恶意软件发现与分析

第2章 恶意软件发现

主要内容

- 2.1 恶意软件基础知识
- 2.2 恶意软件发现技术

2.1 恶意软件基础知识

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、寄生在文件或启动扇区的指令流
  - 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**…**

大众性恶意代码与针对性恶意代码

- 大众性恶意代码
  - 旨在感染尽可能多的机器
  - 最常见的类型
- 针对性恶意代码
  - 针对一个特定的目标
  - 很难检测、预防和消除
  - 需高级的分析技术
  - 例如： Stuxnet

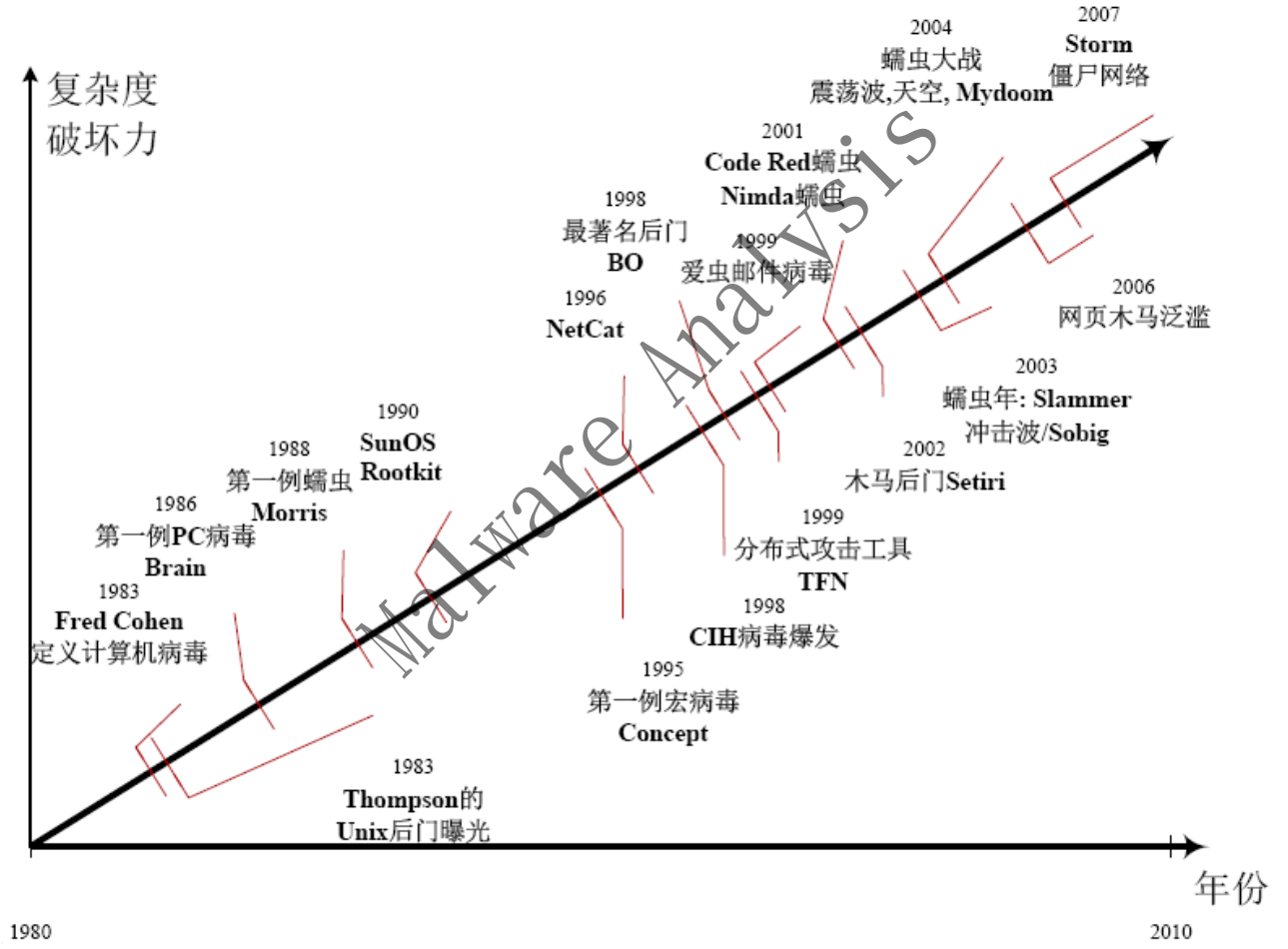
恶意代码的发展史

- **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

- **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**…**
  - **2015-2016**：蓝莲花APT、白象APT
  - **2017**：深蓝之蓝

恶意代码类型

- 后门
  - 允许攻击者控制该系统
- 僵尸网络
  - 所有被感染的计算机从同一台控制命令服务器接收命令
- 下载器
  - 只用来下载其他恶意代码的恶意代码
  - 通常在攻击者第一次获得访问权限时使用

恶意代码类型

- 间谍软件
  - 嗅探、键盘记录、密码哈希采集
- 启动器
  - 用于启动其他恶意程序的恶意代码
  - 通常使用非传统技术确保其隐蔽性或以更高的权限访问系统
- Rootkit
  - 用来隐藏其他恶意代码的恶意代码
  - 通常搭配一个后门

恶意代码类型

- 勒索软件
  - 吓唬被感染用户，让他们购买某些东西

Fake FBI warning tricks man into surrendering himself for possession of child porn

29 Jul, 2013 | by Nishtha Kanal | Like 3 +1 0 Tweet 3 Share

Here's a weird one. We've heard of viruses and malware bringing harm to computers but in a rare instance, a "ransomware" has brought a positive outcome. A man in the US turned himself in to the police after a pop-up caused by a ransomware informed him that child porn had been identified on his machine.

Jay Matthew Riley, a 21-year-old from Virginia was browsing the Internet, when a pop-up containing an "FBI warning" informed him that it had detected child pornography on his machine. The message went on to tell Riley to pay up a fine online or face the consequences.

恶意代码类型

- 发送垃圾邮件的恶意代码
  - 攻击者将被感染的计算机租给垃圾信息散布者
- 蠕虫或计算机病毒
  - 可以自我复制和感染其他计算机的恶意代码

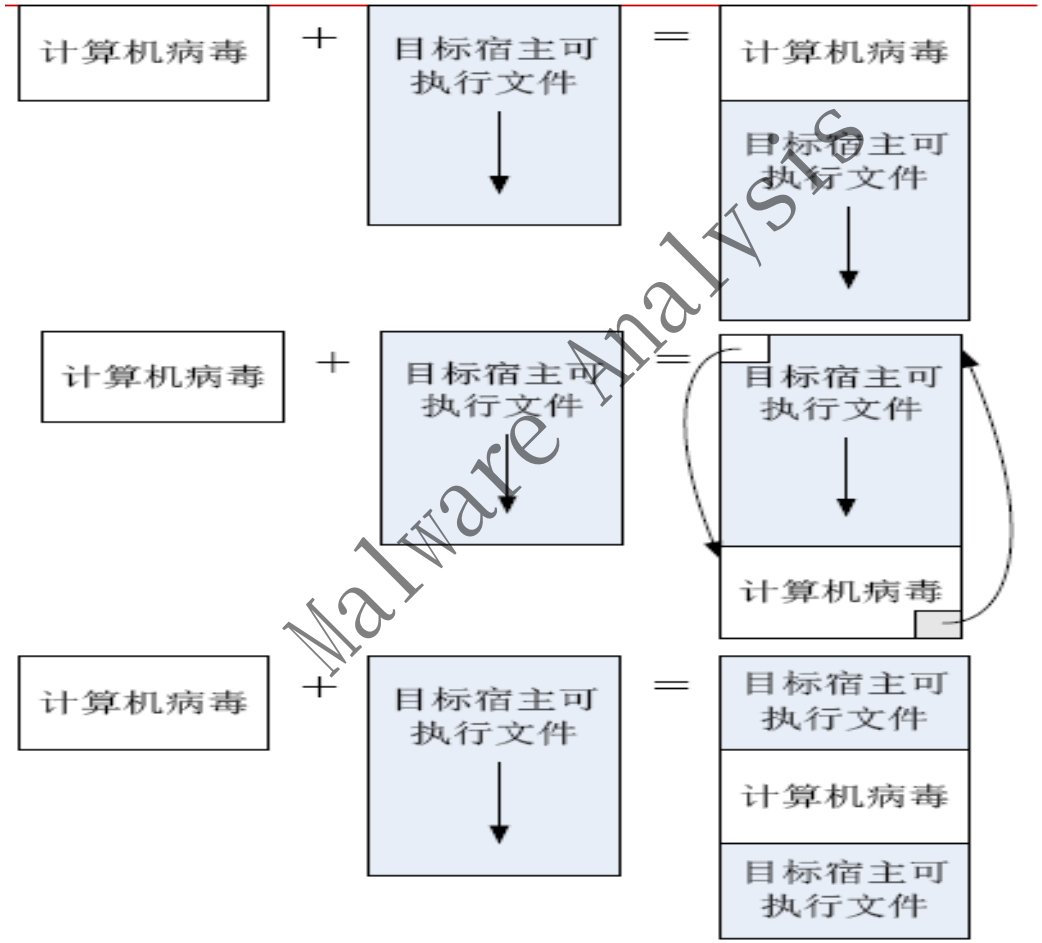
计算机病毒

- **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

计算机病毒的感染机制



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
- 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VS.计算机病毒

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。

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

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

Rootkit和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

Rootkit和木马后门之间的位置对比

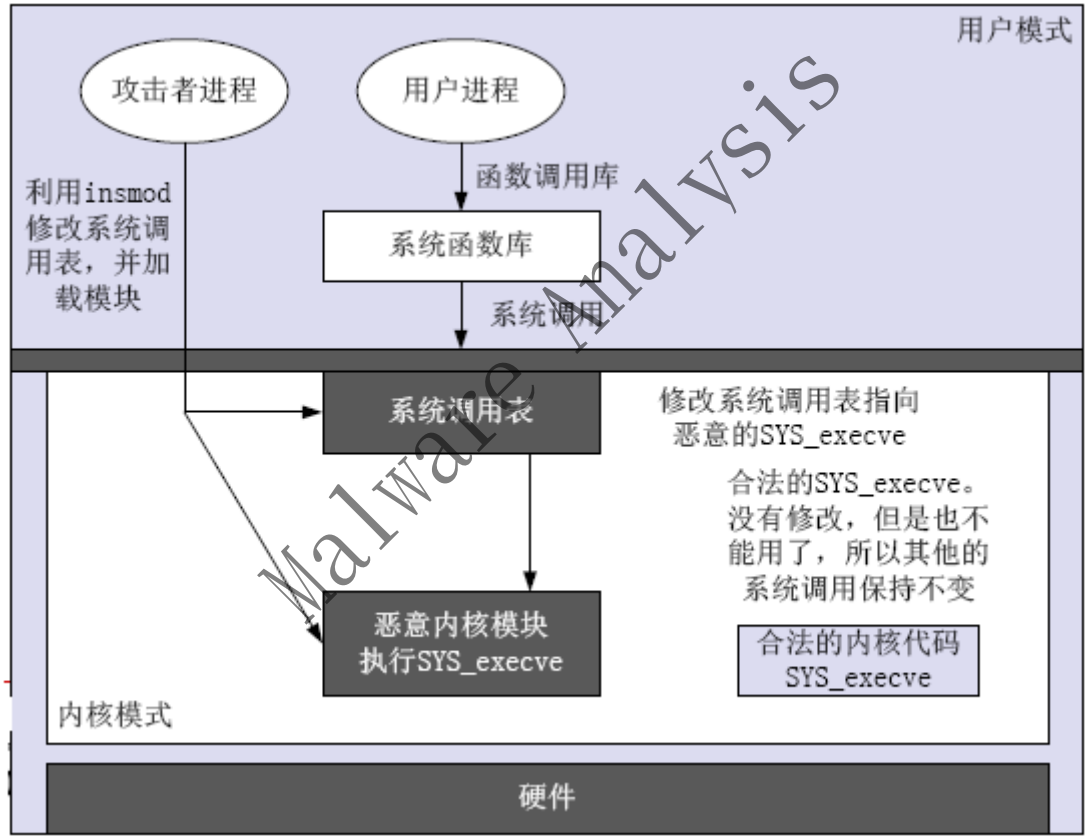
用户模式Rootkit

- 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**
  - **Linux：UML、KML**
  - **Win32：VM Rootkit**

内核模式Rootkit



2.1 恶意软件发现技术

恶意代码发现技术

- 恶意代码发现的目标
  - 在主机和网络环境中发现已知或者未知的恶意代码
- 常见的恶意代码发现技术
  - 文件扫描和主机行为检测
  - 通信行为检测
  - 相似性检测
- 挑战：
  - “未知的恶意代码”是当前面临的挑战

恶意代码发现分析的方法

- **静态分析方法**
  - 在不运行恶意代码的情况下，利用分析工具对恶意代码的静态特征和功能模块进行分析的方法
- **动态分析方法**
  - 通过监视恶意代码运行过程从而了解恶意代码功能

静态分析方法

- 基于代码特征的分析方法
  - 分析过程中，不考虑恶意代码的指令意义，而是分析指令的统计特性、代码的结构特性等
- 基于代码语义的分析方法
  - 要求考虑构成恶意代码的指令的含义，通过理解指令语义建立恶意代码的流程图和功能框图，进一步分析恶意代码的功能结构

基于代码特征的分析方法

- 常用于对执行程序类型的恶意代码进行分析
  - C语言编写的程序中存在一条语句CreateMuetex(NULL,NULL,“MYTEST”);那么在生成的PE文件中会存在一个静态数据“MYTEST”，通过分析PE结构可以从静态数据节中提取静态数据
  - 用C语言编写的恶意代码中使用下面的语句URLDownloadToFile(0,"http://www.microsoft.com/a.exe","c:\\a.exe",0,0)从网站下载可执行程序到C盘根目录，这个动作很有可能是进行恶意代码升级

基于代码语义的分析方法

- 基于代码语义的分析过程，首先使用反汇编工具对恶意代码执行体进行反汇编，然后通过理解恶意代码的反汇编程序了解恶意代码的功能。从理论上讲通过这种方法可以得到恶意代码所有功能特征
- 但是，目前基于语义的恶意代码分析方法主要还是依靠人工来完成，人工分析的过程需要花费分析人员的大量时间，对分析人员本身的要求也很高

动态分析方法

- 外部观察
  - 利用系统监视工具观察恶意代码运行过程时系统环境的变化，通过分析这些变化判断恶意代码的功能
- 跟踪调试
  - 通过跟踪恶意代码执行过程使用的系统函数和指令特征分析恶意代码功能的技术

外部观察

- 恶意代码作为一段程序在运行过程中通常会对系统造成一定的影响，有些恶意代码为了保证自己的自启动功能和进程隐藏的功能，通常会修改系统注册表和系统文件，或者会修改系统配置
  - 1、通过网络进行传播、繁殖和拒绝服务攻击等破坏活动
  - 2、通过网络进行诈骗等犯罪活动
  - 3、通过网络将搜集到的机密信息传递给恶意代码的控制者
  - 4、在本地开启一些端口、服务等后门等待恶意代码控制者对受害主机的控制访问

跟踪调试

- 在实际分析过程中，跟踪调试可以有两种方法
  - 1.单步跟踪恶意代码执行过程，监视恶意代码的每一个执行步骤，在分析过程中也可以在适当的时候执行恶意代码的一个片断，这种分析方法可以全面监视恶意代码的执行过程，但是分析过程相当耗时
  - 2.利用系统hook技术监视恶意代码执行过程中的系统调用和API使用状态来分析恶意代码的功能，这种方法经常用于恶意代码检测

恶意代码发现分析方法比较

分析内容	代码特征分析法	代码语义分析法	外部观察法	跟踪调试法
隐藏功能		能		能
加密功能	能	能		能
触发功能		能		能
自启动功能		能	能	能
自主攻击和繁殖功能		能	部分	能
破坏功能		能	部分	能
对分析人员的依赖程度	低	较高	低	高
对分析环境的破坏	否	否	大	可控

恶意代码与良性代码发现分析的比较

- **恶意代码发现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

文件扫描和主机行为检测

文件扫描和主机行为检测

- **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。

静态扫描和检测方法

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

- 使用反病毒软件进行检测
  - 卡巴斯基、赛门铁克等
  - 奇虎360、瑞星、金山、江民等
- **VirusTotal**
  - “世界病毒扫描网”
  - **https：//www.virustotal.com/**
- 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、百度**）

VirusTotal对示例恶意代码的识别

virustotal

SHA256:	ccf51ecd278899afd81fbe3968ac754a964efe4600045458abe2130b63891e89
File name:	F:\OldE\课题项目\海南省自然科学基金\附件1：《海南省科技成果验收（结题）材料统一格式》等材料\VirusInfection\PEVirus_3.exe
Detection ratio:	6 / 41
Analysis date:	2012-05-15 07:44:23 UTC (1 分钟 ago)

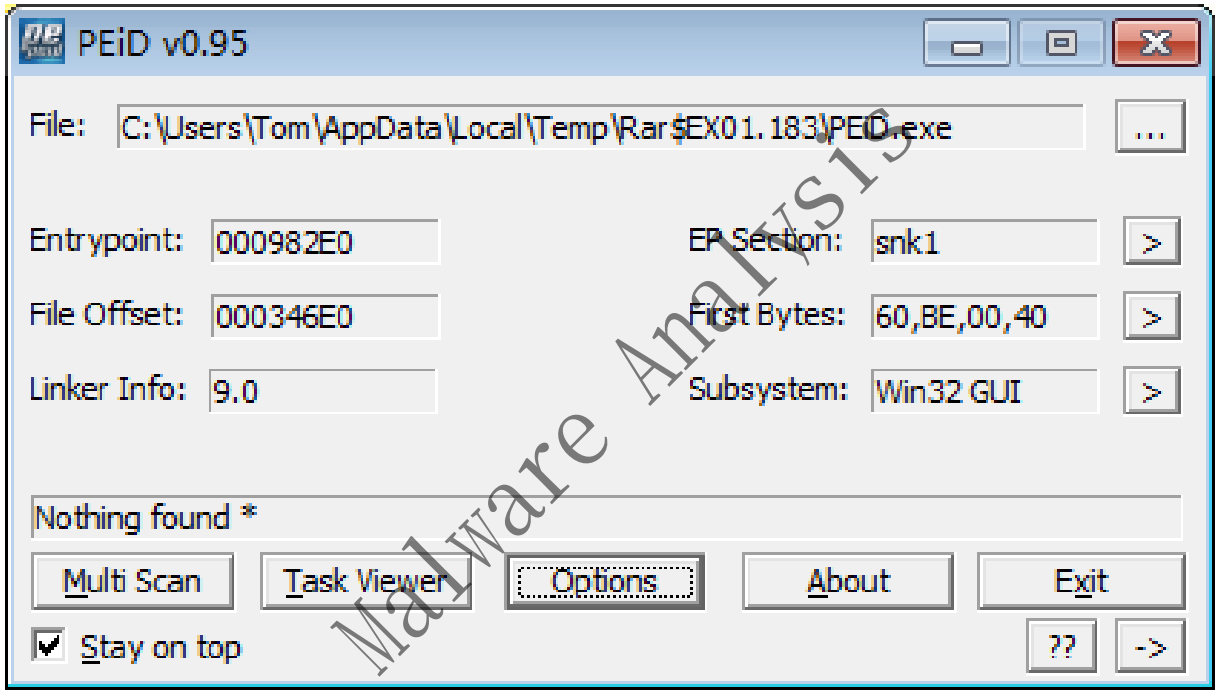
More details

Antivirus	Result	Update
AhnLab-V3	-	20120514
AntiVir	-	20120514
Antiy-AVL	-	20120515
Avast	Win32:Tufik	20120514
AVG	Win32/Tufik.A	20120515

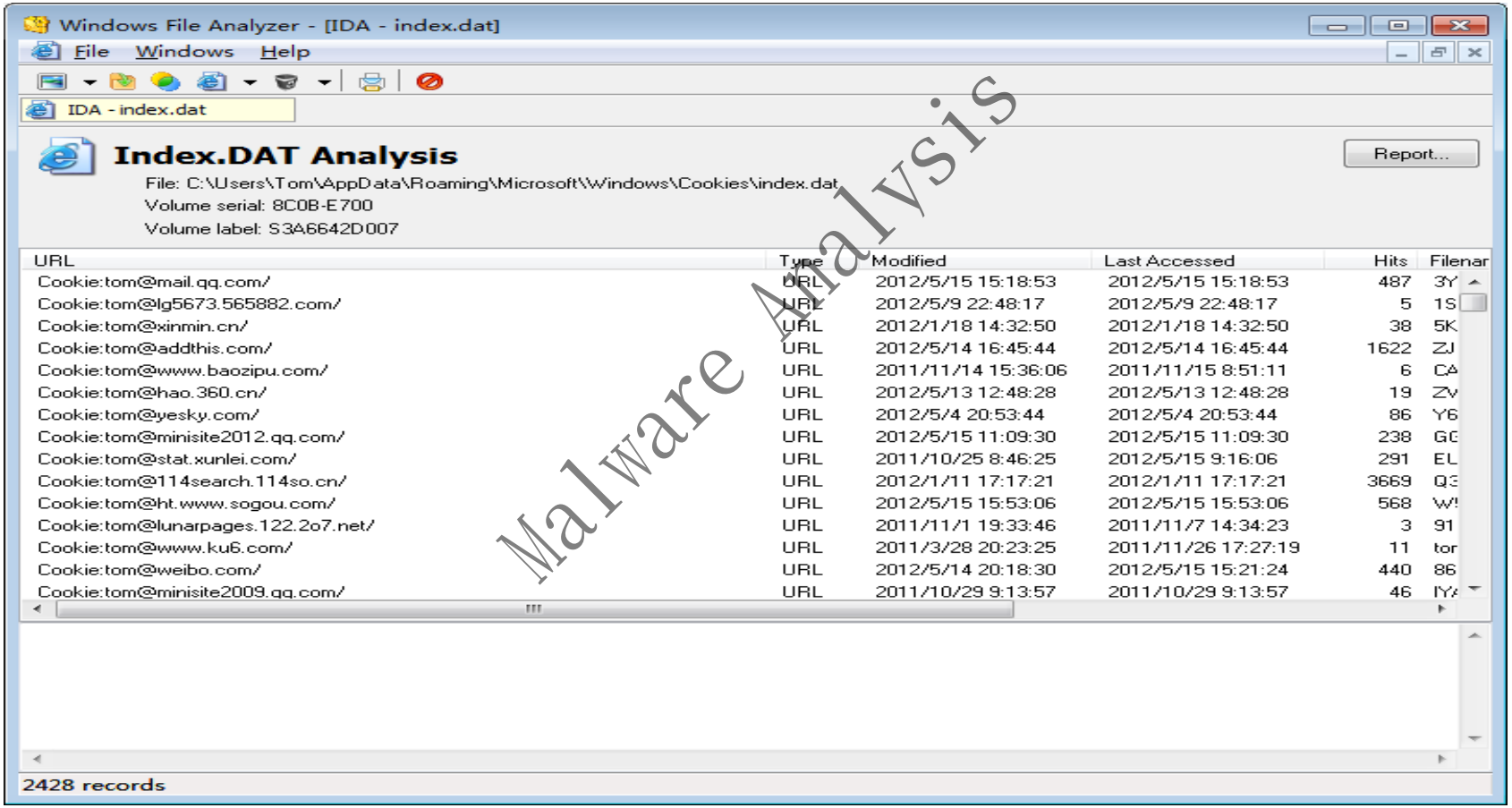
文件格式确定

- **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

PEiD


PEiD v0.95
File:
C:\Users\Tom\AppData\Local\Temp\Rar$EX01.183\PEiD.exe
...
Entrypoint:
000982E0
EP Section:
snk1
>
File Offset:
000346E0
First Bytes:
60,BE,00,40
>
Linker Info:
9.0
Subsystem:
Win32 GUI
>
Nothing found *
Multi Scan
Task Viewer
Options
About
Exit
Stay on top
??
->
Malware Analysis

Windows File Analyzer

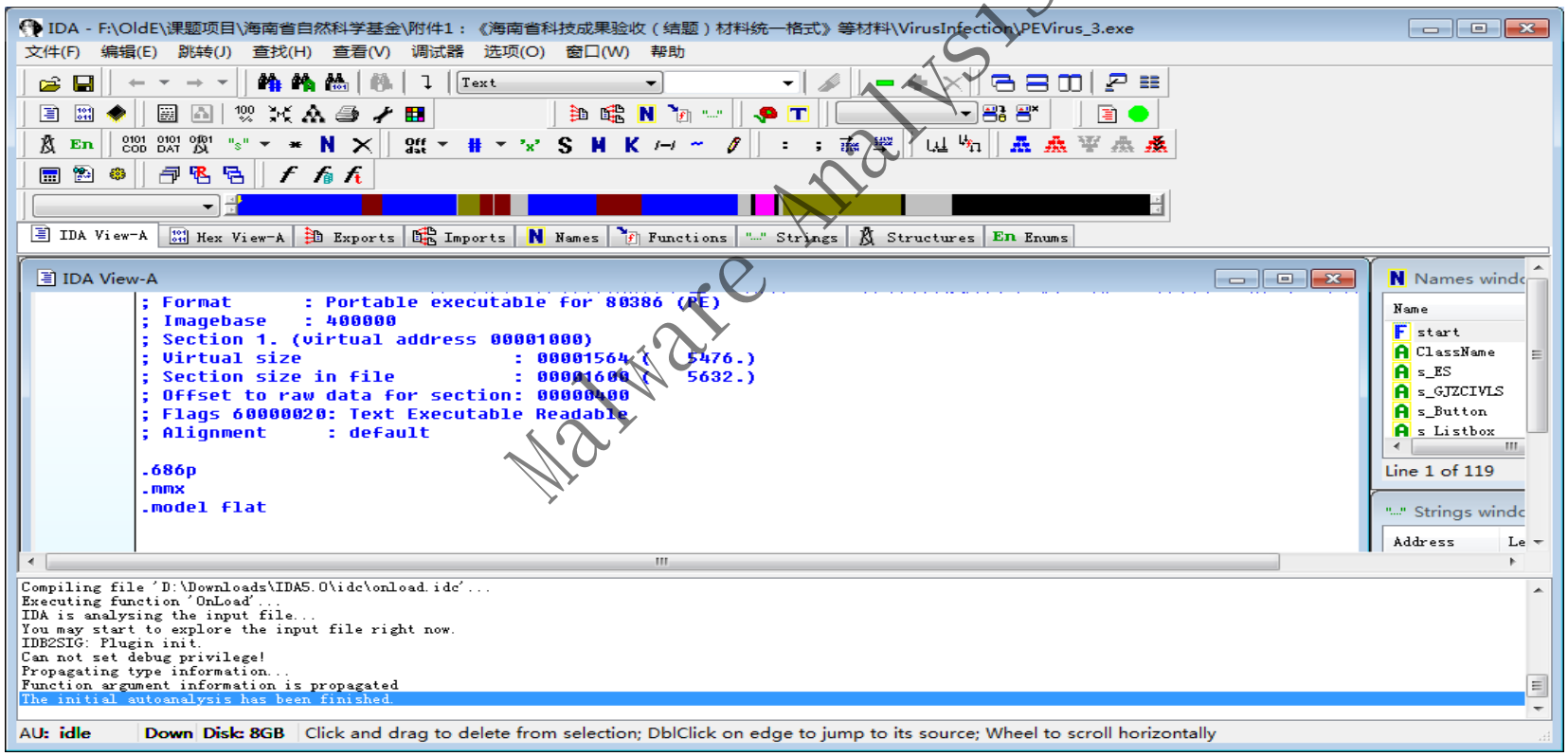


Strings命令-查看可打印字符串

- **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**…**

反汇编（Disassemble）

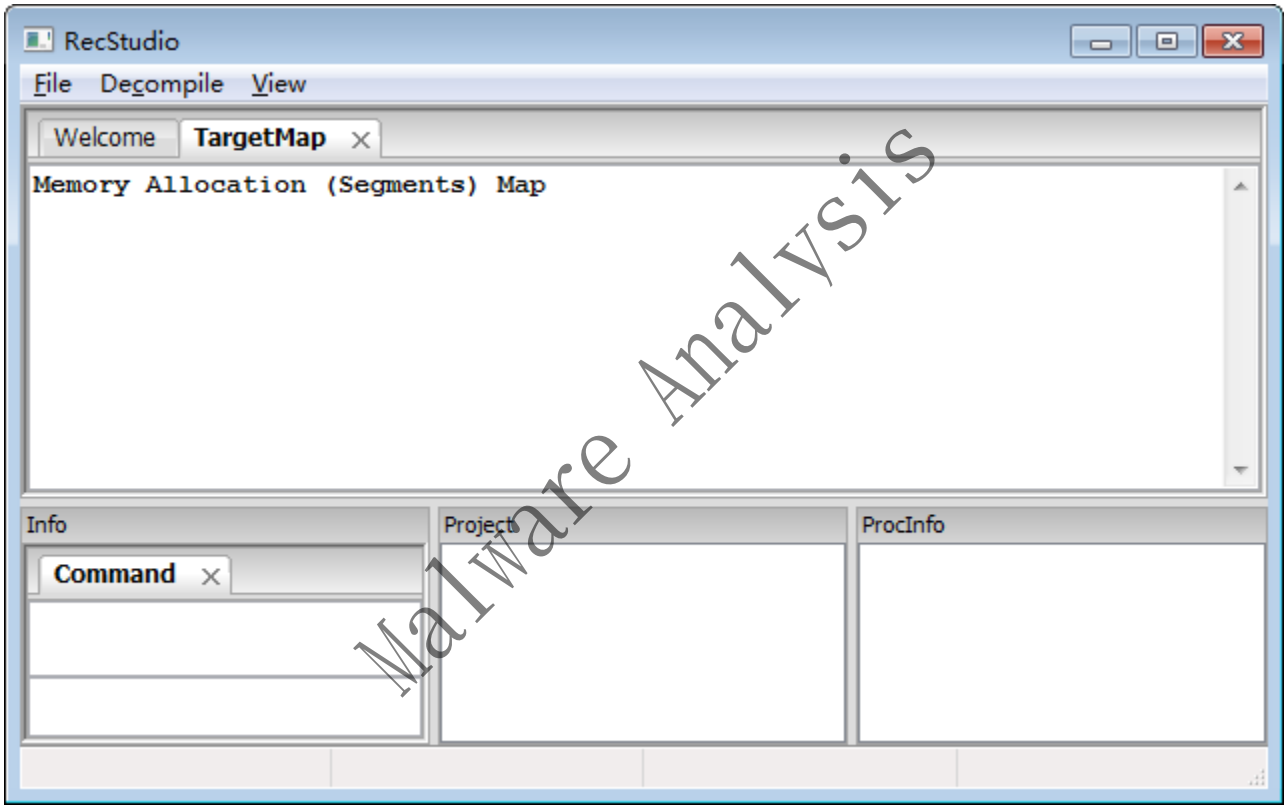
- 反汇编： **IDA Pro，Ollydbg，VC，…**



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

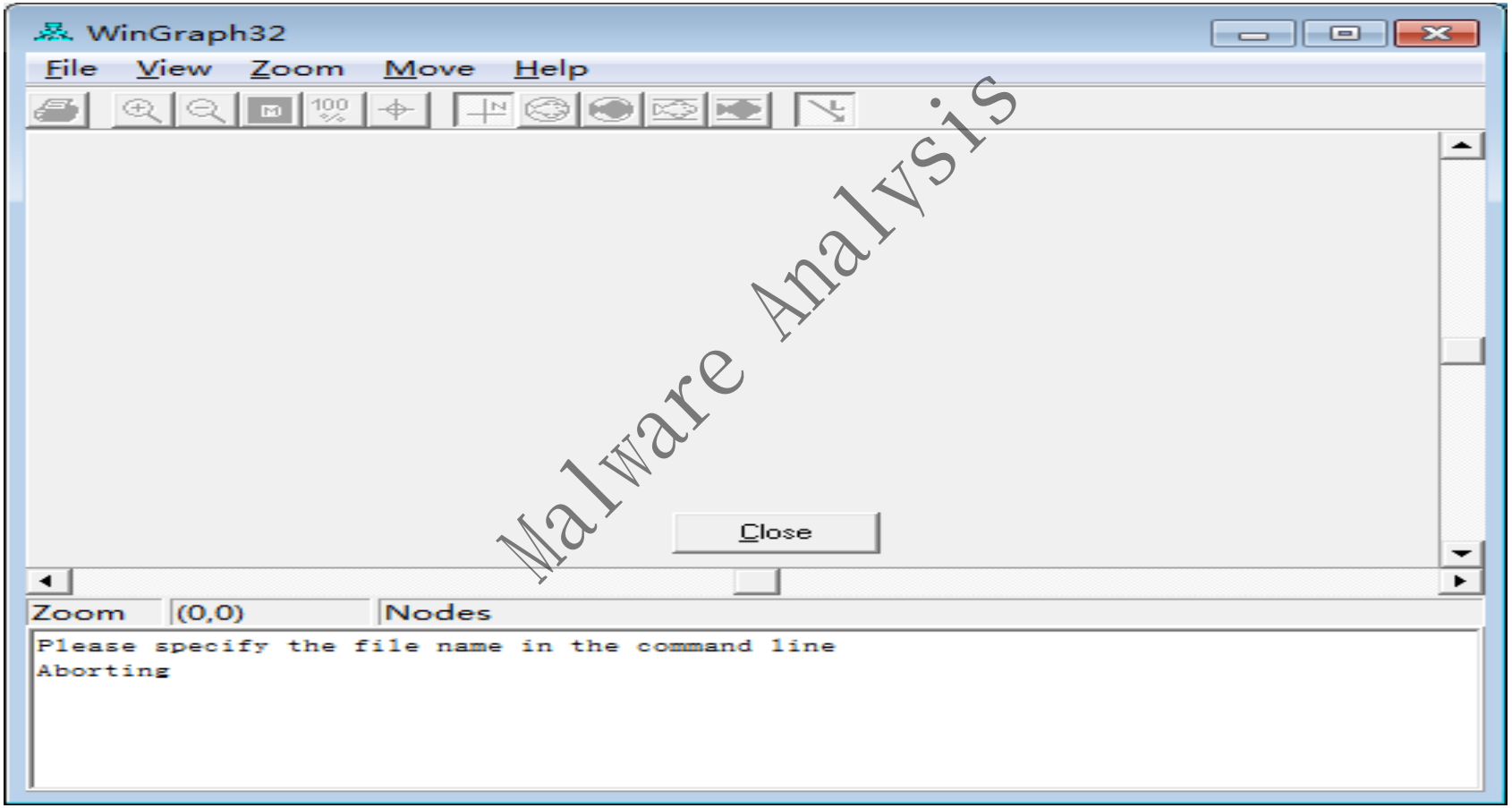
RecStudio



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）**

WinGraph32


WinGraph32
File View Zoom Move Help
Close
Zoom (0,0) Nodes
Please specify the file name in the command line
Aborting
Malware Analysis

CFG（Control Flow Graph）

- **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
- **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**
- 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*

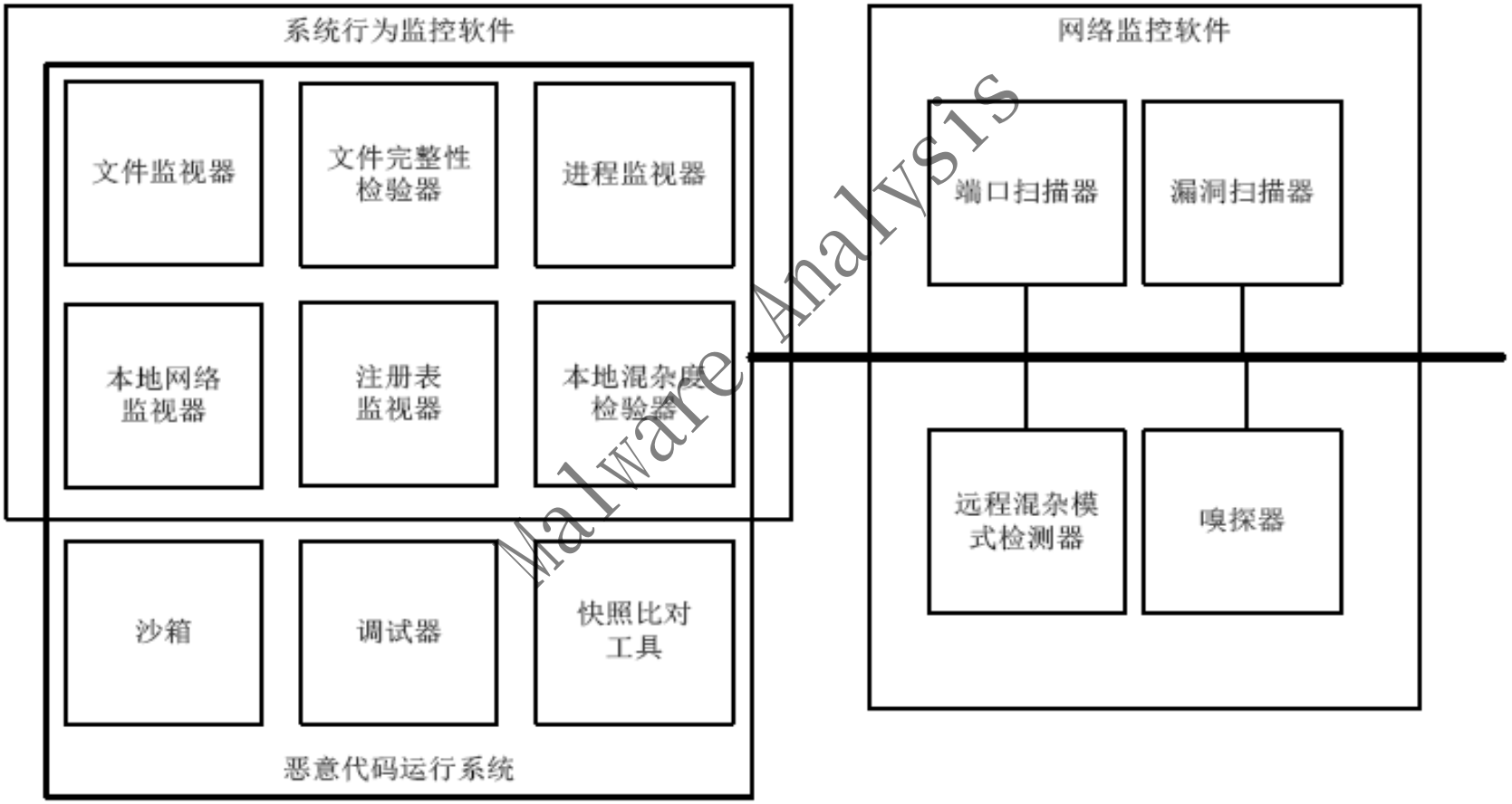
动态扫描和分析方法列表

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

动态分析中的监视与控制



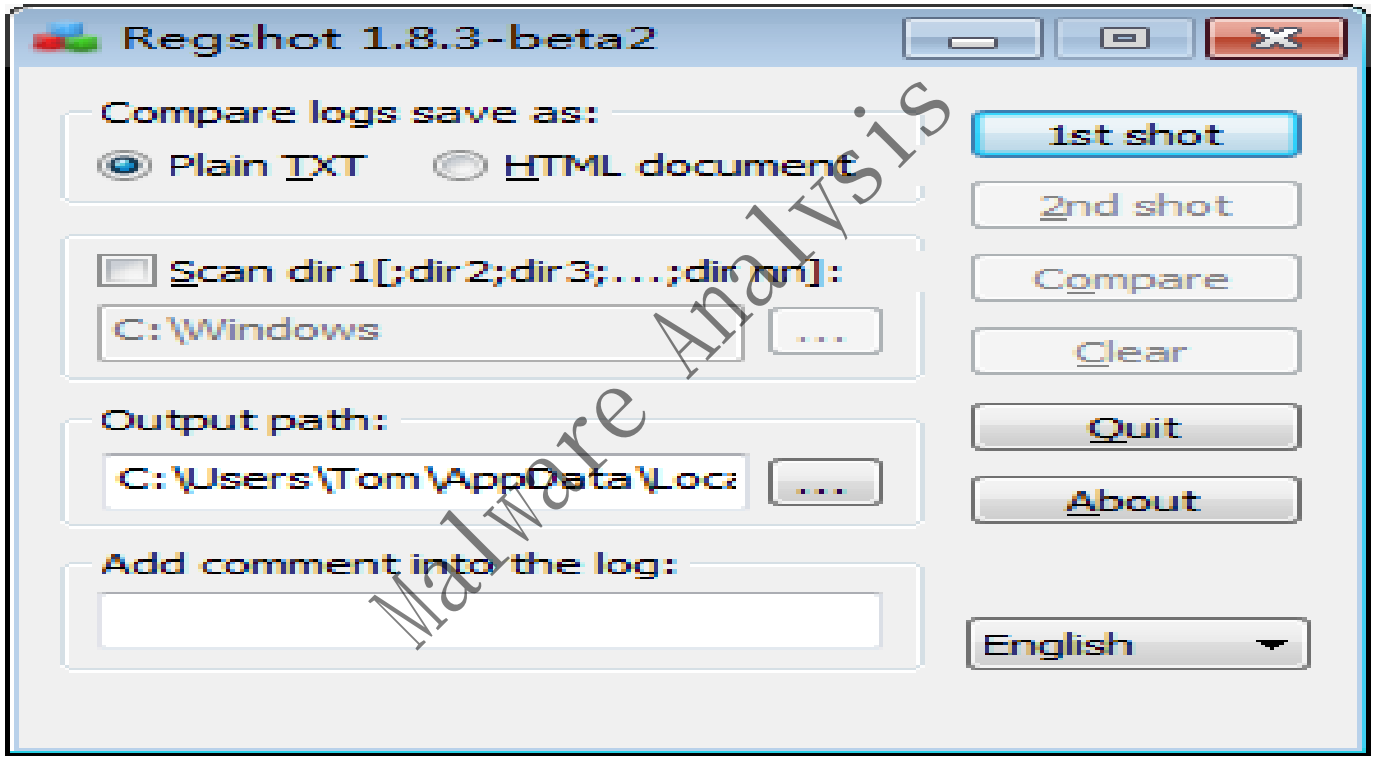
动态行为监控方法

- **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**、**ProcMon**
  - 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**、**…**

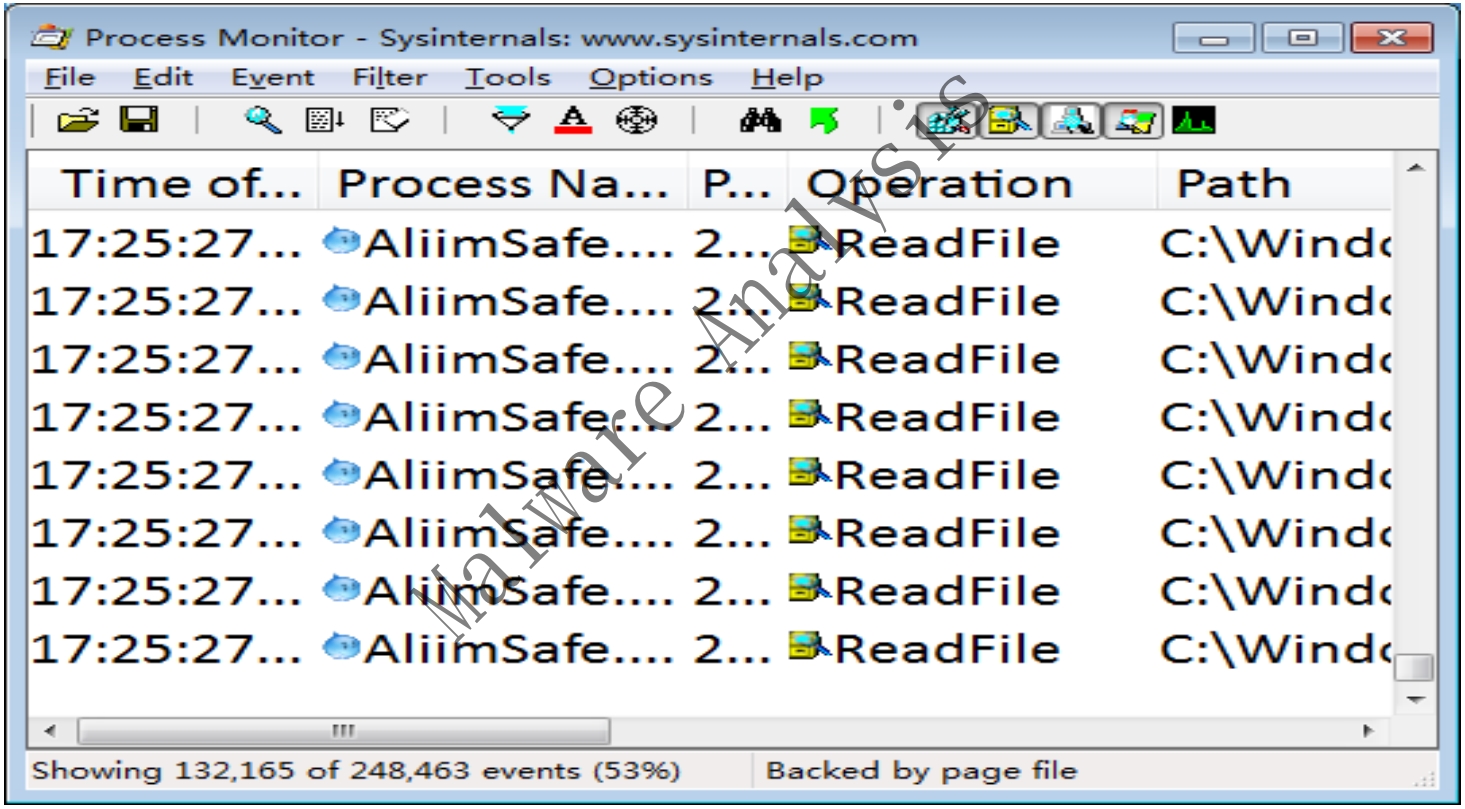
基于快照比对的方法

- 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

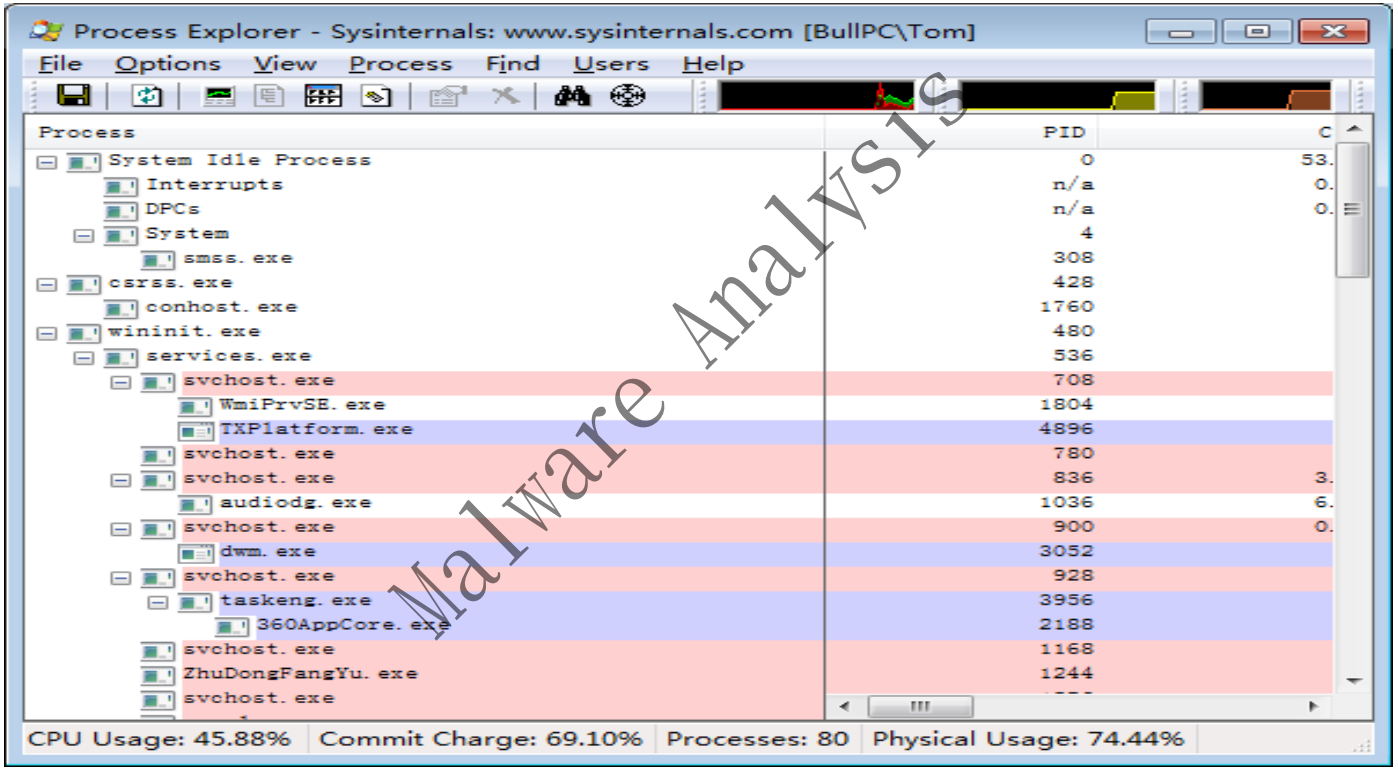
RegShot


Regshot 1.8.3-beta2
Compare logs save as:
Plain TXT
HTML document
Scan dir1[;dir2;dir3;...;dirnn]:
C:\Windows
...
Output path:
C:\Users\Tom\AppData\Loca
...
Add comment into the log:
1st shot
2nd shot
Compare
Clear
Quit
About
English
Malware Analysis

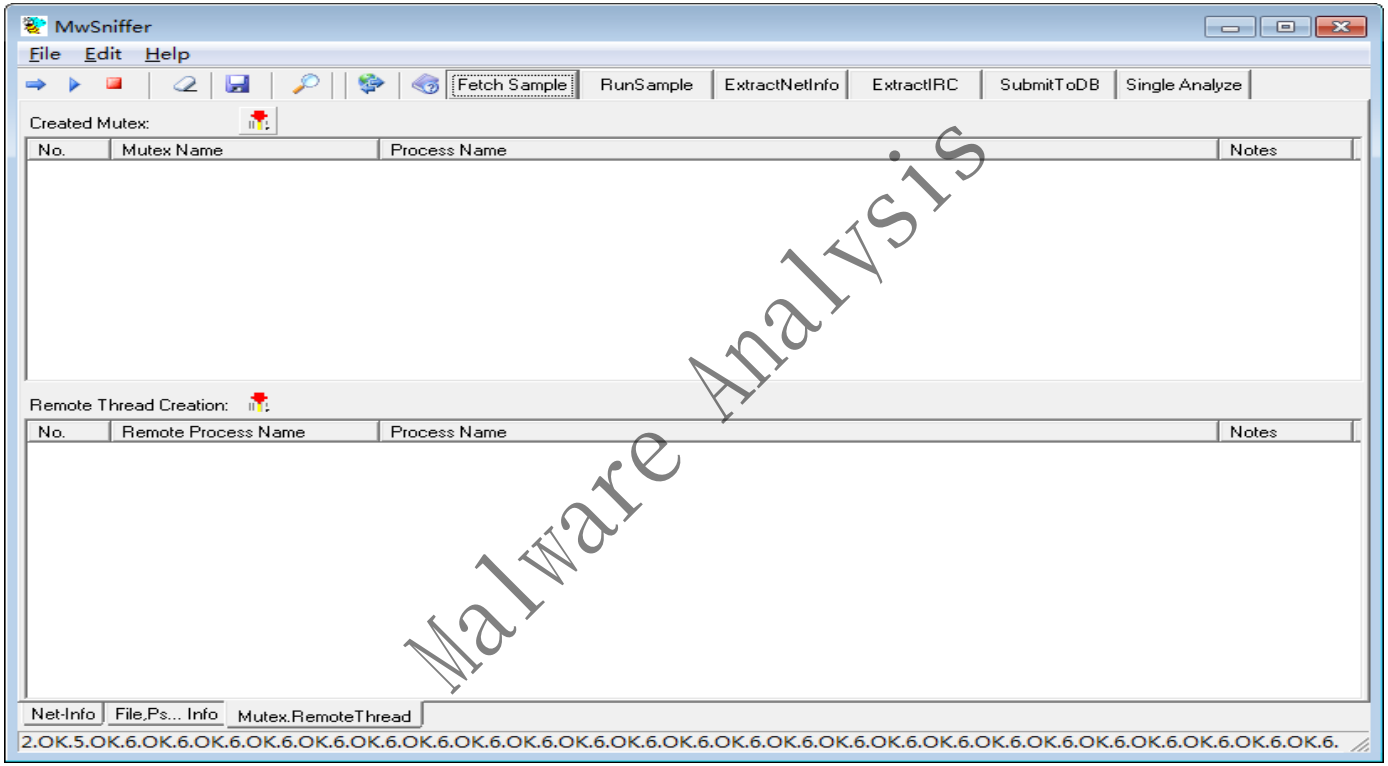
Process Monitor—文件、进程监控



Process Explorer—进程行为监控



MwSniffer –系统行为监控



“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
    - 并行化->标配服务器**64**路并行

动态调试技术

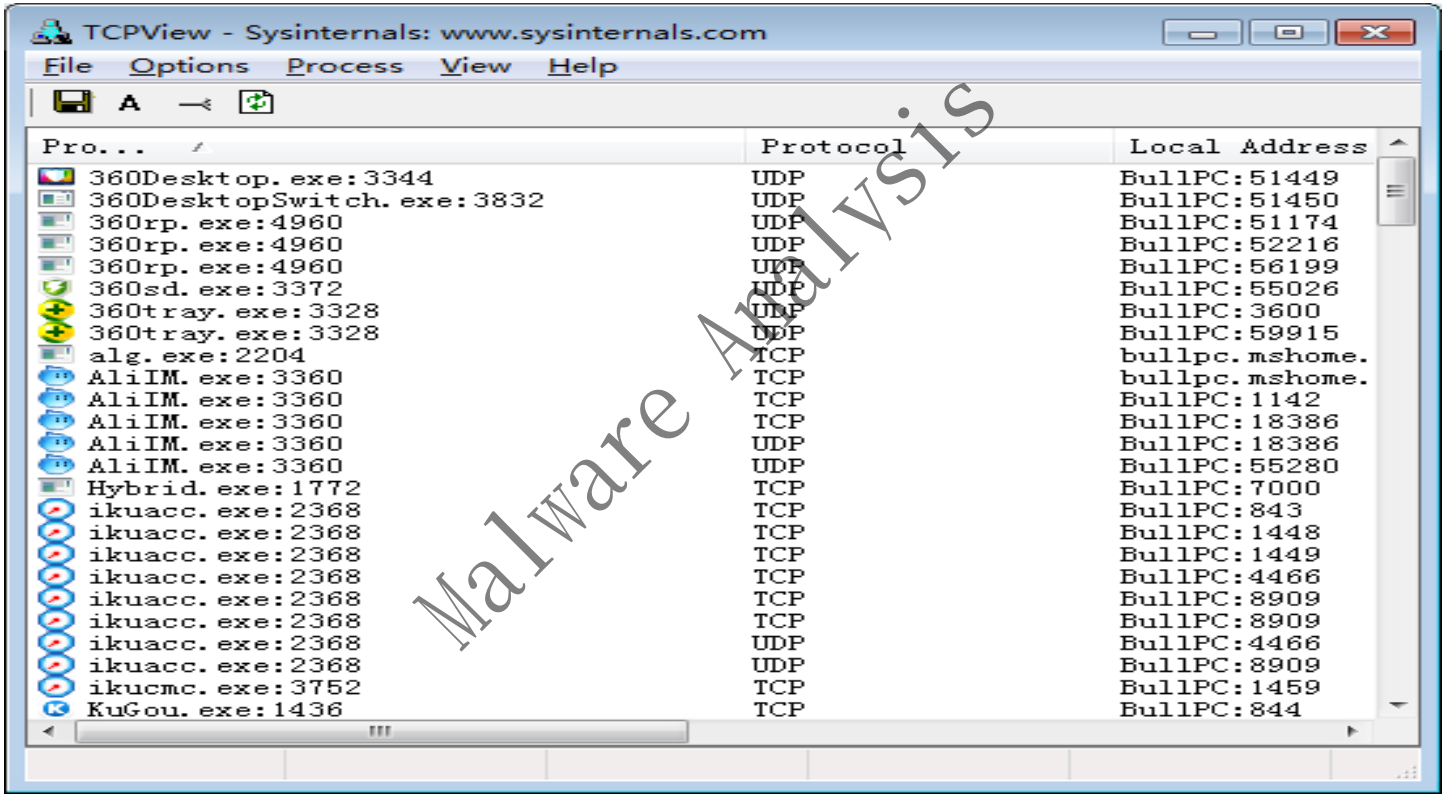
- 动态调试
  - 程序运行时刻
  - 它的执行过程进行调试（**debugging**）
  - 二进制调试
- 动态调试技术
  - 断点
  - 单步模式
  - 寄存器和内存状态查看与修改
- 动态调试工具
  - **Windows：Ollydbg、windbg、IDA Pro、SoftICE**
  - **Linux：gdb、systrace、ElfShell**

通信行为检测

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**）

TCPView—网络行为监控



相似性检测

相似性检测

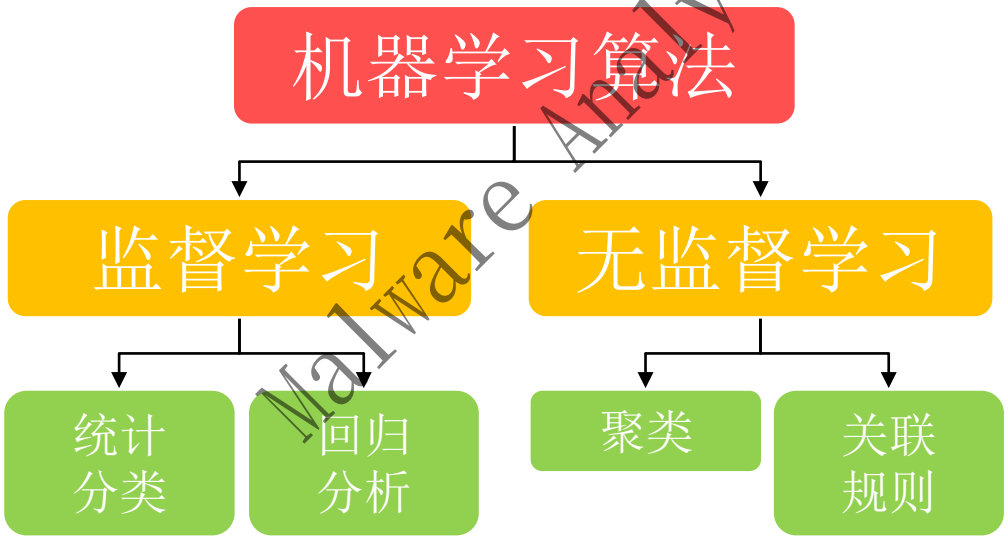
- 高效、快速的数据分析方法
  - 恶意代码呈现出爆发式增长的趋势，日均出现上百万的样本量。但是当前恶意代码分析的形式主要依赖人工提取特征码。相关资料显示，平均一名熟练的分析人员一天只能分析12.8 个样本，供需矛盾严重。
- 主要包括2方面
  - 考虑如何抽取或演化出针对恶意软件家族或者特定平台的精确特征，主要是动静态特征信息。
  - 研究如何高效精确地对样本进行聚类与分类，进而检测。

基于机器学习的相似性分析

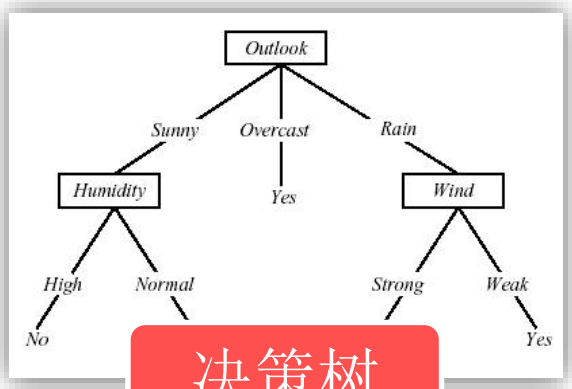
- 机器学习
  - 机器学习是近20多年兴起的一门多领域交叉学科，涉及概率论、统计学、逼近、凸分析、算法复杂度理论等多门学科。机器学习理论主要是设计和分析一些让计算机可以自动“学习”的算法。即从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法
- 机器学习算法
  - 机器学习算法的主要任务是分类和回归。分类主要应用于离散型数据，回归主要用于预测数值型连续的数据，例如拟合曲线等。
  - 使用机器学习算法首先考虑目的，如果想要预测目标变量，则可以选择监督学习算法，否则可以选择无监督学习算法。其次考虑数据问题，主要了解特征值是离散还是连续，是否存在缺失及异常。一般来说发现好算法的关键是反复试错的迭代过程。

机器学习常见算法

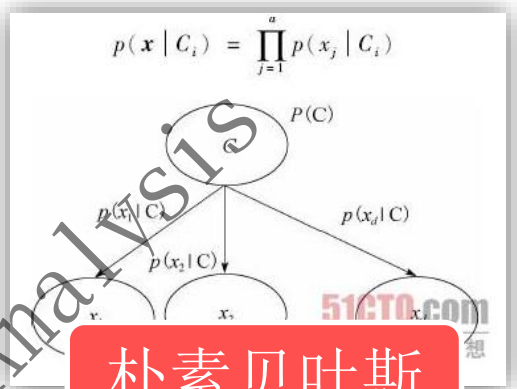
- 常见的机器学习算法



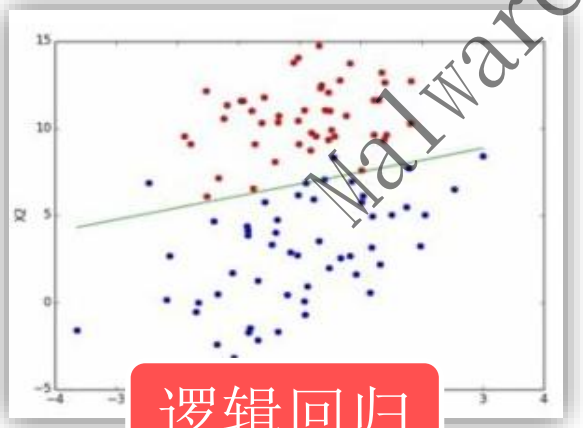
监督学习算法



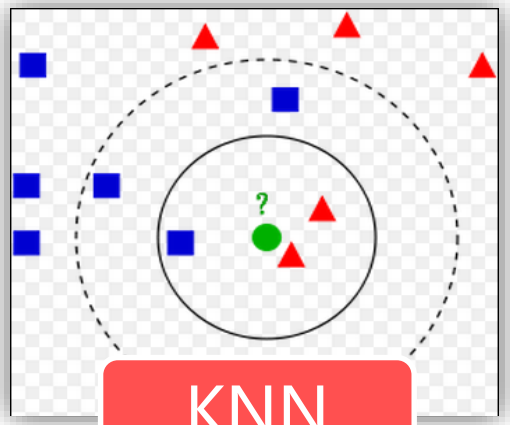
决策树



朴素贝叶斯

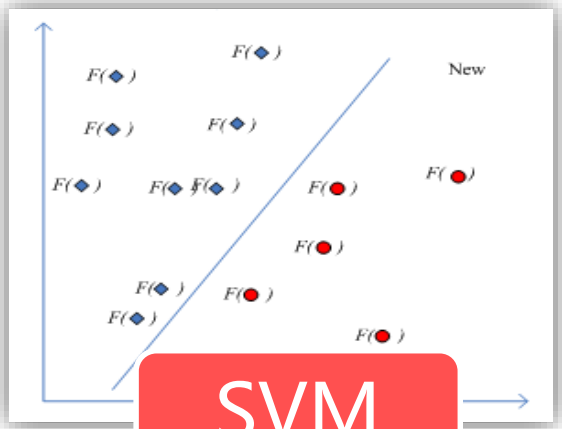


逻辑回归

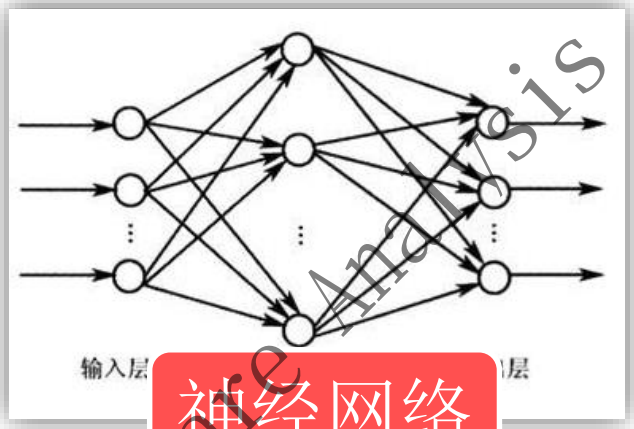


KNN

监督学习算法



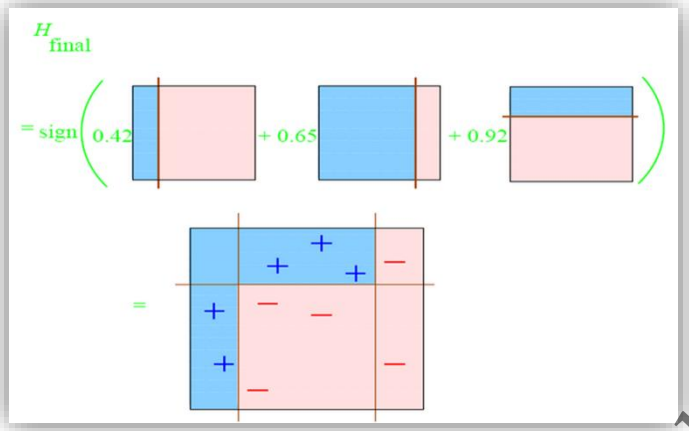
SVM



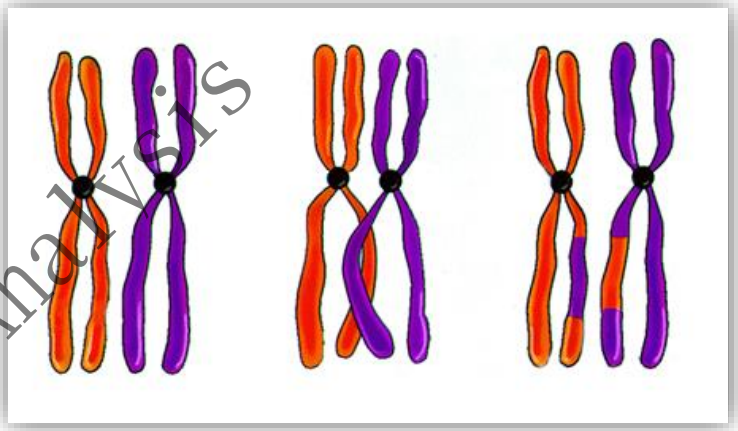
神经网络

随机森林

算法优化

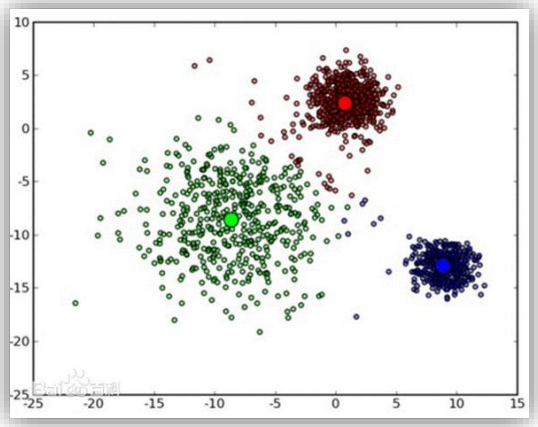


AdaBoost

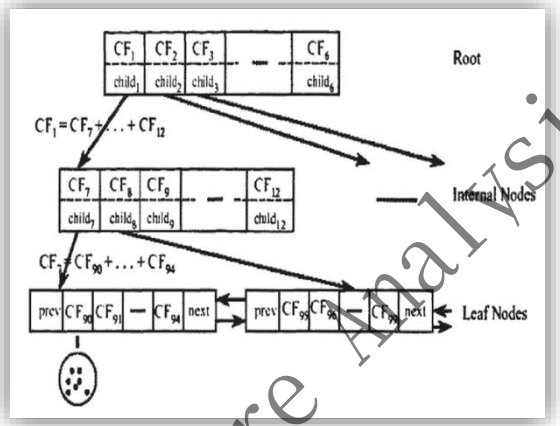


遗传算法

无监督学习算法



K-means



BIRCH

$\overline{C}_2$

TID	Set-of-Itemsets
100	{ {1 3} }
200	{ {2 3}, {2 5}, {3 5} }
300	{ {1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5} }
400	{ {2 5} }

Apriori

机器学习检测恶意代码基本框架

- 机器学习算法：基本思想就是，利用数据挖掘可以从已存在的大量数据中挖掘出有意义的模式，利用机器学习可以帮助归纳出已知恶意代码的识别知识，以此来进行相似性搜索，帮助发现未知恶意代码。
- 实现基于数据挖掘和机器学习的恶意代码检测系统，有两个关键：
  - 选择于分类相关的特征
  - 选择最有效的分类器

基于相似性分析的方法（例）

- 特征
  - 恶意代码有多种特征，可以通过API（ Application Program Interface ）函数调用序列、指令序列、PE文件头、机器码字节序列、字符串等特征来实现恶意代码检测
- 恶意代码检测的几个思路
  - 1、行为特征：攻击树，行为：API调用，威胁指数 stv(Parent)=∑sty(Child)*weight
  - 2、归一化：恶意代码归一化，再进行特征码匹配，针对恶意代码变形
  - 3、纹理指纹：恶意代码映射为无压缩灰阶图片，结合图像分析技术分析代码变种
  - 4、有效窗口和朴素贝叶斯：有效窗口降低恶意代码噪音，朴素贝叶斯进行分类

特征提取和处理（例）

- 例如：分为以下如下步骤：
  - 第一步：特征提取，可分为两类：1、静态特征，如字节序列，PE字符串序列等；2、动态特征，如API系统调用序列、文件与进程操作等。
  - 第二步：特征处理，便于后续学习。方法主要两类：1、不进行任何处理，直接选择提取的特征信息；2、利用N-gram滑动窗口，提取特征序列。
  - 第三步：特征降维，如信息增益、Relife、Fisher、SHI等。
  - 第四步：分类学习，对降维处理后的数据集利用不同学习算法，如决策树、支持向量机、贝叶斯估计等，进行学习，利用学习的分类器对测试集进行测试。
  - 第五步：对测试集进行分析与评估。

特征提取和处理（例）

- 根据恶意代码API调用序列特征生成动作曲线，多个动作曲线重合概率较多的点筛选出，这些点称为可以动作点，将可疑动作点拟合为可疑动作曲线，根据可疑动作曲线的重合率判断恶意代码，利用贝叶斯算法测试。
- 典型恶意代码API调用序列

- 可疑动作曲线的优点在于相比于典型恶意代码API，减少API序列的随机性，进而减少漏判误判。

特征提取和处理（例）

- 提取恶意代码动态行为特征分析检测基本架构如左图所示。
- 第一步：搭建运行恶意代码的虚拟环境，便于加壳等一些处理过的恶意代码分析。
- 第二步：虚拟环境中执行恶意代码，并提取恶意代码动态行为语义特征。本课题选取恶意代码执行期间调用的API序列作为语义特征。
- 第三步:使用提取的语义特征进行模型建立，即组织语义信息建立语义模型。本课题对运行API调用序列加工，抽象为代表语义特征的动作曲线。
- 第四步：建立模型库，采用朴素贝叶斯等进行检测。