 1983). "Reflections on Trusting Trust" (PDF). *1983 Turing Award Lecture*, ACM.
- 1986年: 第一例PC病毒Brain
- 1988年: 第一例蠕虫Morris Worm
- 1990年: SunOS rootkit
- 1995年: Concept宏病毒
- 1998年: CIH病毒－首例破坏计算机硬件的病毒
- 1998年: 最著名的后门软件－*Back Orifice*

恶意代码的发展史(2)

- 1999-2000年：邮件病毒/蠕虫, Melissa, ILOVEYOU
- 2001年(蠕虫年)：Code Red I/II, Nimda
- 2002年：反向连接木马Setiri, …
- 2003年-2004年：蠕虫大爆发
  - 2003: Slammer/Blaster/Nachi/Sobig/…
  - 2004: Mydoom/Witty/Sasser/Santy/…
- 2007-2008年：Storm worm
  - 基于Overnet构建了Stromnet, 一个专属的P2P网络

恶意代码发展史上著名的案例



国内著名的恶意代码实例与事件

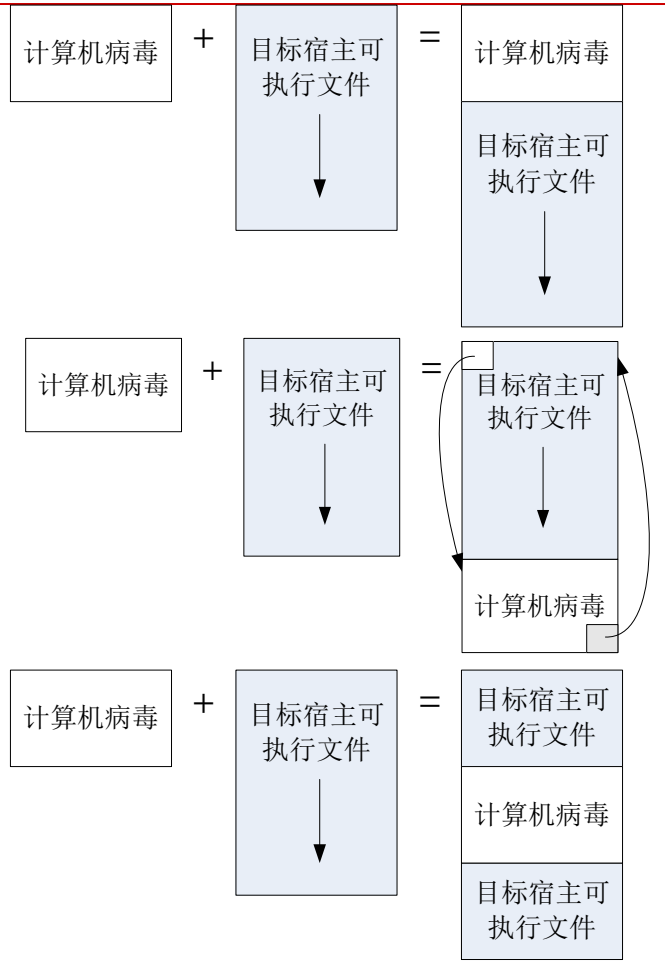
- **1986**年，中国公安部成立计算机病毒研究小组
- **1989**年，国内首例病毒攻击事件，**Kill**发布
- **90**年代，反病毒业界逐步形成
  - 冠群金辰、瑞星、江民、金山
- **90**年代末新世纪初，本土化恶意代码流行
  - **1998- CIH**病毒
  - **1999-**冰河
  - **2003-**灰鸽子
  - **2004-**证券大盗
  - **2007-2008**： 熊猫烧香， 机器狗、磁碟机...

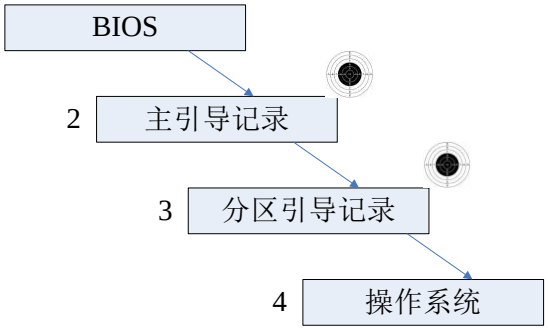
计算机病毒

- 定义
  - 计算机病毒是一种能够自我复制的代码，通过将自身嵌入其他程序进行感染，而感染过程通常需要人工干预才能完成
- 特性
  - 感染性：最本质的特性
  - 潜伏性
  - 可触发性
  - 破坏性
  - 衍生性

计算机病毒的感染机制

- 感染可执行文件
  - 前缀感染
  - 后缀感染
  - 插入感染
- 感染引导扇区
- 感染数据文件—宏指令





计算机病毒的传播机制

- 计算机病毒 **VS.** 蠕虫
  - 病毒: 借助人类帮助从一台计算机传至另一台计算机
  - 蠕虫: 主动跨越网络传播
- 传播方式
  - 移动存储: 软盘→U盘
  - 电子邮件及其下载: 邮件病毒
  - 文件共享: **SMB**共享服务、**NFS**、 **P2P**

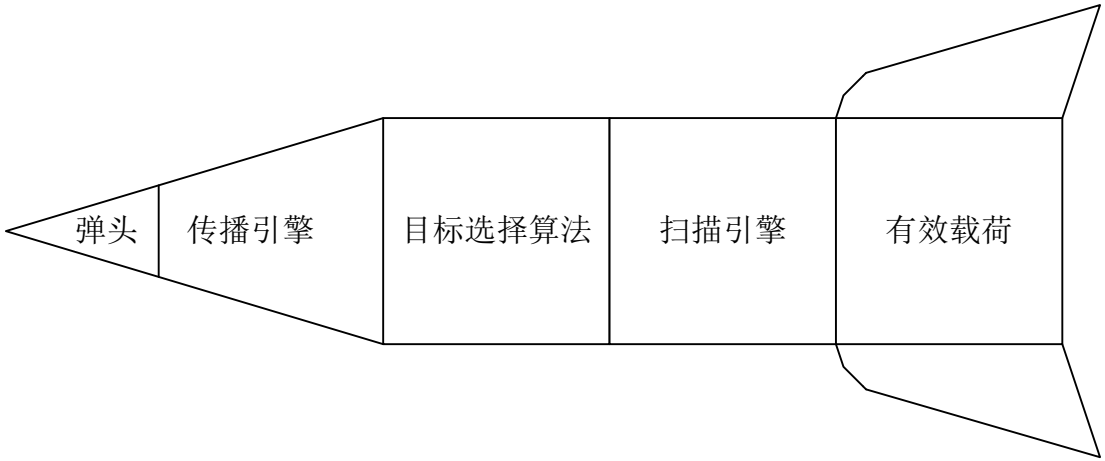
网络蠕虫

- 网络蠕虫定义特性
  - 主动传播性

恶意代码类型	计算机病毒	网络蠕虫
复制性	自我复制，感染性	自我复制，感染性
定义特性	感染宿主文件/扇区	通过网络的自主传播
宿主	需要寄生宿主	不需要宿主，独立程序
传播路径	感染文件或扇区，通过文件交换或共享传播	直接通过网络传播，包括内网和因特网
传播是否需要用户交互	通常需要用户交互，例如运行一个程序或打开文档	一般来说，不需要用户交互，通过目标系统上的安全漏洞或错误配置进行传播。但对于一小部分蠕虫，例如邮件蠕虫，用户交互是必要的。

- 网络蠕虫传播机制
  - 主动攻击网络服务漏洞
  - 通过网络共享目录
  - 通过邮件传播

网络蠕虫的组成

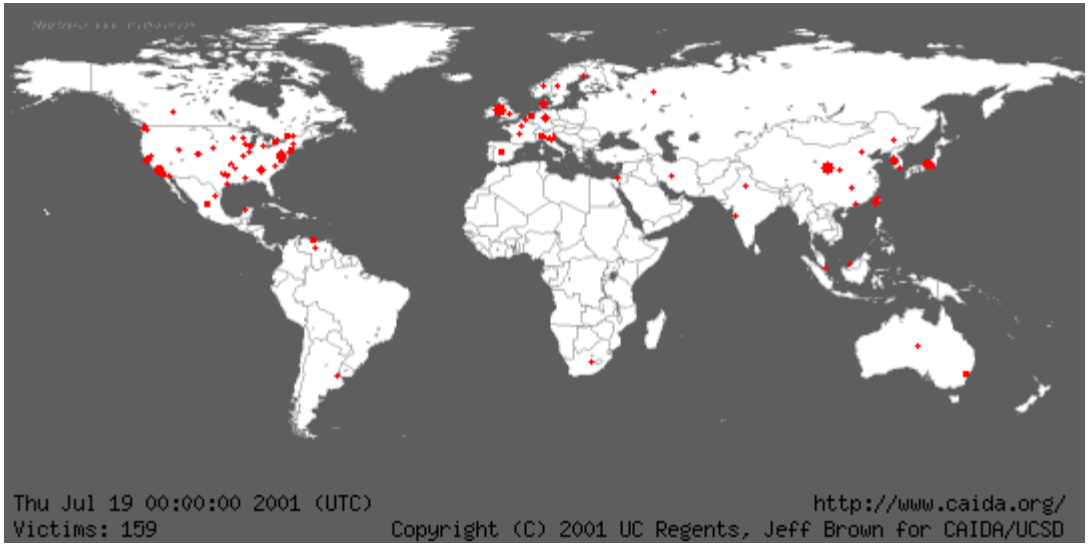


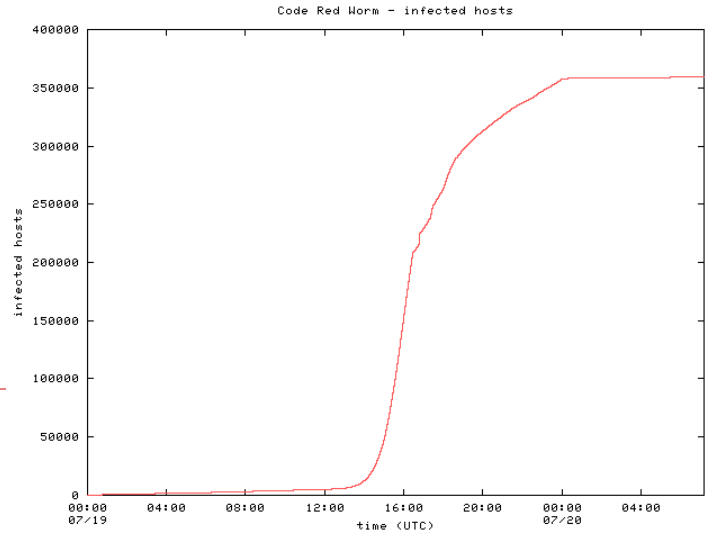
- 蠕虫的“弹头”
  - 渗透攻击模块
- 传播引擎
  - **FTP/TFTP/HTTP/SMB/**直接传送/单包
- 目标选择算法+扫描引擎
  - 扫描策略
- 有效负荷(攻击负荷)
  - **Payload:** 传播自身, 开放后门, **DDoS**攻击...

“红色代码”蠕虫

- 2001年7月19日，”红色代码”蠕虫爆发
- 在红色代码首次爆发的短短9小时内，以迅雷不及掩耳之势迅速感染了250,000台服务器（通过IIS服务漏洞）
- 最初发现的红色代码蠕虫只是篡改英文站点主页，显示“*Welcome to http://www.worm.com! Hacked by Chinese!*”
- 随后的红色代码蠕虫便如同洪水般在互联网上泛滥，并会在每月20日～28日对白宫的WWW站点的IP地址发动DoS攻击，使白宫的WWW站点不得不全部更改自己的IP地址。

“红色代码”的蔓延速度

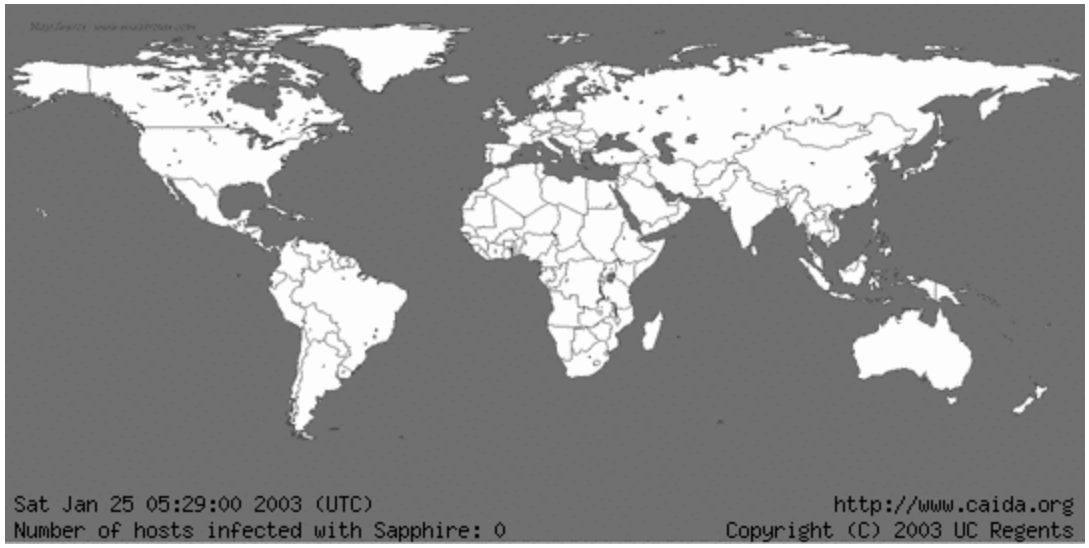

Thu Jul 19 00:00:00 2001 (UTC)
Victims: 159
http://www.caida.org/
Copyright (C) 2001 UC Regents, Jeff Brown for CAIDA/UCSD


Code Red Worm - infected hosts
infected hosts
400000
350000
300000
250000
200000
150000
100000
50000
0
00:00
07/19
04:00
08:00
12:00
16:00
20:00
00:00
07/20
04:00
time (UTC)

SQL Slammer蠕虫

- Slammer的传播数度比“红色代码”快两个数量级
- 在头一分钟之内，感染主机数量每8.5秒增长一倍；
- 3分钟后该病毒的传播速度达到峰值（每秒钟进行5500万次扫描）；
- 接下来，其传播速度由于自身挤占了绝大部分网络带宽而开始下降；
- 10分钟后，易受攻击的主机基本上已经被感染殆尽

30分钟后
在全球的感染面积



超级蠕虫

- 跨平台蠕虫: **Sadmind/IIS蠕虫**
- 多探测目标蠕虫
- **Zero-day探测蠕虫**
- 快速传播蠕虫: **Slammer, Warhol(15m)**
- 多态蠕虫**(polymorphic): AdMutate多态引擎**
- 变形蠕虫**(metamorphic)**
- 真正恶意的蠕虫
  - 毁坏性的攻击负荷**(DDoS,** 格盘, 破坏硬件**)**
- **"**良性**"**蠕虫: 对抗蠕虫的蠕虫
  - **Nachi VS. Blaster**

后门

- **War Games**
  - **JoShua**
  - **Falken教授留下的WOPR系统访问后门**
- **"Reflections on Trusting Trust" (PDF).**
  - **Ken Thompson, *1983 Turing Award Lecture*, ACM.**
  - **One Unix host with Ken's backdoor in Bell Labs, they never found the attack**
  - **Trust, but Test!**
  - **Source code Auditing is not enough**
- 后门的定义
  - 后门是允许攻击者绕过系统常规安全控制机制的程序，按照攻击者自己的意图提供通道。

后门

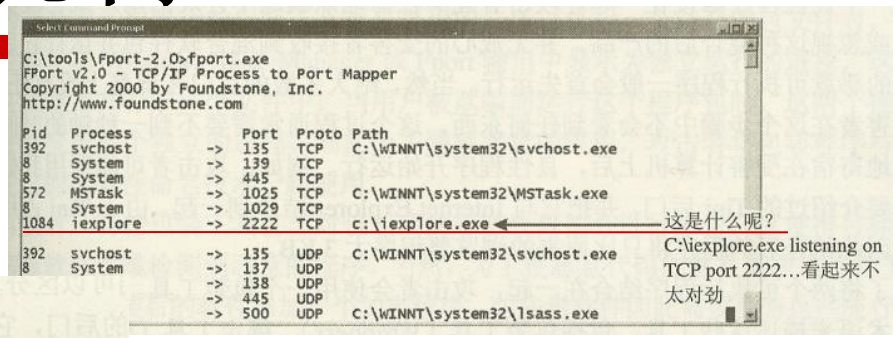
- 后门类型
  - 本地权限提升、本地帐号
  - 单个命令的远程执行
  - 远程命令行解释器访问－**NetCat**
  - 远程控制**GUI**－**VNC, BO,** 冰河**,** 灰鸽子
  - 无端口后门**: ICMP**后门**,** 基于**Sniffer**非混杂模式的后门，基于**Sniffer**混杂模式的后门
- 自启动后门
  - **Windows**：自启动文件/文件夹；注册表自启动项；计划任务
  - **Linux/Unix**：**inittab, rc.d/init.d,** 用户启动脚本**, cron**计划任务

木马

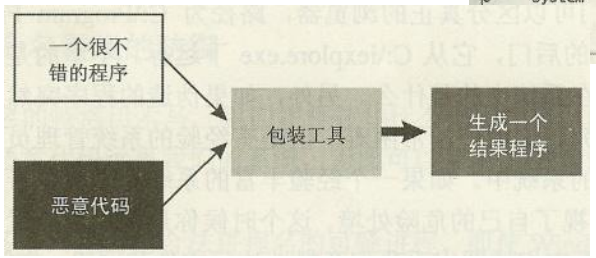
- 特洛伊木马(Trojan Horse)
  起源 - 特洛伊战争
- 木马: 特洛伊木马(Trojans)
  - 定义: 看起来具有某个有用或善意目的，但实际掩盖着一些隐藏恶意功能的程序。
  - 错误观点: 提供对受害计算机远程控制的任何程序，或受害计算机上的远程命令行解释器看做木马，他们应被视为后门。
  - 如果将后门工具伪装成良性程序，才具备真正的木马功能。

木马的常见伪装机制

- 命名伪装



- 软件包装



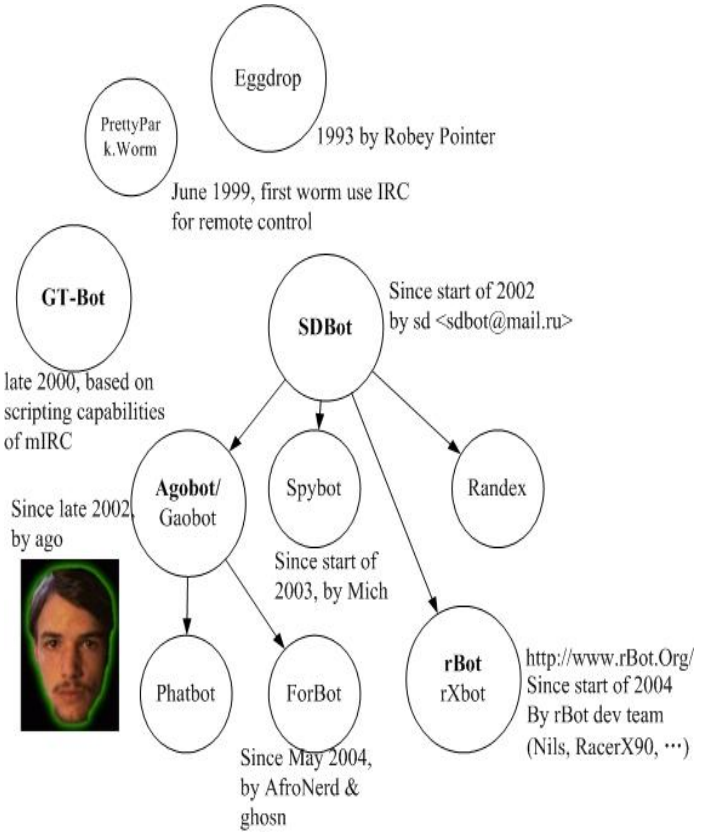
- 木马化软件发行站点
  - **Tcpdump/libpcap木马化事件**
- 代码"Poisoning"
  - 软件开发者/厂商有意给代码加入后门
  - "复活节彩蛋": Excel 2000中隐藏的赛车游戏

僵尸程序与僵尸网络

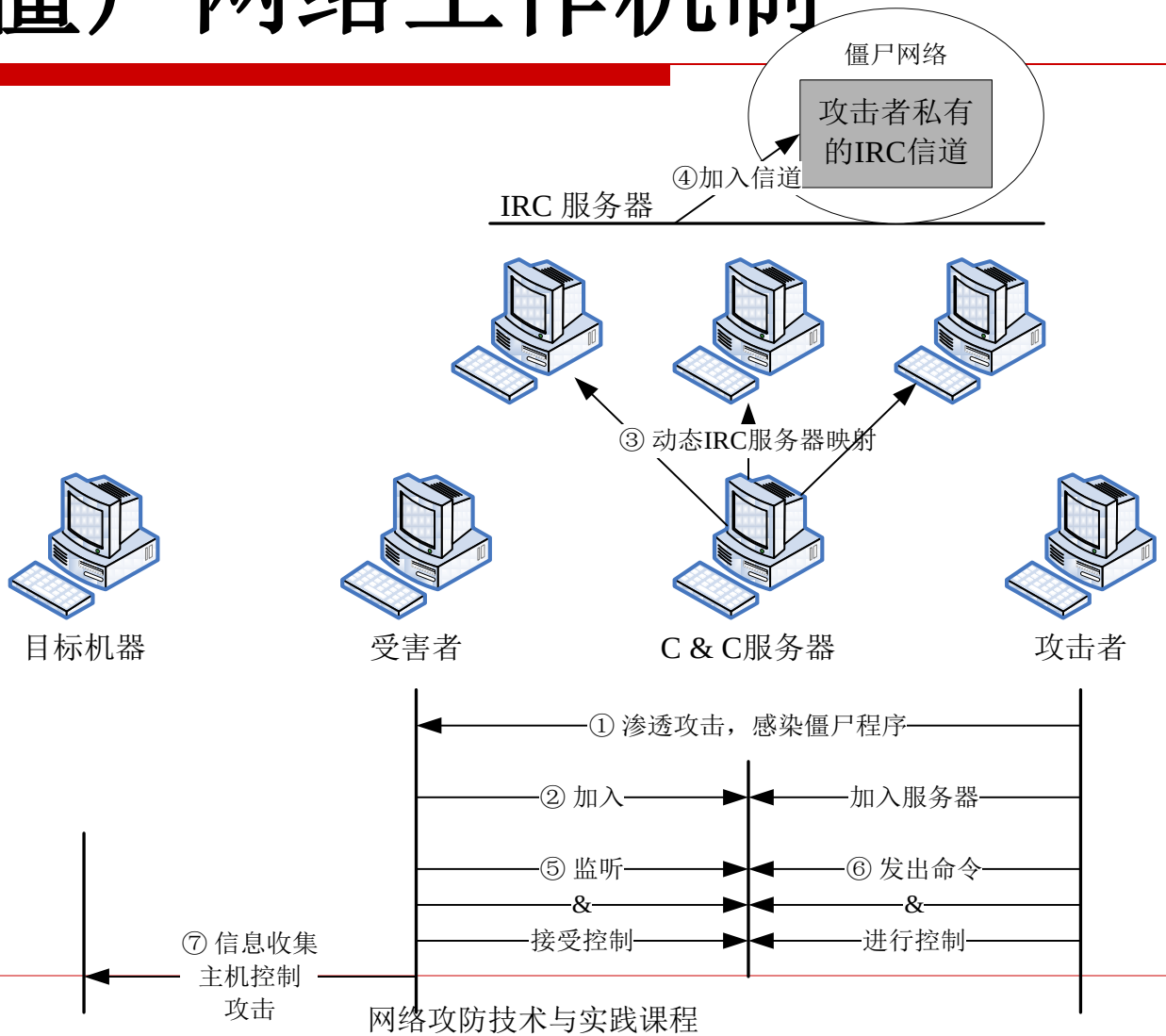
- 僵尸程序(Bot)
  - 来自于robot, 攻击者用于一对多控制目标主机的恶意代码
- 僵尸网络（BotNet）
  - 攻击者出于恶意目的，传播僵尸程序控制大量主机，并通过一对多的命令与控制信道所组成的网络。
  - 定义特性：一对多的命令与控制通道的使用。
- 僵尸网络危害－提供通用攻击平台
  - 分布式拒绝服务攻击
  - 发送垃圾邮件
  - 窃取敏感信息
  - 点击欺诈...

僵尸网络类型

- **IRC僵尸网络**
  - 传统僵尸网络-基于**IRC**互联网实时聊天协议构建
  - 著名案例: **sdbot, agobot**等
- **HTTP僵尸网络**
  - 僵尸网络控制器—**Web**网站方式构建
  - 僵尸程序中的命令与控制模块：通过**HTTP**协议向控制器注册并获取控制命令
  - 著名案例: **bobax, rustock,** 霸王弹窗
- **P2P僵尸网络**
  - 命令与控制模块的实现机制—**P2P**协议
  - **P2P**僵尸程序同时承担客户端和服务器的双重角色
  - 著名案例: **storm worm**



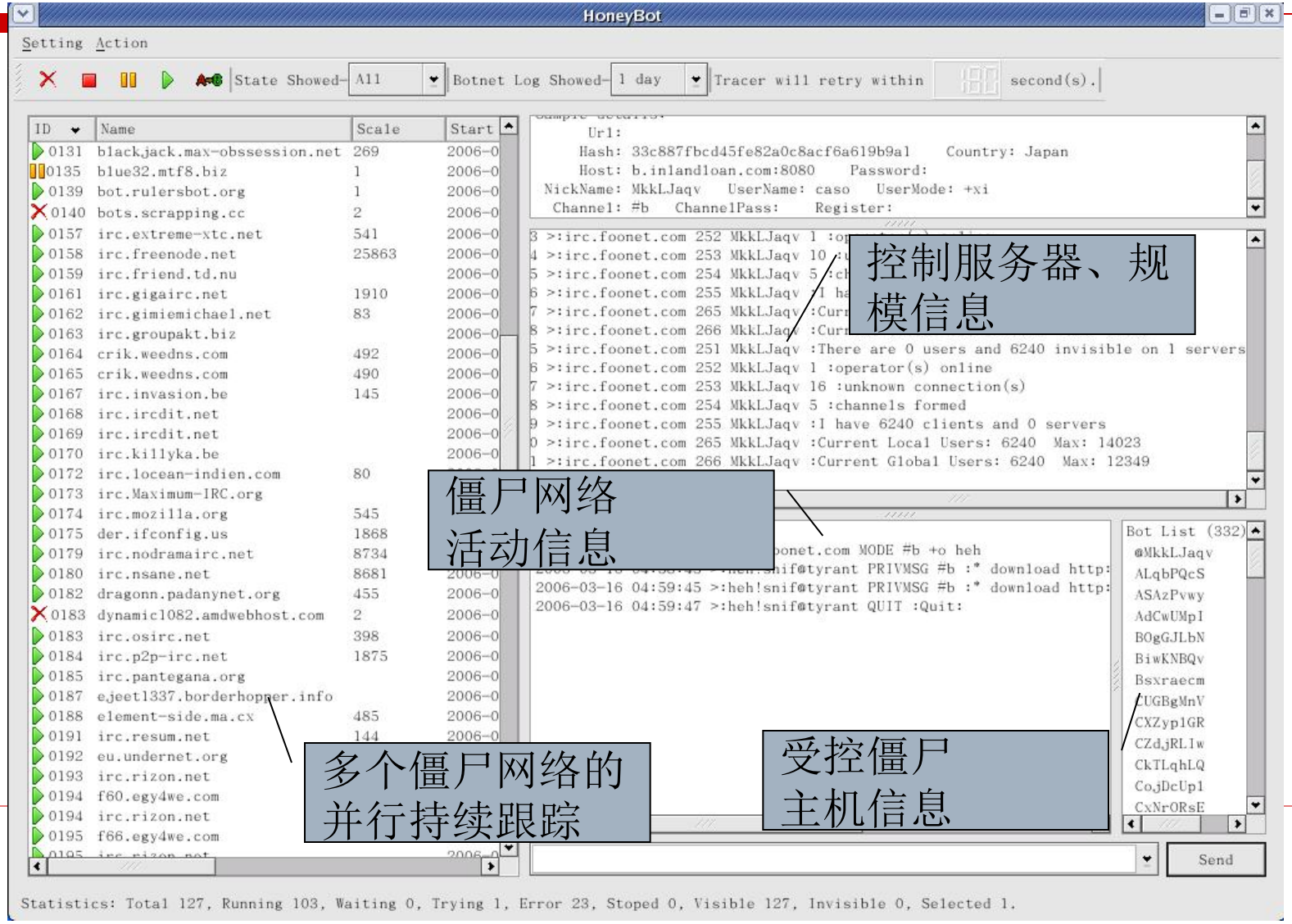
IRC僵尸网络工作机制



HoneyBot僵尸网络跟踪系统

- 僵尸网络跟踪框架
  - 多线程并行持续跟踪（多线程调度及管理）
  - 隐蔽性: 支持SOCKS代理
  - 界面友好性: 基于Qt的图形界面
  - 跟踪数据深入分析处理: 数据库输入/输出
  - 跟踪数据全面性: 规模、服务器信息、僵尸程序列表、控制指令、迁移轨迹
- 僵尸网络跟踪组件
  - 针对不同的僵尸网络控制协议
  - IRC僵尸网络跟踪组件

HoneyBot僵尸网络跟踪实现效果



僵尸网络跟踪—僵尸网络列表

序号	控制主机	端口	密码	昵称	用户名	模式	
1	freedom.dude-x.net	65535		[F][Shit]-389176359			#
2	squall.h1x.com	6667	pass	Macdonald	m	+xB	#
3	deathfield.com	3920		CHN	ixqpqvg		#
4	creative.proircd.net	6667		[RAPEDV1]-0068			#
5	izzla.chickenkiller.co	32000	123456789	[0||659399]	XP-5090	+iwB	#
6	ome.paltalkdc.co	7000		LL-8034002488	ezkieyacagiz	+x+i	#
7	im.egy4we.co	7000		[fo]80340024	ezkieyacag	+xi	#
8	ia.dcznet.u	65267	r00t	|2071021336	fhqgcrkusu	+n+U	#
9	im.egy4we.co	7000		[fo]80340024	ezkieyacag	+xi	#
10	im.egy4we.co	7000		[fo]80340024	ezkieyacag	+xi	#
11	4.206.189.22	6667	34fn2m3kl	[0||613353]	XP-9422	[0||613353] to	#
12	im.egy4we.co	7000		[fo]80340024	ezkieyacag	+xi	#
13	0.sytes.ne	58	?* IRC: Sets the usermode for us	[T]-803400248	ezkieyacagi	+i	#
14	nfo.fastsuper.co	6667	nadjoe	[0||637399]	XP-5090	is	#
15	ree.avautoupdate.inf	8080	blue00	[0||221038]	XP-3822	[0||221038] to	#
16	rbin.hp-slo.ne	8885	102	|530230	sggczo	-x+i	#
17	ome.paltalkdc.co	7000		R-8034002488	ezkieyacagiz		
18	rleet.dynup.ne	8641					
19	ome.paltalkdc.co	7000		LL-8034002488	ezkieyacagiz	+x+i	#
20	4.206.189.22	6667	l0ck3d	[0||631393]	XP-9486	[0||631393] to	#

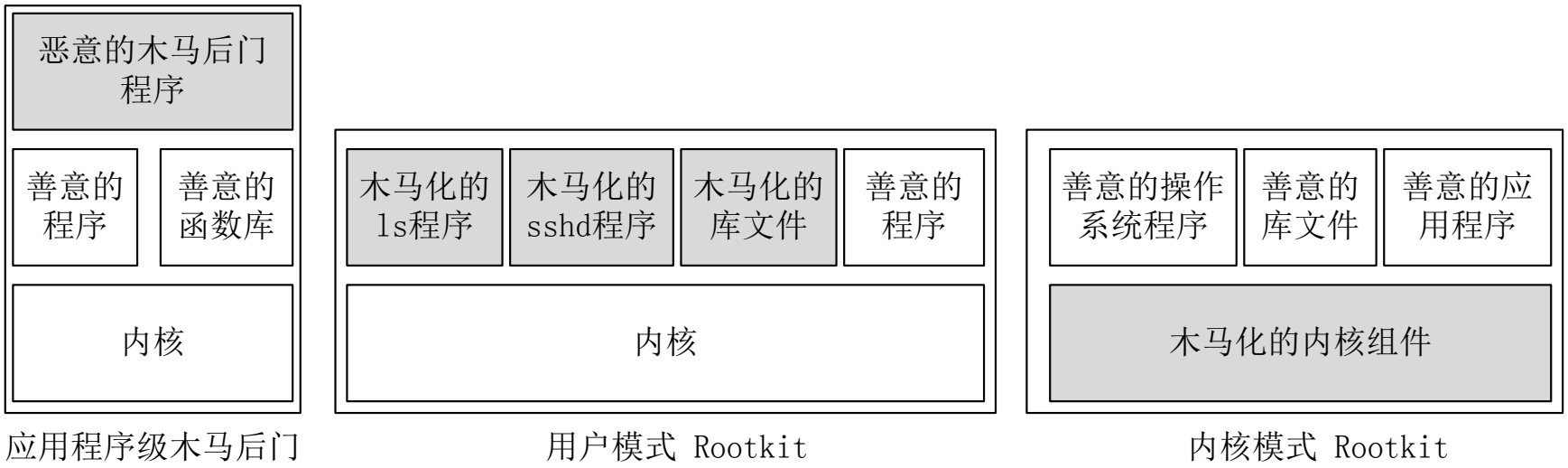
僵尸主机地域分布


Hades集中控制管理平台
Hades集中控制管理平台
USERNAME Hades
修改帐户信息
退出
控制面板首页
全部展开 | 全部折叠
恶意软件捕获系统管理平台
分布式站点管理控制
恶意软件样本库
恶意软件样本捕获统计
站点样本捕获统计
恶意软件分析报告
僵尸网络追踪
跟踪僵尸网络列表
僵尸网络跟踪趋势
僵尸网络控制端口分布
僵尸网络地区分布
僵尸网络控制点地域分布
僵尸网络跟踪日志
僵尸网络规模分布
僵尸网络规模曲线
僵尸主机地域分布
僵尸主机地域分布
Map
Satellite
Greenland
Iceland
Finland
Sweden
Norway
Russia
United Kingdom
Canada
United States
North Atlantic Ocean
Kazakhstan
Mongolia
China
Japan
Italy
Turkey
Iraq
Iran
Afghanistan
Pakistan
India
Myanmar
Thailand
Indonesia
Algeria
Libya
Egypt
Saudi Arabia
Mauritania
Mali
Niger
Chad
Sudan
Nigeria
Ethiopia
Kenya
DR Congo
Tanzania
Mexico
Cuba
Venezuela
Colombia

Rootkit

- **Rootkit的定义**
  - 一类隐藏性恶意代码形态，通过修改现有的操作系统软件，使攻击者获得访问权并隐藏在计算机中。
- **Rootkit与特洛伊木马、后门**
  - **Rootkit**也可被视为特洛伊木马
    - 获取目标操作系统上的程序或内核代码，用恶意版本替换它们
  - **Rootkit**往往也和后门联系在一起
    - 植入**Rootkit**目的是为攻击者提供一个隐蔽性的后门访问
  - 定义特性：隐藏性
- **Rootkit分类: 用户模式、内核模式**

两类Rootkit和普通的应用程序级木马后门之间的位置对比



- 应用程序级木马后门：操作系统之上由攻击者添加至受害计算机的恶意应用程序
- 用户模式Rootkit：木马化操作系统用户模式应用程序
- 内核模式Rootkit：对内核组件的恶意修改和木马化

用户模式Rootkit

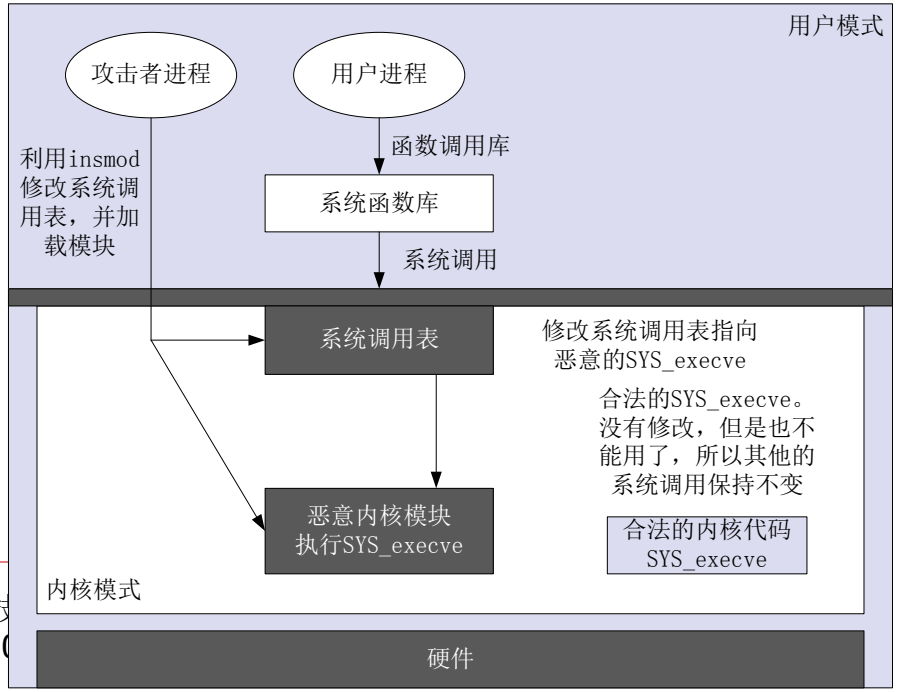
- 用户模式Rootkit
  - 恶意修改操作系统在用户模式下的程序/代码，达到隐藏目的

- **Linux**
  - **LRK (Linux RootKit)**
  - **URK (Universal RootKit)**：适用于多种**Unix**平台
  - **Linux**用户模式**Rootkit**的防御: 文件完整性检测**Tripwire**, 专用检测工具**chkrootkit**

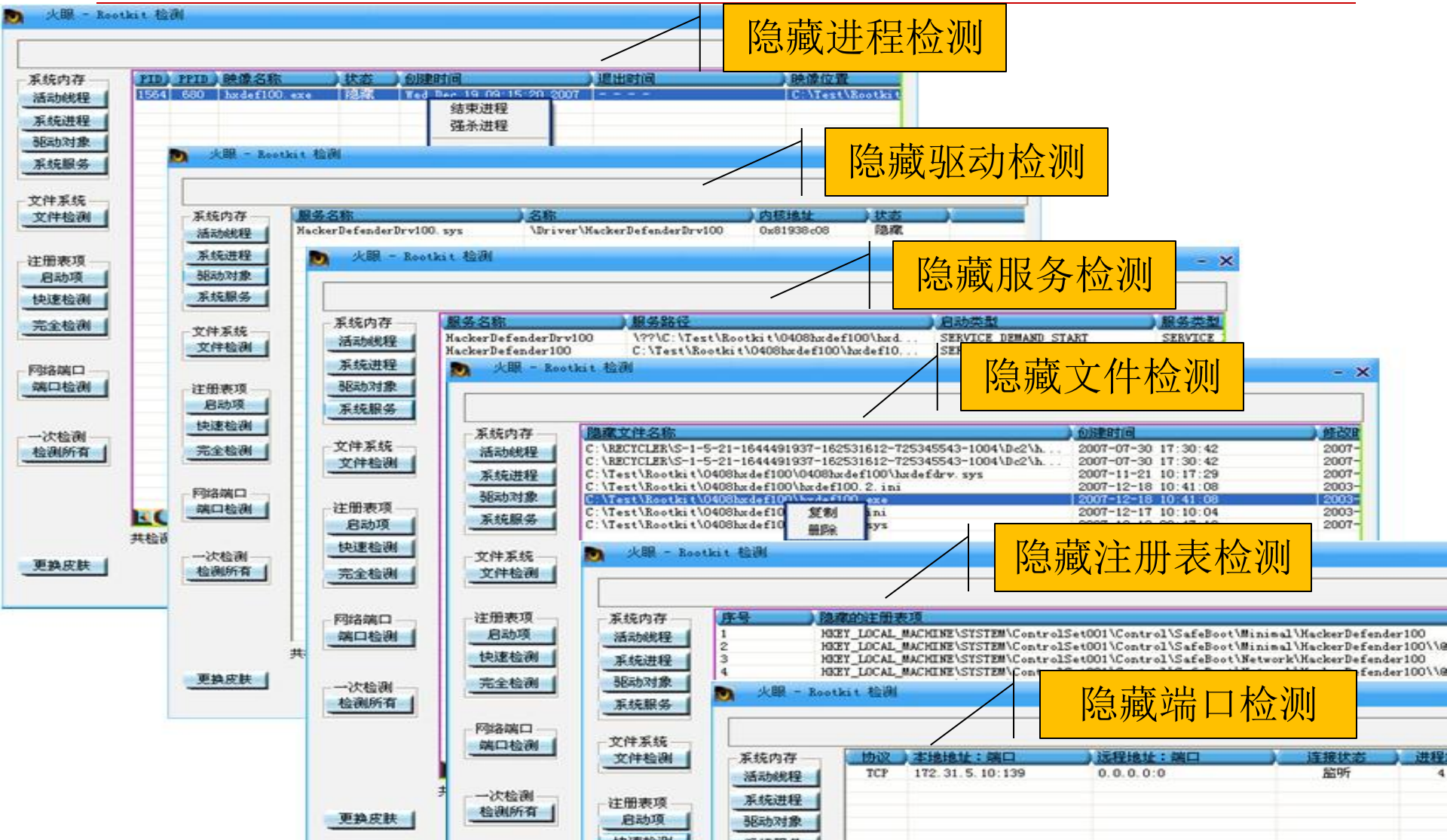
- **Win32**
  - **FakeGINA, AFX Rootkit (DLL**注入、**API Hooking)**

内核模式Rootkit

- 内核模式Rootkit
  - 恶意修改操作系统内核，从而达到更深的隐藏和更强的隐蔽性
- Linux内核模式Rootkit
  - Adore, Adore-ng, KIS (Kernel Intrusion System)
  - 防御: SELinux, LIDS, … 检测: chkrootkit, KSTAT, …
- Win32内核模式Rootkit
  - NT Rootkit, Fu Rootkit
- 虚拟机模式Rootkit
  - 黑客帝国《Matrix》
  - Linux: UML, KML
  - Win32: VM Rootkit



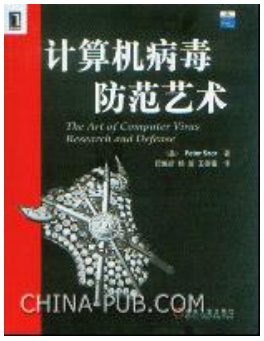
火眼Rootkit检测软件



恶意代码相关推荐书籍

- 基础
  - **Ed. Skoudis, Lenny Zelter, Malware: Fighting Malicious Code(决战恶意代码)** 电子工业出版社.

- 进阶
  - **Peter Szor, The Art of Computer Virus Research and Defense(计算机病毒防范技术), 机械工业出版社.**



  - 段钢(看雪学院), 加密与解密(第三版), 电子工业出版社.

内容

1. 恶意代码基础知识
2. 恶意代码分析技术
   - 课题实践：恶意代码静态分析
   - 课堂实践：分析Crackme程序
3. 课外作业：分析一个自制恶意代码样本

恶意代码分析

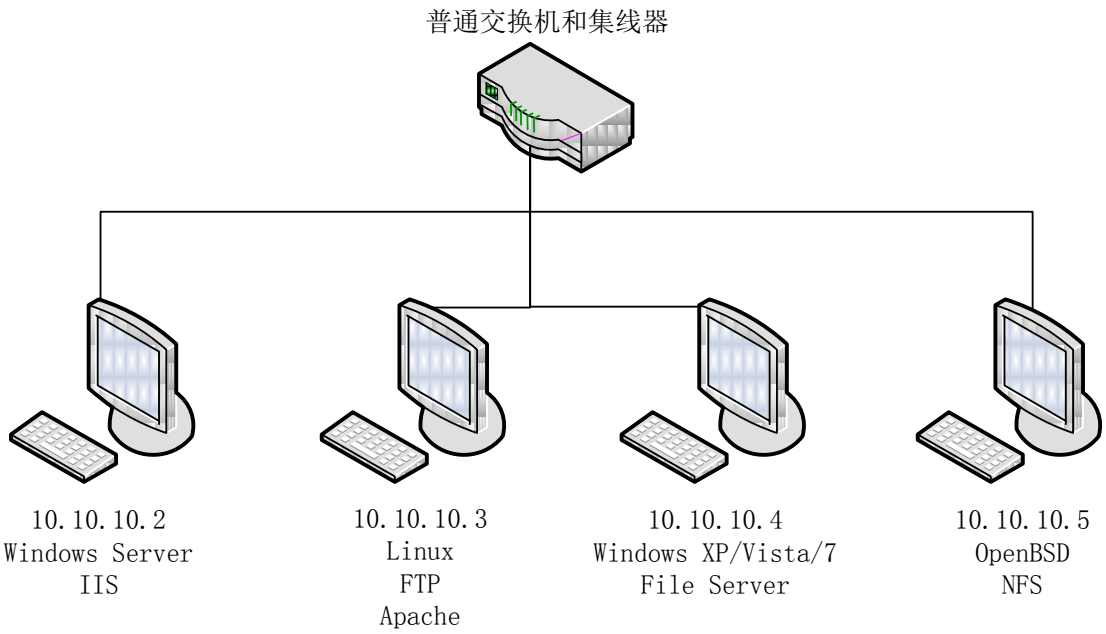
- 没有分析的生活是毫无意义的。- 苏格拉底, 公元前469-399
- 恶意代码分析与良性代码分析
  - 相同点：通用代码分析技术

区别项	恶意代码分析	良性代码分析
目的公开性	目的未知，需分析和推测其目的	一般情况下，目的是公开且明确的，可辅助分析过程
目的恶意性	恶意目的，需要受控环境	良性，无需受控环境
是否存在源码	绝大多数情况无源码，二进制分析	开源软件存在源码，源码分析；闭源软件则需二进制分析
使用对抗分析技术	各种多样化对抗分析，博弈问题	一般无对抗分析，商业软件也引入对抗分析保护产权

- 恶意代码分析的关键点

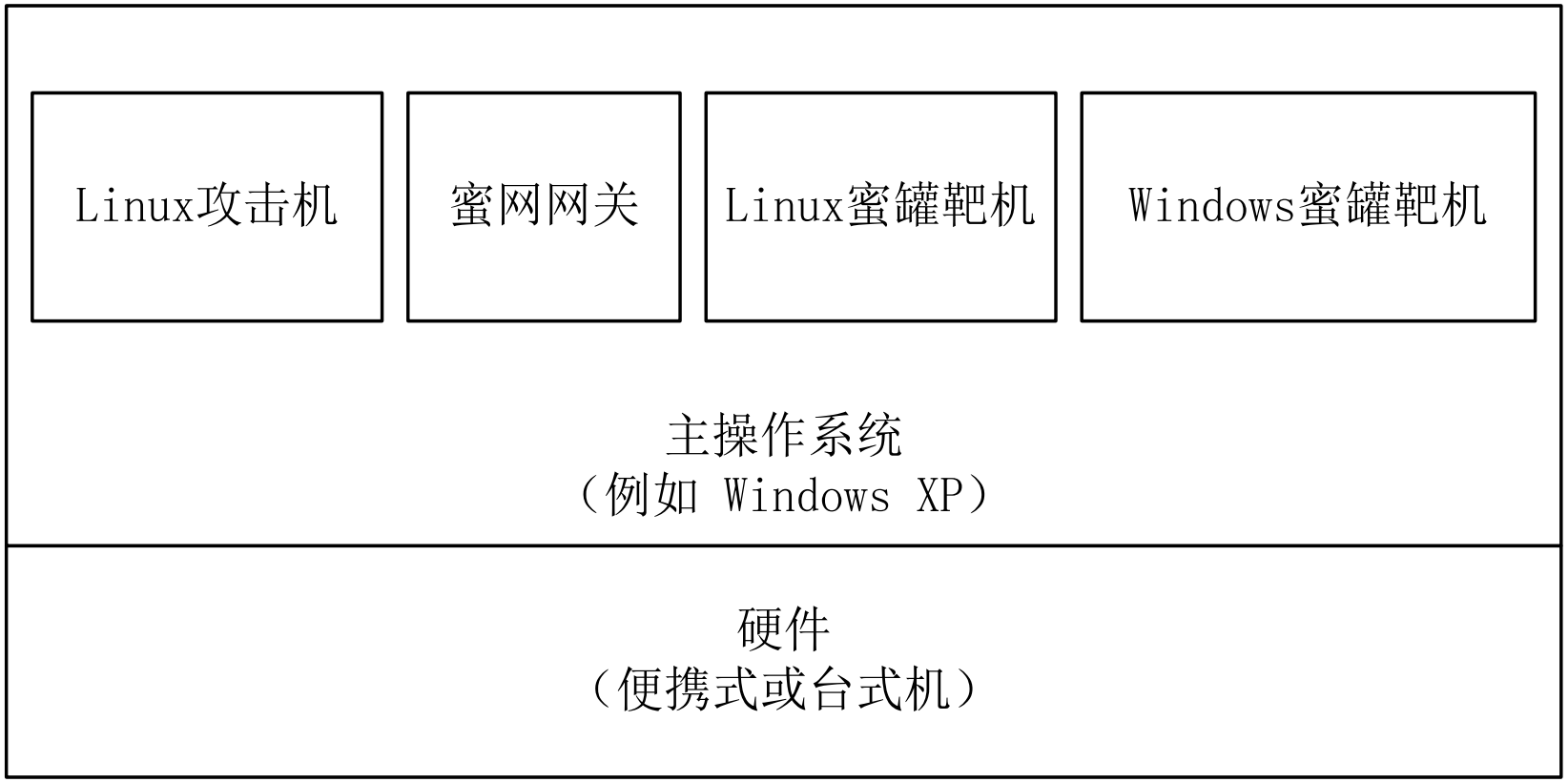
  构建受控的分析环境, 通过静态/动态方法实施分析

病毒发烧友的恶意代码分析环境



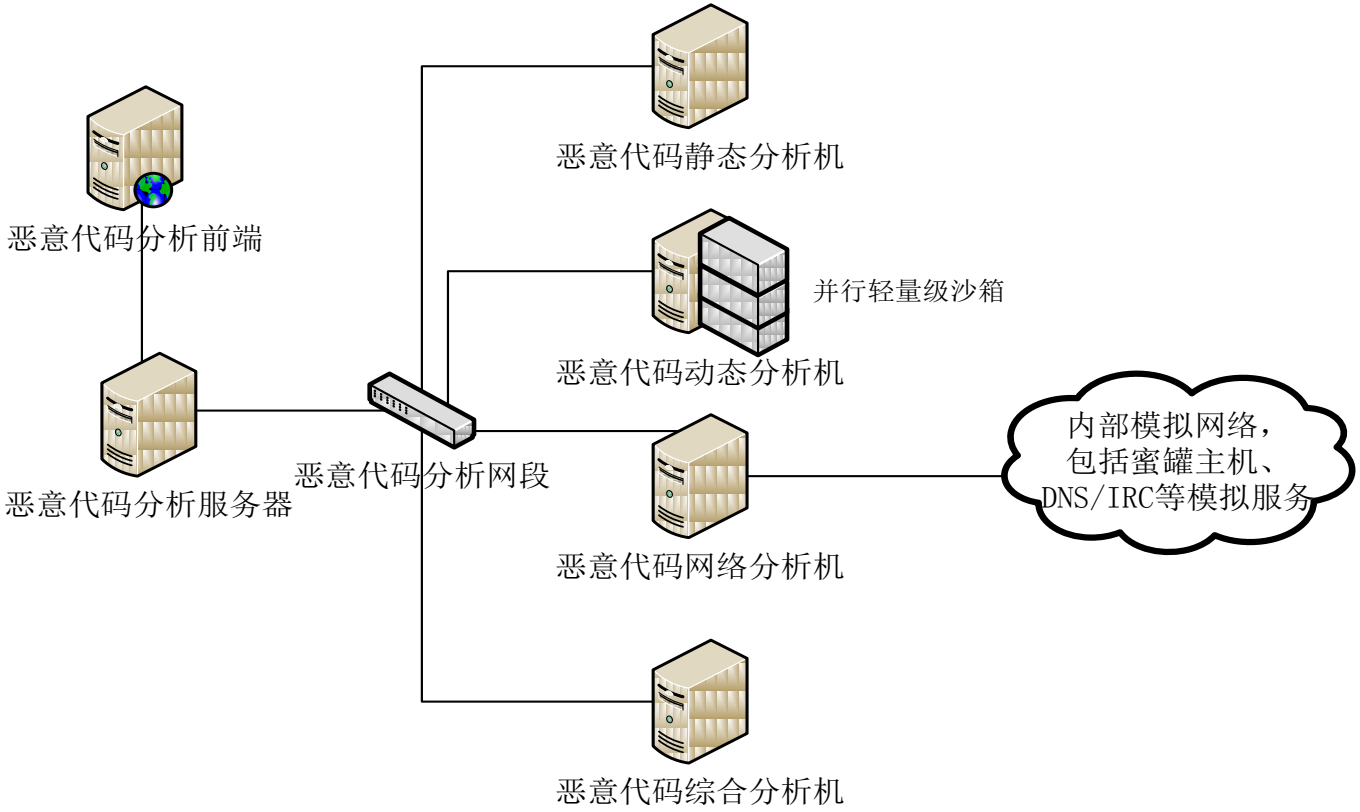
- 硬盘保护卡
  - 快速恢复

基于虚拟化构建便携式分析环境

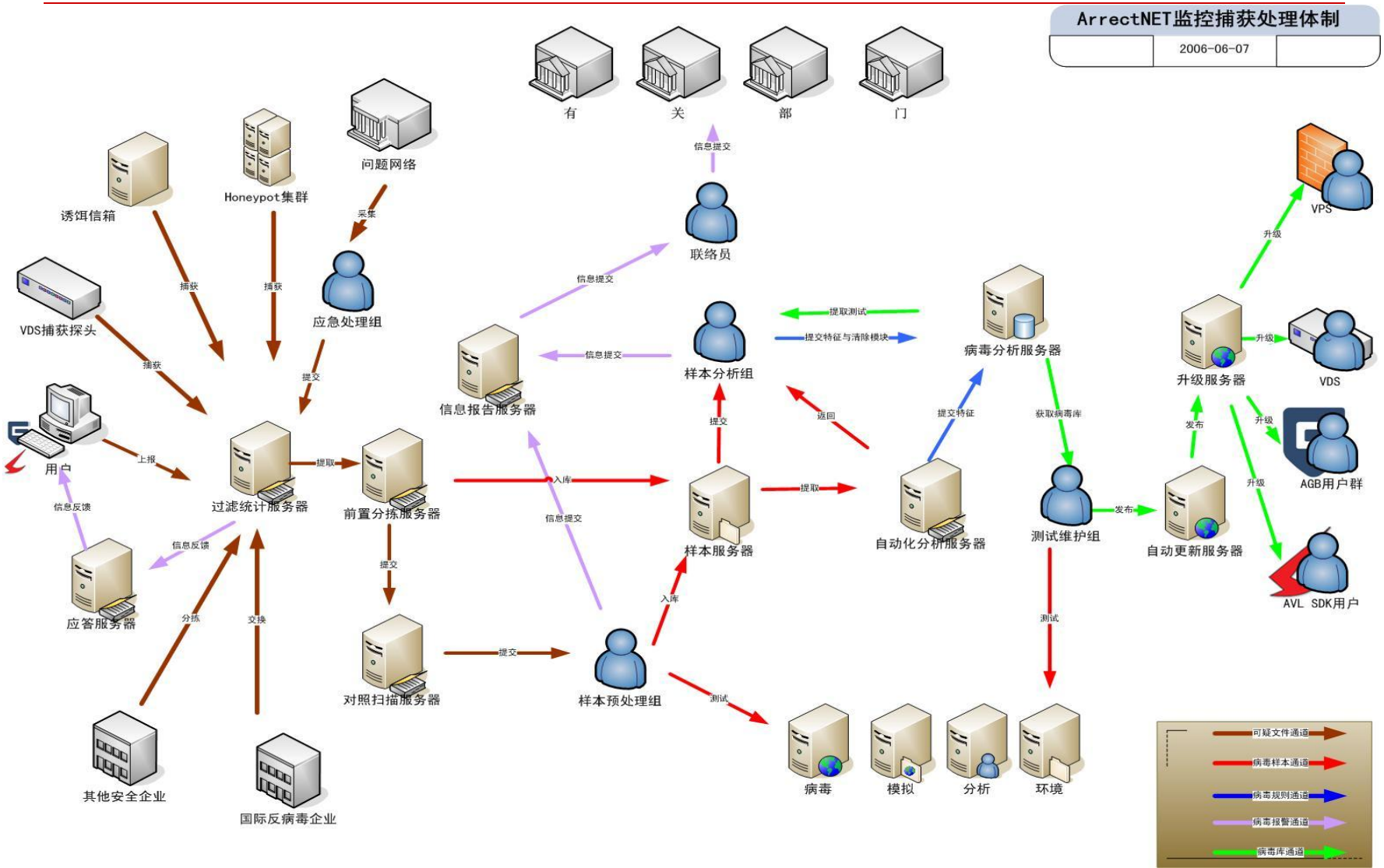


用于研究的恶意代码自动分析环境

- 国家242信息安全计划项目



反病毒厂商的恶意代码分析环境



恶意代码分析方法概述

- 静态分析
  - 通过反病毒引擎扫描识别已知的恶意代码家族和变种名
  - 逆向分析恶意代码模块构成，内部数据结构，关键控制流程等，理解恶意代码的机理，并提取特征码用于检测。
- 动态分析
  - 通过在受控环境中执行目标代码，以获取目标代码的行为及运行结果。

恶意代码静态分析方法列表

分析方法	目的	使用工具	难度
恶意代码扫描	标识已知恶意代码	反病毒引擎, VirusTotal	低
文件格式识别	确定攻击平台和类型	file, peid, FileAnalyzer	低
字符串提取	寻找恶意代码分析线索	strings	低
二进制结构分析	初步了解二进制文件结构	binutils (nm, objdump)	中
反汇编	二进制代码->汇编代码	IDA Pro, GDB, VC, …	中高
反编译	汇编代码->高级语言	REC, DCC, JAD, …	中高
代码结构与逻辑分析	分析二进制代码组，理解二进制代码逻辑成结构	IDA Pro, Ollydbg, …	高
加壳识别和代码脱壳	识别是否加壳及类型；对抗代码混淆恢复原始代码	UPX,VMUnpacker,手工	高

恶意代码的扫描

- 使用你的反病毒软件进行检测
  - 卡巴斯基、赛门铁克等
  - 瑞星、金山、江民等
- **VirusTotal**
  - “世界病毒扫描网”
- 开源恶意代码扫描引擎**ClamAV**
  - **http://www.clamav.net/**
  - **You can have your own AV engine with ClamAV.**
- 从反病毒厂商获得已知恶意代码的分析报告和结果 **(Google之)**

VirusTotal对示例恶意代码的识别

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

VT Community

not reviewed

Safety score: -

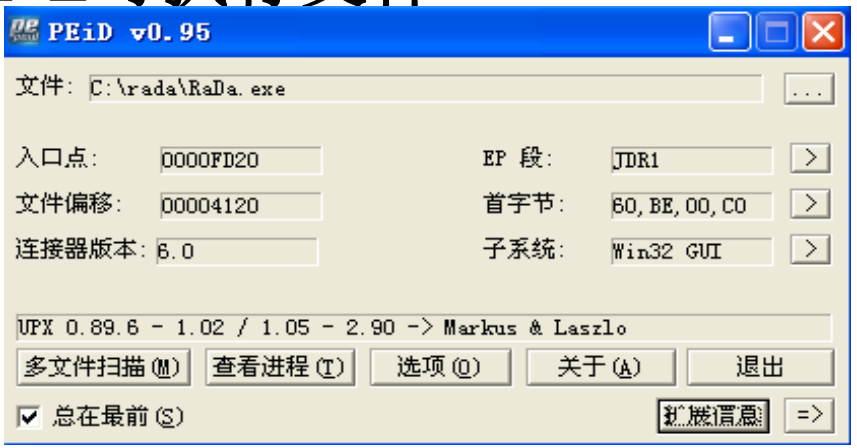
File name: RaDa.exe
Submission date: 2010-10-29 06:15:38 (UTC)
Current status: finished
Result: 41 /43 (95.3%)

Compact　　Print results

Antivirus	Version	Last Update	Result
AhnLab-V3	2010.10.29.00	2010.10.28	-
AntiVir	7.10.13.67	2010.10.28	TR/Rada.A
Antiy-AVL	2.0.3.7	2010.10.29	Trojan/Win32.Rada.gen
Authentium	5.2.0.5	2010.10.29	W32/Trojan2.GKYH
Avast	4.8.1351.0	2010.10.29	Win32:Rada
Avast5	5.0.594.0	2010.10.29	Win32:Rada
AVG	9.0.0.851	2010.10.28	Generic14.AMWE
BitDefender	7.2	2010.10.29	Trojan.Rada.A
CAT-QuickHeal	11.00	2010.10.26	TrojanDDoS.Rada.a
ClamAV	0.96.2.0-git	2010.10.29	DoS.Rada

文件格式确定

- **file:** 确定恶意代码目标平台和文件类型
  - **Linux**平台包含命令：确定文件类型->平台
- **PEID:** 文件类型、编译链接器、是否加壳
  - **Win32**平台针对**PE**可执行文件



- **File Analyzer**
  - 分析**Win32**平台窗口程序中包含的特殊文件

Strings命令-查看可打印字符串

- Strings
  - Linux自带/Windows sysinternals strings工具
  - IDA Pro
- 可能获得的有用信息
  - 恶意代码实例名
  - 帮助或命令行选项
  - 用户会话
  - 后门口令
  - 相关URL信息、Email地址
  - 库、函数调用...
- 结合grep/find寻找关注信息
- 显示乱码？没有任何有用信息？
  - Strip程序删除信息
  - 加壳或变形
  - 字符串Unicode编码:
    - strings -a
    - IDA Pro: strings view

Strings对示例恶意代码进行字符串提取

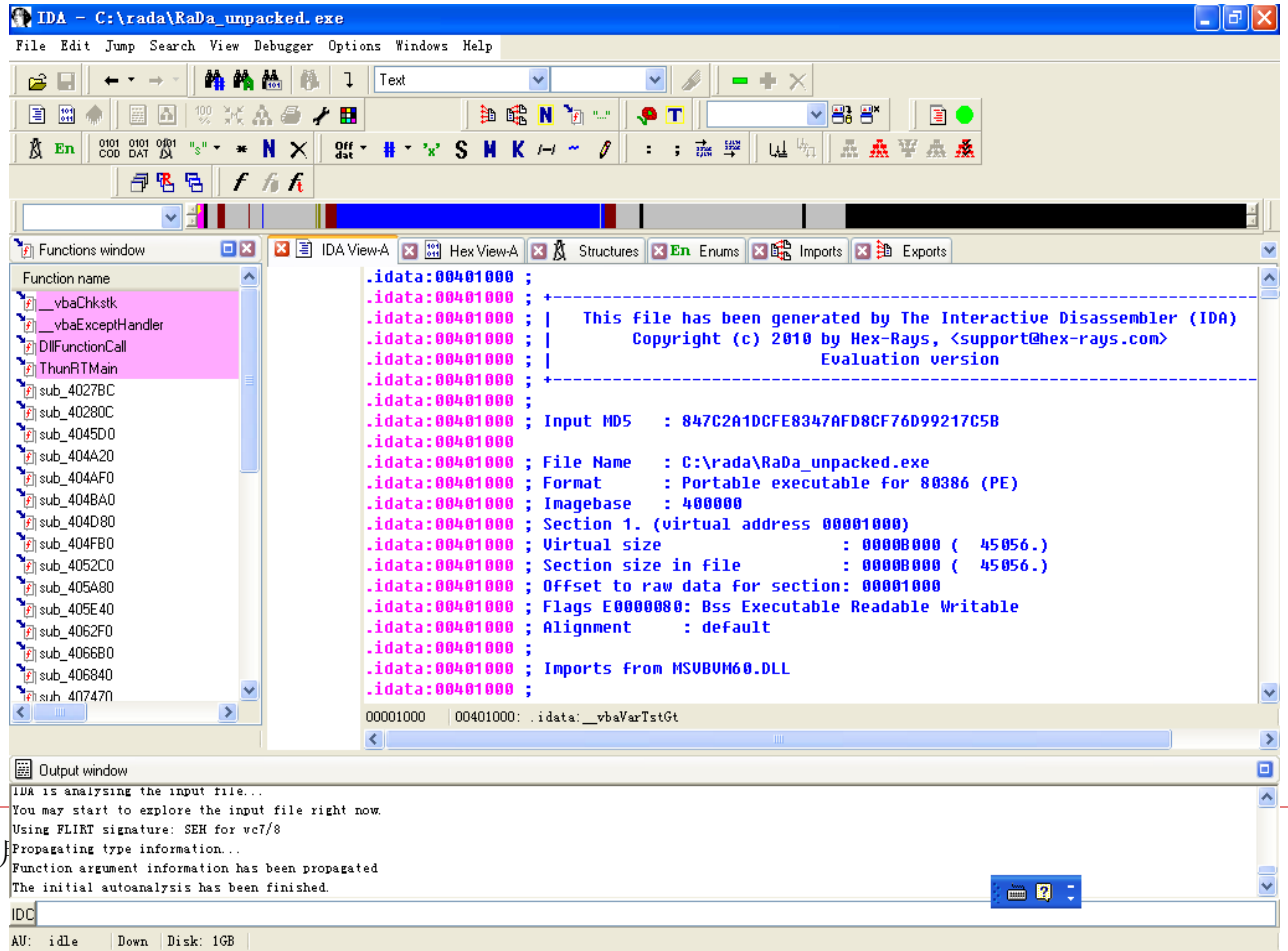
对脱壳前示例恶意代码的strings执行结果	对脱壳后示例恶意代码的strings执行结果
6B@>CEC YMOM@./ RmR].G^ ^@n/ h^ry ... Form1 Module1 ... Command_instal ... ot play/g fun ny securit ch@e usag exit conf Label 237go EVENT _SINK_R KERNEL32.DLL MSVBVM60.DLL **LoadLibraryA** **GetProcAddress** **ExitProcess**	..... **Copyright (C) 2004 Raul Siles & David Perez** --verbose **--visible** **--server** **--commands** ..... --authors Unknown argument: <TITLE>RaDa Current Configuration</TITLE> ----------0123456789012..... ..... Upload file using http And multipart/form-data Copyright (C) 2001 Antonin Foller, PSTRUH Software ..... **SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled = True** ExecQuery MACAddress 00:0C:29: 00:50:56: 00:05:69: Authors: Raul Siles & David Perez, 2004 .....

Binutils-二进制结构分析

- nm
  - 在可执行文件中查找”symbols”重要数据元素
    - 函数名和调用地址；重要变量名和位置；常量等
    - 保存在“符号表”
    - Strip会将其删除
- objdump
  - 从可执行文件中限制不同类型的信息
  - 编译器类型
  - 代码段、数据段位置等
  - 反汇编

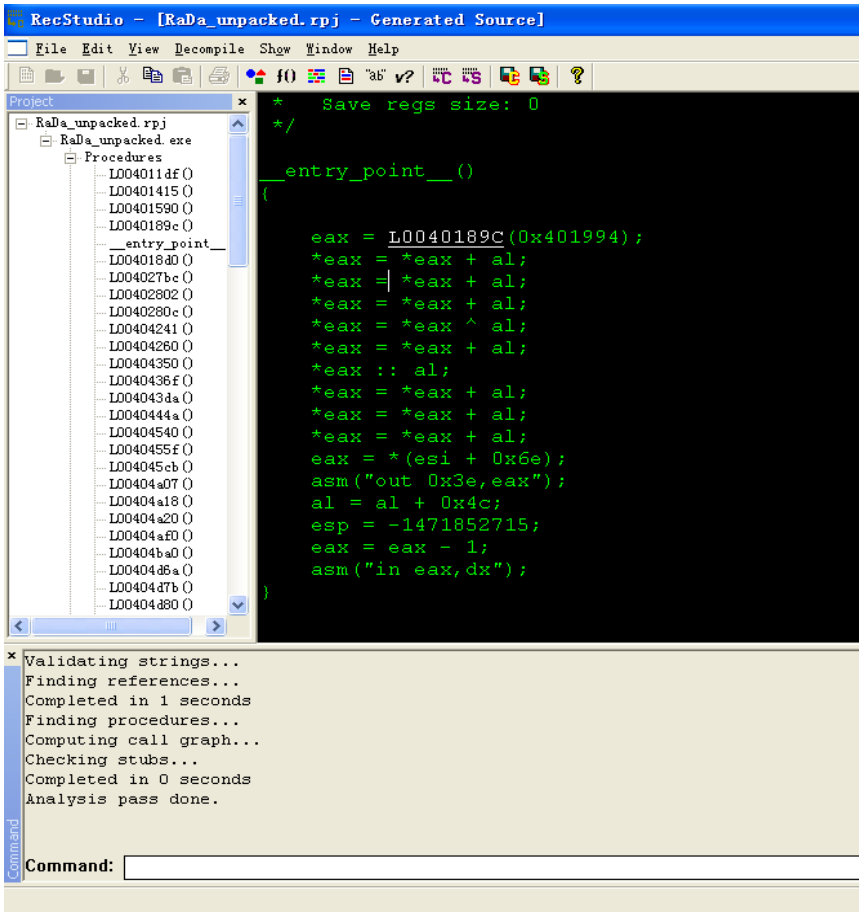
反汇编(Disassemble)

- 反汇编: **IDA Pro, Ollydbg, VC, …**



反编译(Decompiler)

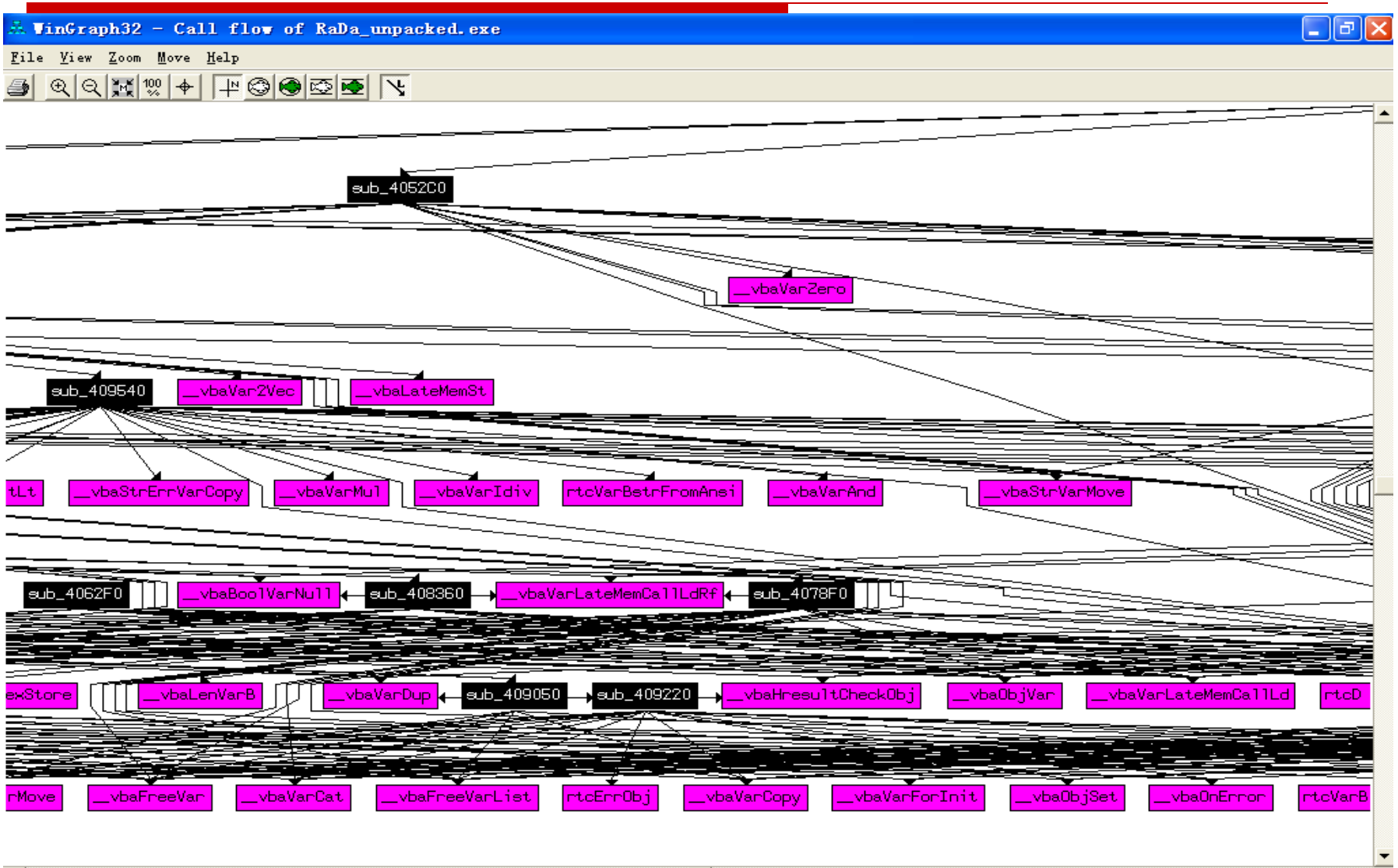
- 反编译: 机器码(汇编语言)→高级编程语言
  - 逆向工程(Reverse Engineering)
  - 再工程(ReEngineering)
- 反编译工具
  - 针对不同高级编程语言
  - Java反编译: JAD, JODE, DAVA, …
  - C/C++反编译: REC, DCC, …
  - Delphi, Flash, …反编译



二进制程序结构和逻辑分析

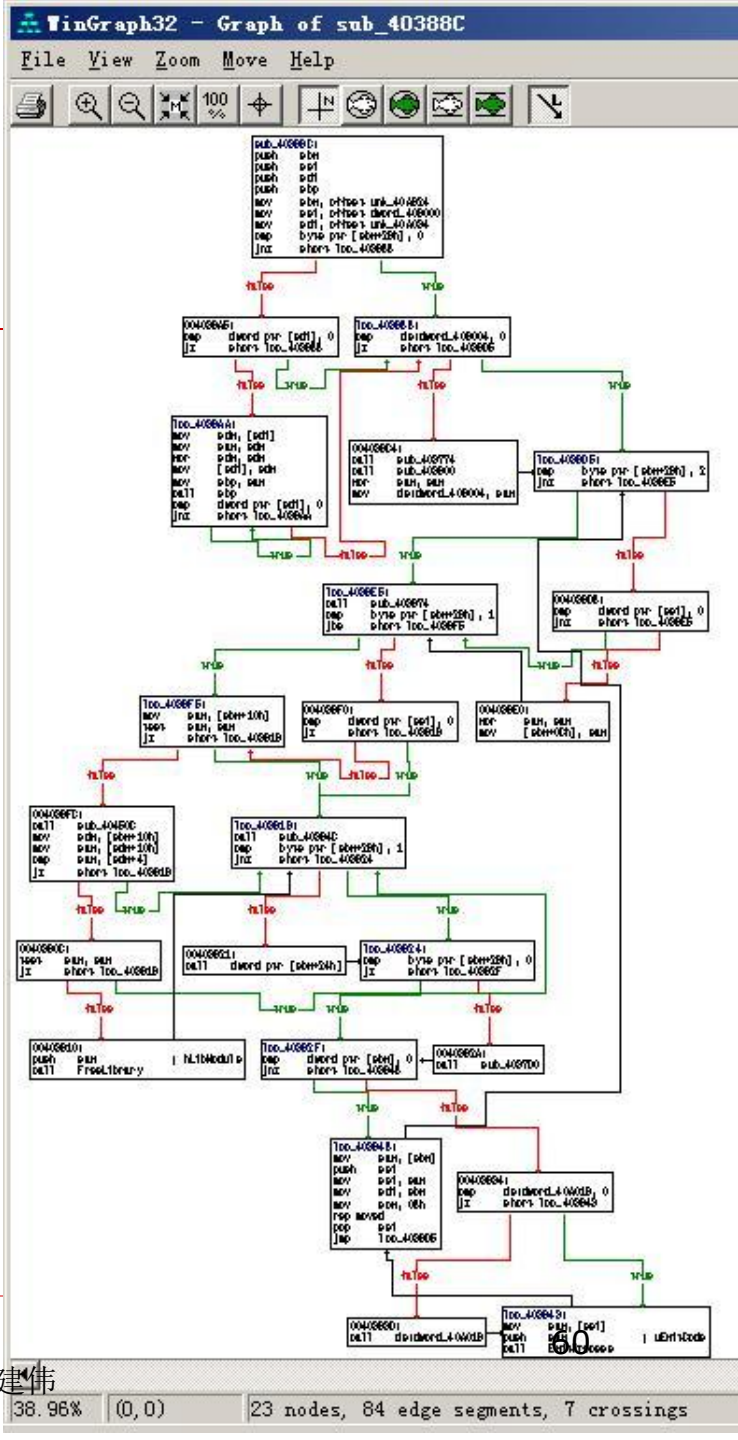
- 程序结构
  - 高层视图：**Call Graph**
    - 用户函数
    - 系统函数
    - 函数调用关系
  - 分析系统函数调用列表可在高层分析二进制程序的行为逻辑
- 程序逻辑
  - 完备视图**CFG (Control Flow Graph)**

Call Graph


WinGraph32 - Call flow of RaDa_unpacked.exe
File View Zoom Move Help
sub_4052C0
__vbaVarZero
sub_409540
__vbaVarZVec
__vbaLateMemSt
tLt
__vbaStrErrVarCopy
__vbaVarMul
__vbaVarIdiv
rtcVarBstrFromAnsi
__vbaVarAnd
__vbaStrVarMove
sub_4062F0
__vbaBoolVarNull
sub_408360
__vbaVarLateMemCallLdRf
sub_4078F0
exStore
__vbaLenVarB
__vbaVarDup
sub_409050
sub_409220
__vbaHresultCheckObj
__vbaObjVar
__vbaVarLateMemCallLd
rtcD
rMove
__vbaFreeVar
__vbaVarCat
__vbaFreeVarList
rtcErrObj
__vbaVarCopy
__vbaVarForInit
__vbaObjSet
__vbaOnError
rtcVarB

CFG

- **CFG:** 程序控制流图
  - 基本块
  - 分支
  - 跳转
  - 循环
- **CFG**完备地反映了一个程序的执行逻辑
  - 完备分析**CFG:** 费时 （右图：仅仅是一个小规模函数**CFG**）
  - 选择性关注



恶意代码混淆机制技术原理

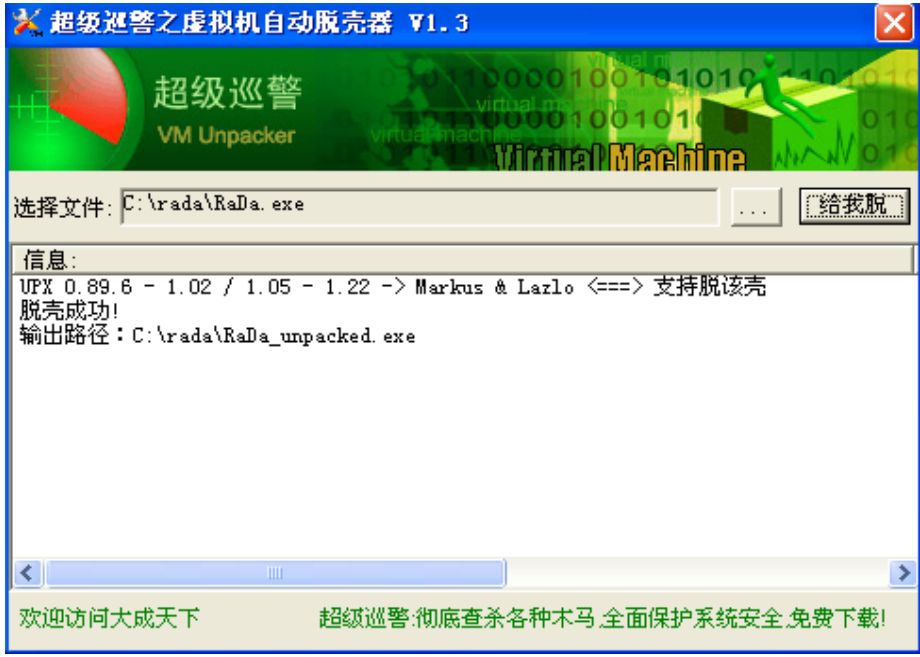
- 加密(encryption)
  - 固定加密/解密器
  - 对解密器进行特征检测
- 多样性(Olgomorphic)
  - 多样化解密器
- 多态(polymorphic)
  - 多态病毒能够通过随机变换解密器从而使得每次感染所生成的病毒实例都具有唯一性。
  - 花指令, 无序的指令变换, 寄存器置换
  - 应对：虚拟机执行脱壳
- 变形(metamorphic)
  - 直接在病毒体上通过各种代码混淆技术
  - 每个感染实例都具有不同的形式

恶意代码加壳

- 恶意代码加壳
  - 常用壳: **UPX, PEPack, ASPack, PECompact, …**
  - 壳的分类：压缩壳、加密壳、伪装壳、…
  - 多重加壳：嵌套使用各类壳
  - 终极免杀技术分类讲解
- 加壳其他应用场景
  - 减小应用程序大小规模
  - 保护应用程序版权，加大破解难度: 软件狗加密

脱壳

- 常见壳的自动脱壳工具
  - **UPX: upx -d**
  - **PEPack: UnPEPack**
  - **ASPack压缩壳: ASPack unpacker**
  - **…**
  - 推荐：超级巡警脱壳器 **(VMUnpacker)**
- 手工脱壳
  - 关键步骤: 寻找程序入口点，**dump**出程序，修复**PE**文件（导入、导出表等）
  - 看雪学院: *www.pediy.com*



内容

1. 恶意代码基础知识
2. 恶意代码分析技术
   - 课题实践：恶意代码静态分析
   - 课堂实践：分析Crackme程序
3. 课外作业：分析一个自制恶意代码样本

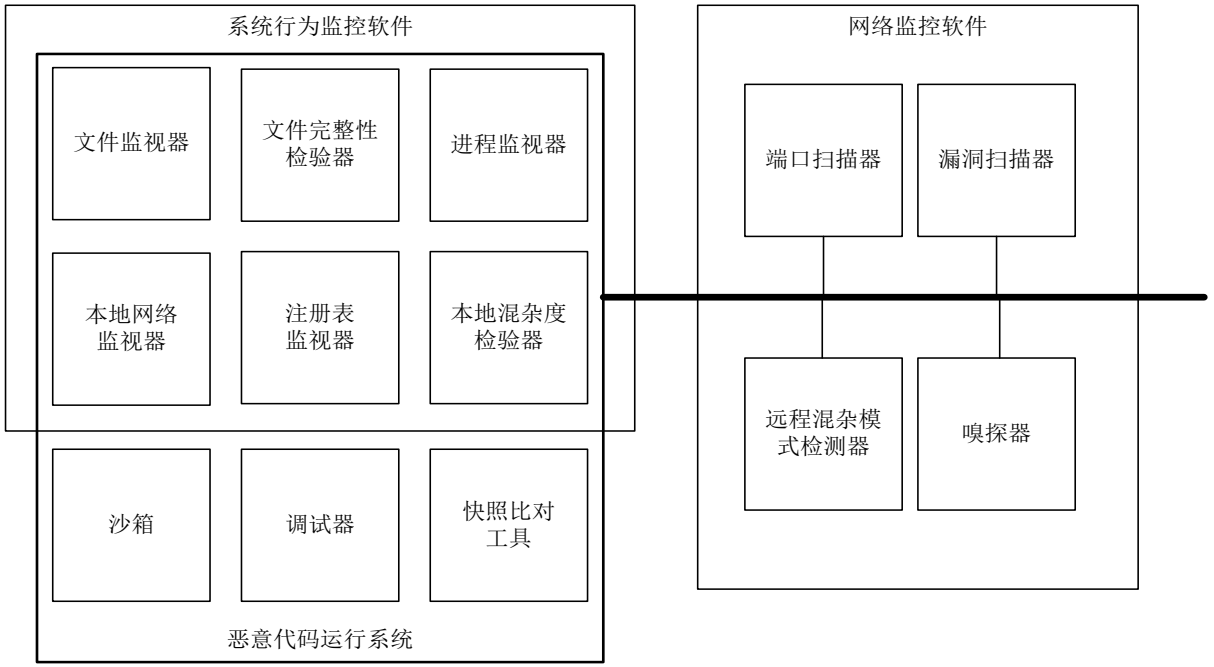
课堂实践 – 恶意代码静态分析

- 实践任务：对提供的**rada**恶意代码样本（**U**盘或**FTP/materials/course9**下载），在**WinXP_Attacker**虚拟机中进行文件类型识别，脱壳与字符串提取，以获得**rada**恶意代码的编写作者。
- **(1)** 使用文件格式和类型识别工具**(file, PEid)**，给出**rada**恶意代码样本的文件格式、运行平台和加壳工具；
- **(2)** 使用超级巡警脱壳机等脱壳软件，对**rada**恶意代码样本进行脱壳处理；
- **(3)** 使用字符串提取工具**(strings**、**IDA Pro)**，对脱壳后的**rada**恶意代码样本进行分析，从中发现**rada**恶意代码的编写作者是谁？

恶意代码动态分析方法列表

分析方法	目的	使用工具	难度
快照比对	获取恶意代码行为结果	FileSnap,RegSnap,完美卸载	低
动态行为监控(API Hooking)	实时监控恶意代码动态行为轨迹	Filemon, Regmon, Process Explorer, lsof…	中
网络监控	分析恶意代码网络监听端口及发起网络会话	Fport, lsof, TDImon, ifconfig, tcpdump, …	中
沙盒(sandbox)	在受控环境下进行完整的恶意代码动态行为监控与分析	Norman Sandbox, CWSandbox, FVM Sandbox, …	中高
动态跟踪调试	单步调试恶意代码程序，理解程序结构和逻辑	Ollydbg,IDAPro, gdb, SoftICE,systrace, …	高

动态分析中的监视与控制

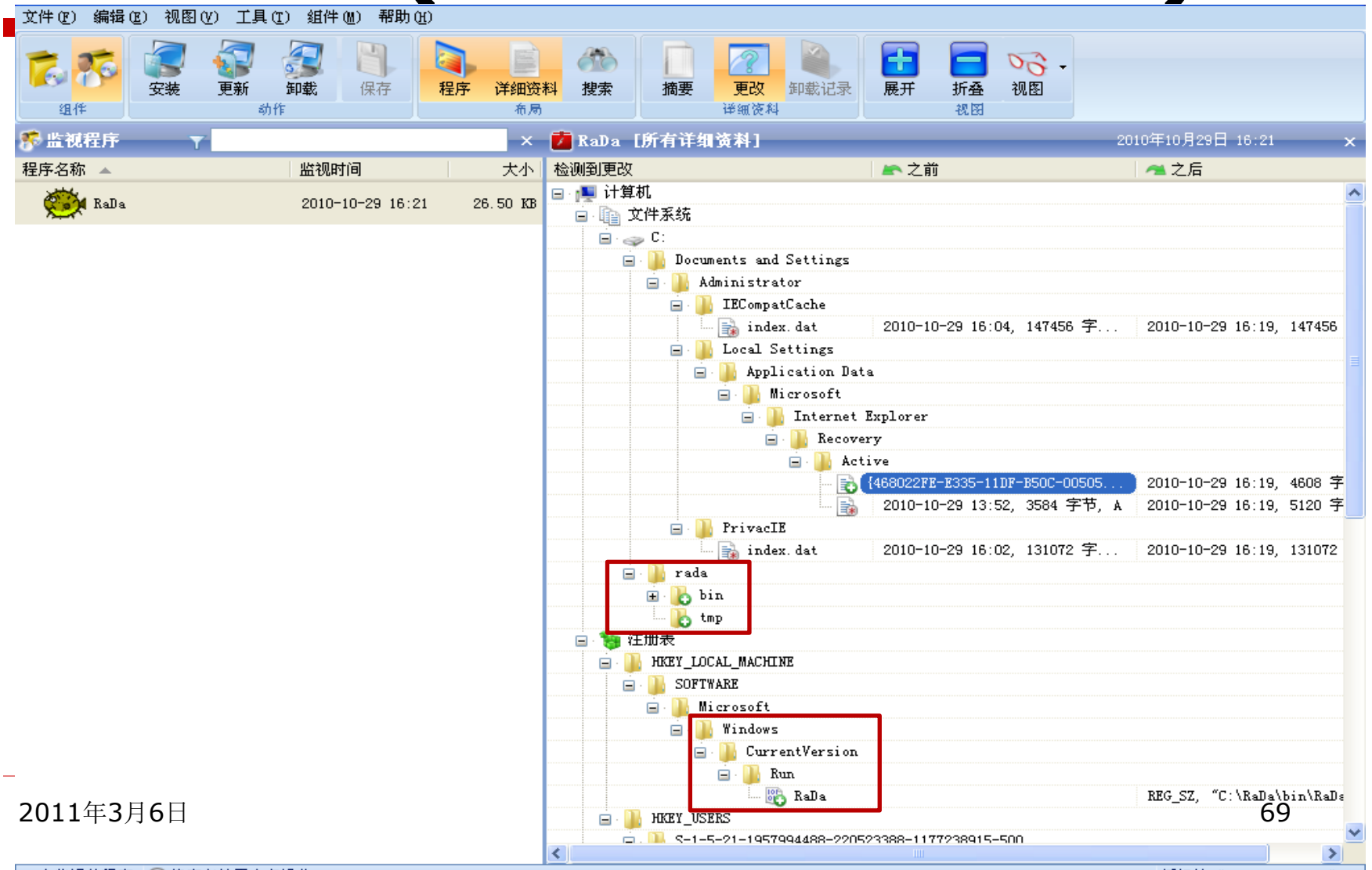


- 行为监视
  - 一系列监控软件来控制和观察恶意代码的运行情况
- 网络控制
  - 最安全的控制策略：与业务网络和互联网保持物理隔离

基于快照比对的方法

- 快照比对方法
  - **1.** 对“干净”资源列表做快照
  - **2.** 运行恶意代码（提供较充分的运行时间 **5**分钟）
  - **3.** 对恶意代码运行后的“脏”资源列表做快照
  - **4.** 对比“干净”和“脏”快照，获取恶意代码行为结果
    - 资源名称列表中的差异：发现新建、删除的行为结果
    - 资源内容的差异：完整性校验，发现修改的行为结果
- 进行快照比对的工具
  - **RegSnap**
  - 完美卸载
  - **HoneyBow之MwFetcher**
- 快照比对方法的弱点：无法分析中间行为，粗粒度

完美卸载 (Total Uninstall)

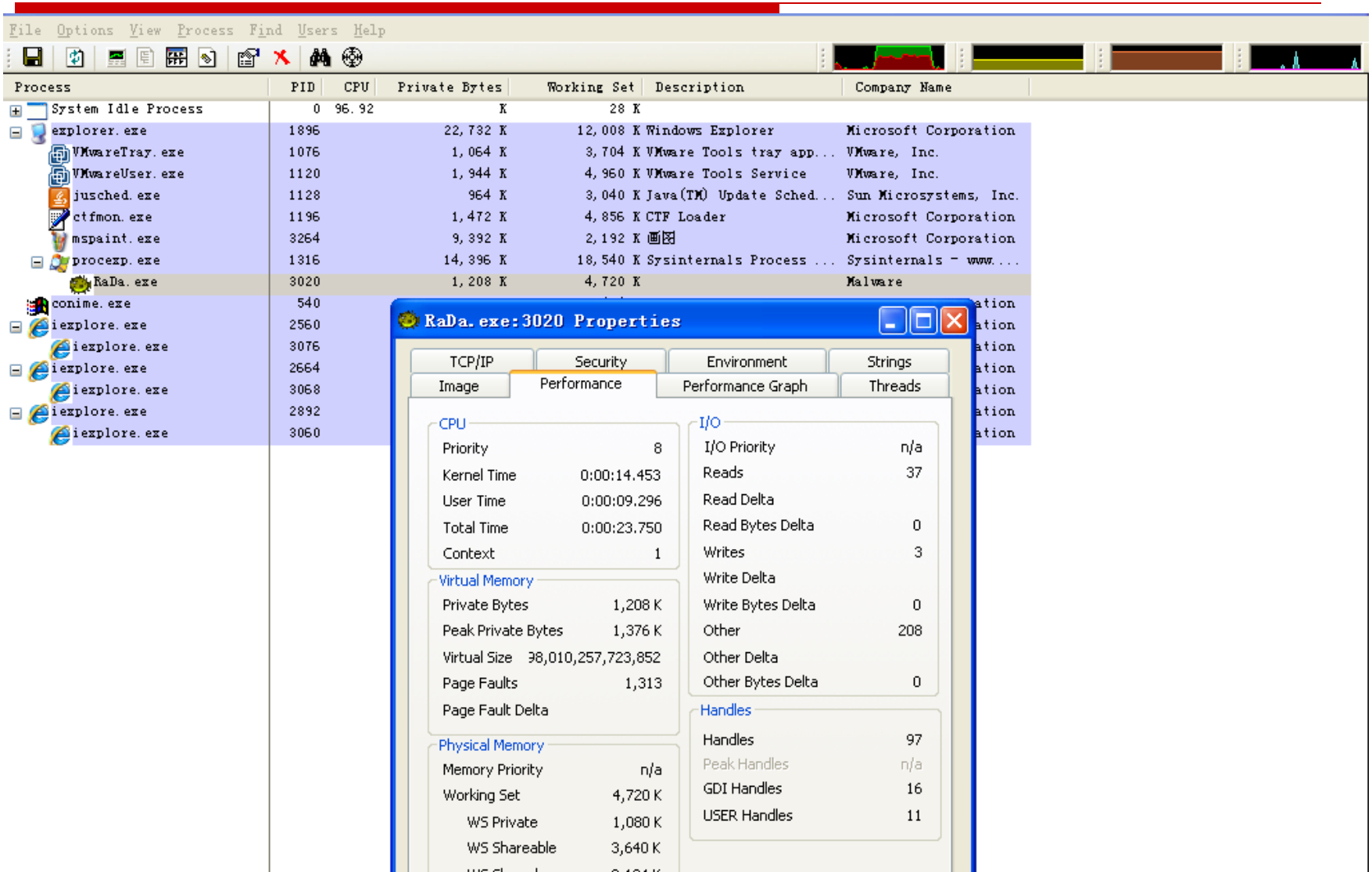


动态行为监控方法

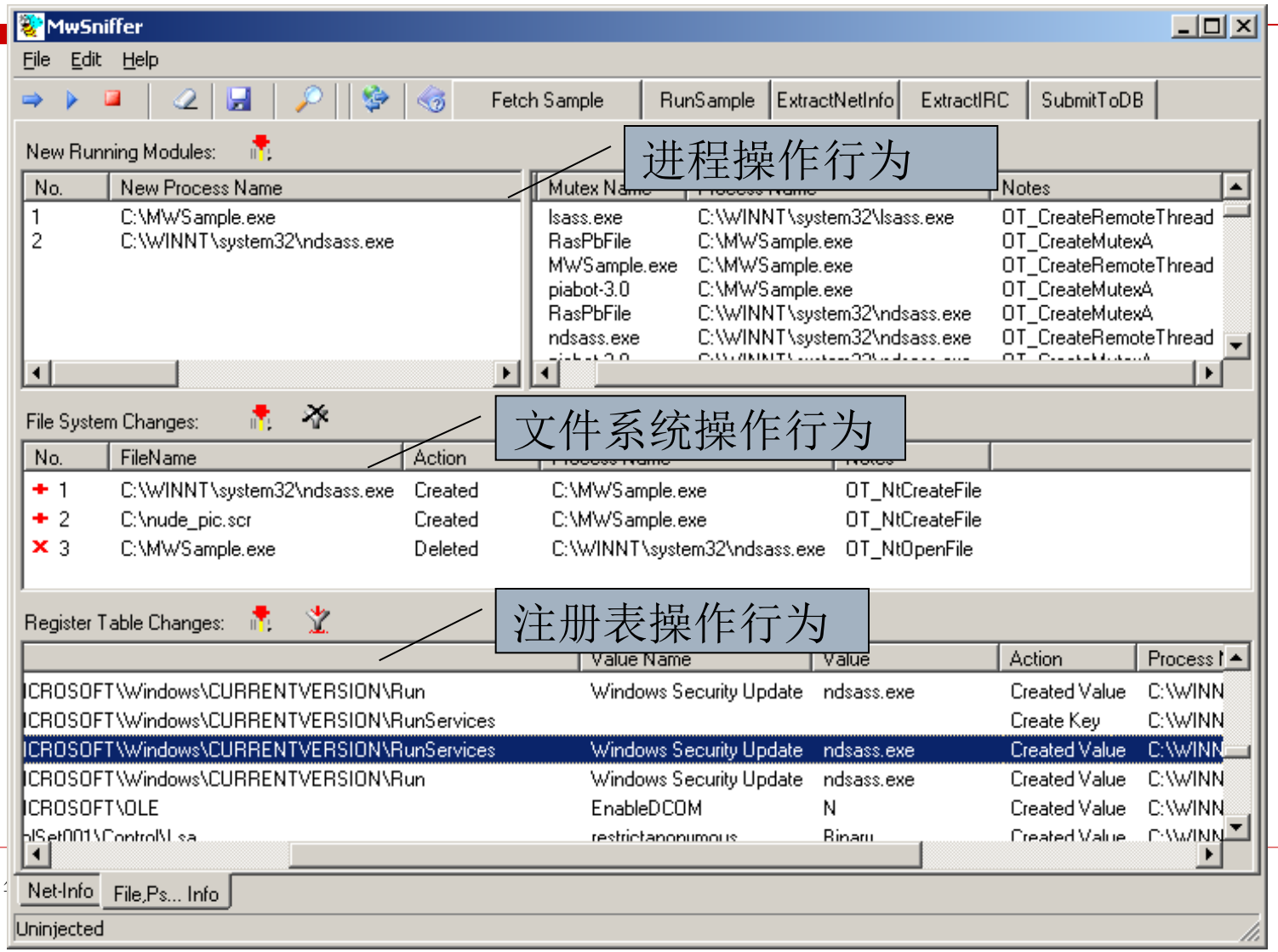
- 行为监控技术
  - **Notification机制**
    - **Win32/Linux系统本身提供的行为通知机制**
  - **API Hooking技术**
    - 对系统调用或**API**调用进行劫持，监控行为

- 系统行为动态监控工具
  - 文件行为监控: **Filemon**
  - 进程行为监控: **Process Explorer, lsof**
  - 注册表监控: **Regmon**
  - 本地网络栈行为监控软件: **lsof, TDImon, promiscdetect**
  - 完整的动态行为监控: **MwSniffer, Sebek, …**

Process Explorer



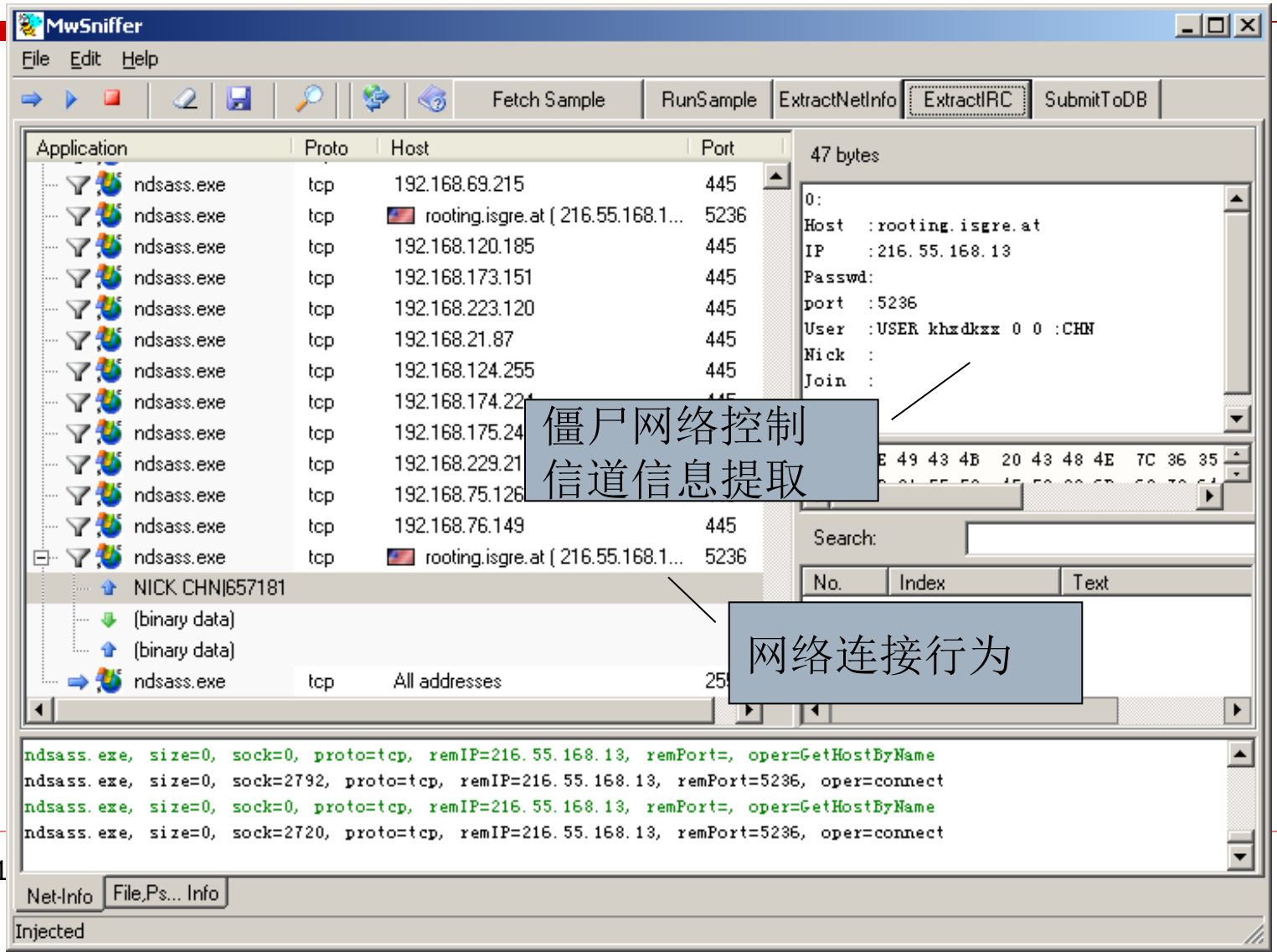
MwSniffer – 系统行为监控



网络监控

- 恶意代码开放的本地端口
  - 本机检查: **fport, TCPView(win32), lsof(linux)**
  - 网络检查: **nmap**
- 恶意代码发起的网络连接
  - 捕获: **tcpdump, wireshark (ethereal), TDImon**
  - 分析: **argus, wireshark, snort**
  - 重现: **tcpreplay**
- 控制恶意代码网络流
  - **IPTables, Snort_inline, …**
- 恶意代码流行攻击方式-**ARP欺骗**
  - **ARP防火墙 (360, AntiARP)**

MwSniffer – 网络行为监控

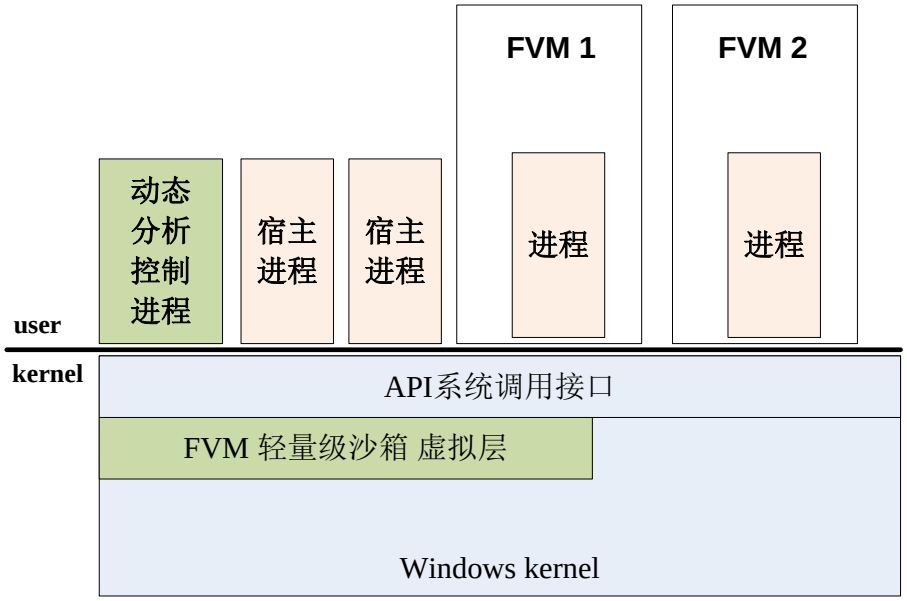


“沙盒”技术 - Sandbox

- 沙盒技术
  - 用于安全运行程序的安全环境.
  - 经常被用于执行和分析非可信的代码.
- 用于防御的沙盒技术
  - **Java Applets, jail(virtual hosting),** 虚拟机, 权能
- 用于恶意代码分析的沙盒技术实例
  - **Norman Sandbox(模拟器):** **http://www.norman.com/microsites/nsic/**
  - **CWSandbox(Native OS): http://www.cwsandbox.org/**
  - **FVM Sandbox(Native OS)** 轻量级并行化沙箱
    - **developer:** 宋程昱
    - 并行化->标配服务器**64**路并行

FVM Sandbox-轻量级并行化沙箱

- 轻量级沙箱技术
  - 基于Windows操作系统（提供样本运行环境）
  - 使用名字空间重定向技术实现资源隔离、可并发化、沙箱的透明化及分析环境的可恢复性
  - 使用COW (Copy on Write)技术保证沙箱的轻量化
  - 基于内核层的SSDT Hooking技术实现进程动态行为的分析和控制



动态调试技术

- 动态调试
  - 程序运行时刻
  - 它的执行过程进行调试(debugging)
  - 二进制调试
- 动态调试技术
  - 断点
  - 单步模式
  - 寄存器和内存状态查看与修改
- 动态调试工具
  - **Windows：Ollydbg、windbg、IDA Pro、SoftICE**
  - **Linux：gdb、systrace、ElfShell**

内容

1. 恶意代码基础知识
2. 恶意代码分析技术
   - 课题实践：恶意代码静态分析
   - 课堂实践：分析Crackme程序
3. 课外作业：分析一个自制恶意代码样本

课堂实践 – 分析Crackme程序

- 实践挑战：在**WinXP_Attacker**虚拟机中使用**IDA Pro**静态或动态分析**crackme1.exe**，寻找特定的输入，使其能够输出成功信息。
- 步骤
  - **1.** 动态运行程序，提供不同输入，观察现象
  - **2.** 采用静态方法分析程序，理解程序逻辑
    - **Strings**
    - **IDA Pro**
  - **3.** 重新运行程序，提供正确输入，使其能够输出成功信息。

内容

1. 恶意代码基础知识
2. 恶意代码分析技术
   - 课题实践：恶意代码静态分析
   - 课堂实践：分析Crackme程序
3. 课外作业：分析一个自制恶意代码样本

课外实践作业9: 分析一个自制恶意代码样本(团队) – 20分

- 作业内容:
  - 本次实践作业的任务是分析一个自制的恶意代码样本，以提高对恶意代码逆向工程分析技术的认识，并提高逆向工程分析的方法、工具和技术。
  - 关于这个二进制文件，我们只能告诉你创建它的目的是为了提高安全业界对恶意代码样本的认识，并指出为对抗现在的恶意代码威胁发展更多防御技术的必要性。现在你作为一名安全事件处理者的任务（如果你接受的话）就是深入分析这个二进制文件，并获得尽可能多的信息，包括它是如何工作的，它的目的以及具有的能力，最为重要的，请展示你获取所有信息所采取的恶意代码分析技术。
- 待分析二进制文件位置:
  - **FTP/exercises/exercise9.zip, MD5 = a75de27ee59ab60e148efe7feee5dd3f**
  - *警告* 这个二进制文件是一个恶意代码，因此你必须采用一些预防措施来保证业务系统不被感染，建议在一个封闭受控的系统或网络中处理这个未知的实验品。

作业问题

- **1.** 提供对这个二进制文件的摘要，包括可以帮助识别同一样本的基本信息。 **(1)**
- **2.** 找出并解释这个二进制文件的目的。 **(2)**
- **3.** 识别并说明这个二进制文件所具有的不同特性。 **(2)**
- **4.** 识别并解释这个二进制文件中所采用的防止被分析或逆向工程的技术。 **(2)**
- **5.** 对这个恶意代码样本进行分类（病毒、蠕虫等），并给出你的理由。 **(2)**
- **6.** 给出过去已有的具有相似功能的其他工具。 **(1)**
- 奖励问题： **bonus 2**分
- **7.** 可能调查出这个二进制文件的开发作者吗？如果可以，在什么样的环境和什么样的限定条件下？
- **Deadline: 12月22日下午17:00**

Thanks

诸葛建伟

zhugejw@gmail.com