Computer System Security **CS3312**

# 计算机系统安全

2024年 春季学期

主讲教师：张媛媛 副教授

上海交通大学 计算机科学与技术系


# 第十一章

# 软件安全：恶意软件

Malicious Software

# 目录/CONTENTS


# 01

# 恶 意 软 件 概 览

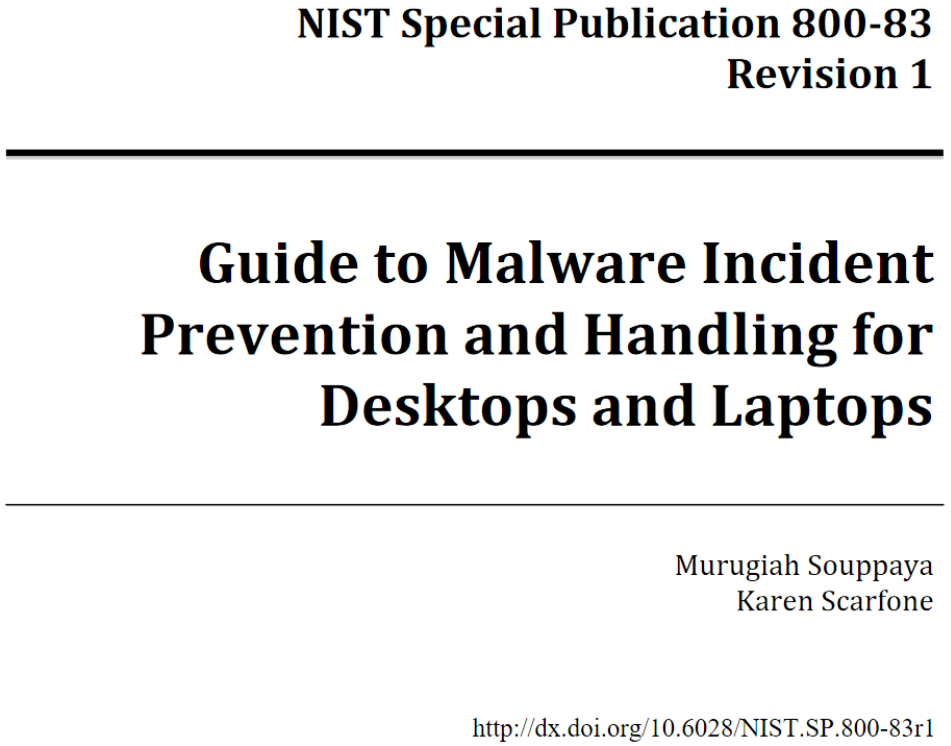
O v e r v i e w

# 漏洞代码 vs. 恶意代码

恶意软件(Malicious Software, Malware)：通常是秘密地插入到系统中的一种程序，其目的是损害目标系统的数据、应用程序或操作系统的机密性、完整性或可用性，或以其他方式骚扰或干扰目标系统运行。

恶意软件——主动，以破坏为目的
漏洞代码——被动，无特定目的

| | | |
|---|---|---|
| advanced persistent threat | infection vector | ransomware |
| adware | keyloggers | rootkit |
| attack kit | logic bomb | scanning |
| backdoor | macro virus | spear-phishing |
| blended attack | malicious software | spyware |
| boot-sector infector | malware | stealth virus |
| bot | metamorphic virus | trapdoor |
| botnet | mobile code | Trojan horse |
| crimeware | parasitic virus | virus |
| data exfiltration | payload | watering-hole attack |
| downloader | phishing | worm |
| drive-by-download | polymorphic virus | zombie |
| e-mail virus | propagate | zero-day exploit |



NIST Special Publication 800-83
Revision 1

Guide to Malware Incident Prevention and Handling for Desktops and Laptops

Murugiah Souppaya
Karen Scarfone

http://dx.doi.org/10.6028/NIST.SP.800-83r1

# 恶意行为

## 传播

病毒感染现有的可执行文件或解释内容，并随后传播到其他系统
利用软件漏洞，无论是本地或通过网络蠕虫或驱动下载，以允许恶意软件复制
社会工程攻击，说服用户绕过安全机制安装木马，或钓鱼攻击

## 有效载荷

损坏系统或数据档案;
窃取服务，以使系统成为僵尸网络的僵尸代理攻击的一部分;
通过键盘记录或间谍软件程序从系统中窃取信息，特别是登录名、密码或其他个人信息;
偷窃是指恶意软件在系统中隐藏自己的存在，不被检测和阻止。

# 恶意软件类型

## 病毒
病毒通过将自己的副本插入到主程序或数据文件中进行自我复制。病毒通常是通过用户交互触发的，例如打开文件或运行程序。

## 蠕虫
蠕虫是一种自我复制、自包含的程序，通常在没有用户干预的情况下自行执行。

## 特洛伊木马
特洛伊木马是一个独立的、非复制的程序，虽然看起来是良性的，但实际上有一个隐藏的恶意目的。木马病毒要么用恶意版本替换现有文件，要么向主机添加新的恶意文件。他们经常向主机交付其他攻击者工具。

## 恶意移动代码
恶意移动代码是具有恶意意图的软件，它从远程主机传输到本地主机，然后在本地主机上执行，通常不需要用户的明确指令。目前流行的恶意移动代码语言包括Java、ActiveX、JavaScript和VBScript。

## 混合攻击
混合攻击使用多种感染或传播方法。例如，混合攻击可以结合病毒和蠕虫的传播方法。


# 02

# 病毒、蠕虫、僵尸网络

Virus, Worm and Botnet

# 病毒 Virus

计算机病毒是一种可以通过修改其他程序或任何可执行内容而“感染”其他程序的软件。寄生在其他程序上。

## 病毒包含三个组件：

感染机制
病毒传播或传播的方式，使其能够复制。这种机制也被称为感染载体。

触发
决定有效载荷何时被激活或交付的事件或条件，有时被称为逻辑炸弹。

有效载荷
除了传播，病毒还能做什么。有效载荷可能包括损伤，也可能包括良性但明显的活动。

By target:
- Boot sector infector
- File infector
- Macro virus
- Multipartite virus

By concealment strategy:
- Encrypted virus
- Stealth virus
- Polymorphic virus
- Metamorphic virus

# 病毒传播的四个阶段

## 休眠阶段

病毒是空闲的。该病毒最终将被某些事件激活，如日期，另一个程序或文件的存在，或磁盘容量超过某些限制。并不是所有的病毒都有这个阶段。

## 传播阶段

这种病毒将自己的一个副本放入其他程序或磁盘上的某些系统区域。该副本可能与传播版本不相同;病毒经常变形以逃避检测。每个受感染的程序现在将包含一个病毒的克隆，该病毒本身将进入一个传播阶段。

## 触发阶段

病毒被激活来执行它原本的功能。与休眠阶段一样，触发阶段可能由各种系统事件引起，包括病毒副本复制自身的次数。

## 执行阶段

完成函数的执行。该函数可能无害(如屏幕上的消息)，也可能有害(如破坏程序和数据文件)。

# Morris蠕虫

历史上第一个蠕虫：Robert Morris, 1988
最早的，因此众所周知的蠕虫感染是由罗伯特·莫里斯（Robert Morris）于1988年发布到Internet上的[ORMA03]

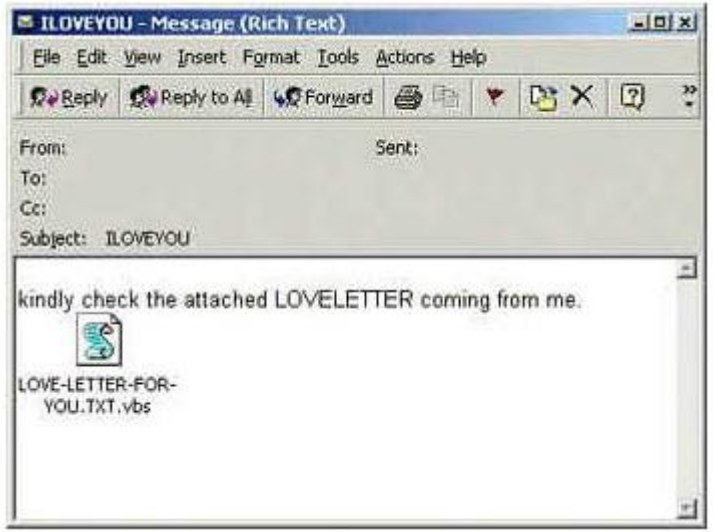
Methods for gaining access:

- It attempted to log on to a remote host as a legitimate user. First attempted to crack the local password file then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried:
  - Each user’s account name and simple permutations of it.
  - A list of 432 built-in passwords that Morris thought to be likely candidates.
  - All the words in the local system dictionary.
- It exploited a bug in the UNIX finger protocol, which reports the whereabouts of a remote user.
- It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

# 蠕虫 Worm

A worm typically uses the same phases as a computer virus

- Dormant
- Propagation
- Triggering
- Execution




Send out such email proactively, using user's address book + enticing subject (e.g. "I Love you" email)

# 登录远程系统

Electronic mail or instant messenger facility:

- A worm e-mails a copy of itself to other systems, or sends itself as an attachment via an instant message service

File sharing:

- A worm either creates a copy of itself or infects other suitable files as a virus on removable media such as a USB drive; it then executes when the drive is connected to another system using the autorun mechanism by exploiting some software vulnerability, or when a user opens the infected file on the target system.

Remote execution capability:

- A worm executes a copy of itself on another system, either by using an explicit remote execution facility or by exploiting a program flaw in a network service to subvert its operations.

Remote file access or transfer capability:

- A worm uses a remote file access or transfer service to another system to copy itself from one system to the other, where users on that system may then execute it.

Remote login capability:

- A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other, where it then executes.

# 蠕虫的传播

Search for appropriate access mechanisms on other systems to infect by examining host tables, address books, buddy lists, trusted peers, and other similar repositories of remote system access details; by scanning possible target host addresses; or by searching for suitable removable media devices to use.

- Scanning targets (network address scanning strategies)
- Random
- Hit-list
- Local subnet

Use the access mechanisms found to transfer a copy of itself to the remote system, and cause the copy to be run.

# 蠕虫传播建模

Worm-spread often well described as infectious epidemic

- Classic SI model: homogeneous random contacts
  - SI: susceptible infectible

Model parameters:

- N: population size
- S(t): susceptible hosts at time t
- I(t): infected hosts at time t  N = S(t) + I(t)
- β: contact rate
  - How many population members each infect host communicates with per unit time
  - E.g. If each infected host scans 10 Internet addresses per unit time, and 2 of Internet addresses run a vulnerable server, then β=0.2

Normalized versions reflecting relative proportion of infected/susceptible hosts

- s(t) = S(t)/N  i(t) = I(t)/N  s(t)+i(t) = 1

# 蠕虫传播建模

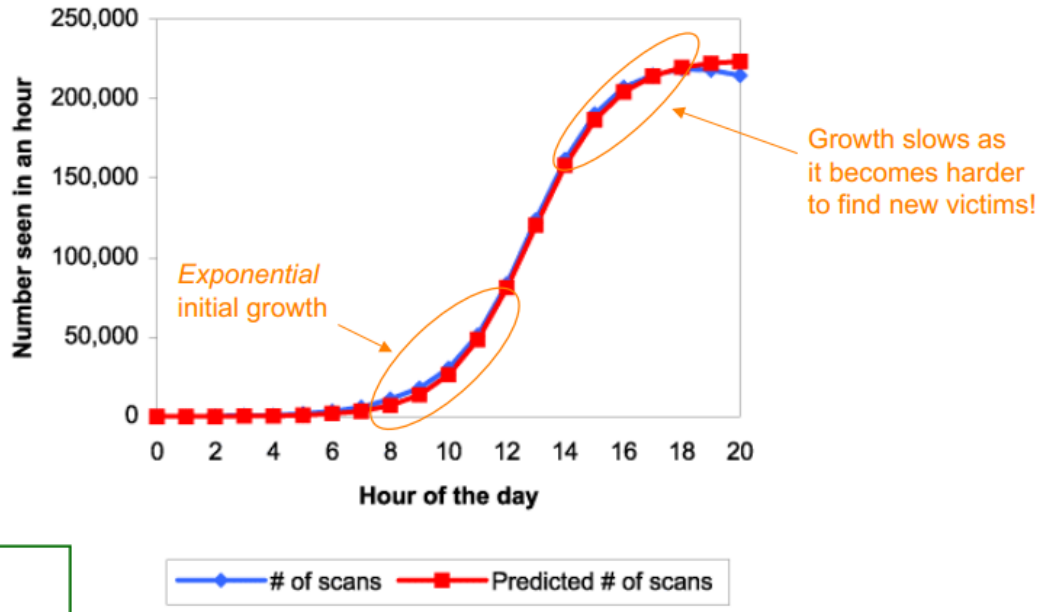
- In continuous time:

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Proportion of contacts expected to succeed

Total attempted contacts per unit time




- Rewriting by using i(t) = I(t)/N, S = N - I:

$$\frac{di}{dt} = \beta i(1-i) \quad \Rightarrow \quad i(t) = \frac{e^{\beta t}}{1+e^{\beta t}}$$

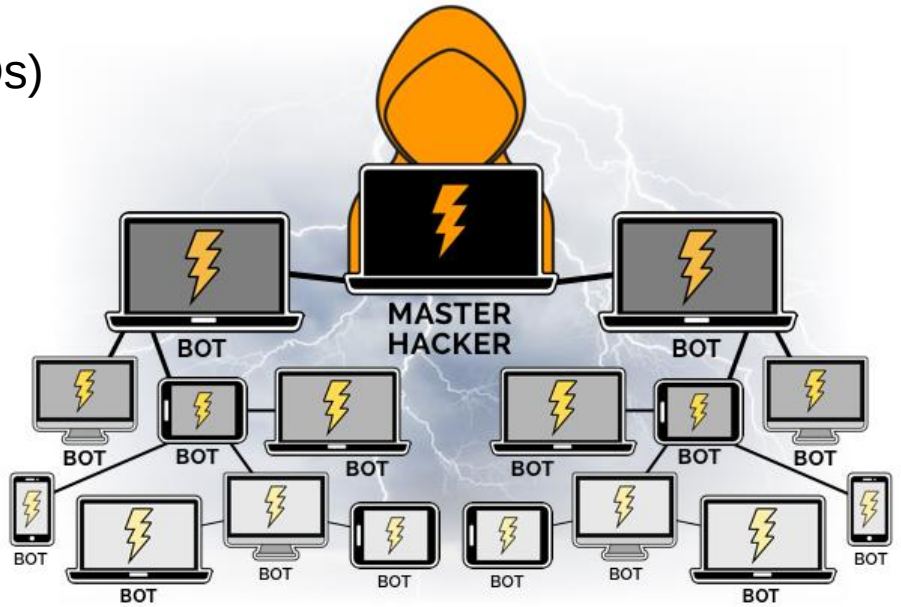
Fraction infected grows as a *logistic*

# 僵尸网络

- Distributed denial-of-service (DDoS) attacks
- Spamming
- Sniffing traffic
- Keylogging
- Spreading new malware
- Installing advertisement add-ons and browser helper objects (BHOs)
- Manipulating online polls/games

mirai

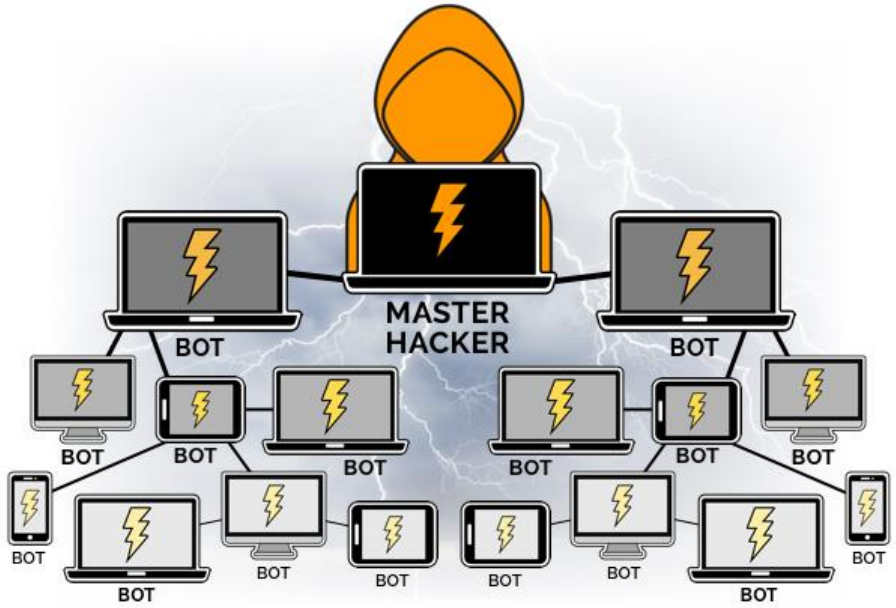
zeus



# 区别于蠕虫

The **remote control facility** is what distinguishes a bot from a worm.

A worm propagates itself and activates itself, whereas a bot is controlled by some form of **command-and-control (C&C)** server network. This contact does not need to be continuous, but can be initiated periodically when the bot observes it has network access.




# 03

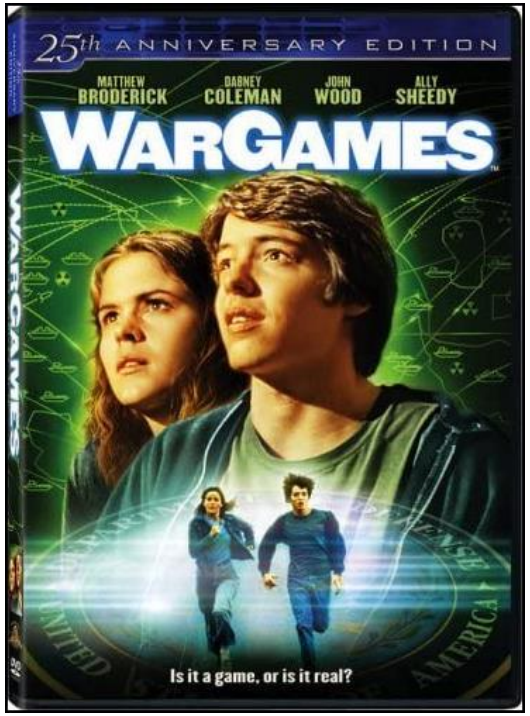
# 其他恶意软件

Other Malware

# 后门 Backdoor

A backdoor, also known as a trapdoor, is a secret entry point into a program that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures.

- 1983年电影《战争游戏》(War Games)中描述的漏洞的基本概念。
- 在Multics的开发过程中，渗透测试是由空军“老虎队”（模拟对手）Air Force “tiger team” 进行的。 使用的一种策略是将伪造的操作系统更新发送给 运行Multics的站点。更新中包含一个特洛伊木马，该特洛伊木马可以通过后门激活，从而使虎队能够进入。 威胁实施得非常好，以至于Multics开发人员都无法找到它，即使他们被告知存在该威胁



# Rootkit

Rootkit 是安装在系统上的一组程序，可以达到隐藏其他程序进程的目的。现在更多地是指被作为驱动程序，加载到操作系统内核中的恶意软件。因为其代码运行在特权模式之下，该攻击能提供对操作系统的所有功能和服务的访问。

内核空间由四大部分组成

- 进程管理（负责分配CPU时间）
- 文件访问（把设备调配成文件系统，并提供一个一致的接口，供上层程序调用）
- 安全控制（负责强制规定各个进程的具体的权限和单独的内存范围，避免各进程之间发生冲突）
- 内存管理（负责进程运行时对内存资源的分配、使用、释放和回收）

内核是一种数据结构，rootkit技术通过修改这些数据结构来隐藏其它程序的进程、文件、网络通讯和其它相关信息（比如注册表和可能因修改而产生的系统日志等）。

例如，通过修改操作系统的EPROCESS链表结构可以达到隐藏进程的效果，hook服务调用表可以隐藏文件和目录，挂钩中断描述符表则可以监听键盘击键等等。Rootkit至今仍然是一个发展中的技术领域。

# Rootkit 表现形态

- Persistent
  每次系统启动时加载的rootkit。
- Memory based
  不具备persistent特性，重启系统rootkit将会消失。
- User mode
  拦截普通API调用，并修改返回值（hook）。
- Kernel mode
  拦截内核模式下的native API调用，并修改返回值。
- VM based
  这类rootkit会安装一个轻量级的虚拟机控制器，在其上运行一个小型的指令翻译或调度系统软件（类OS）。
- External mode
  常见于OS之外，存活于BIOS或系统管理模式中，一般都会尽量靠近硬件。

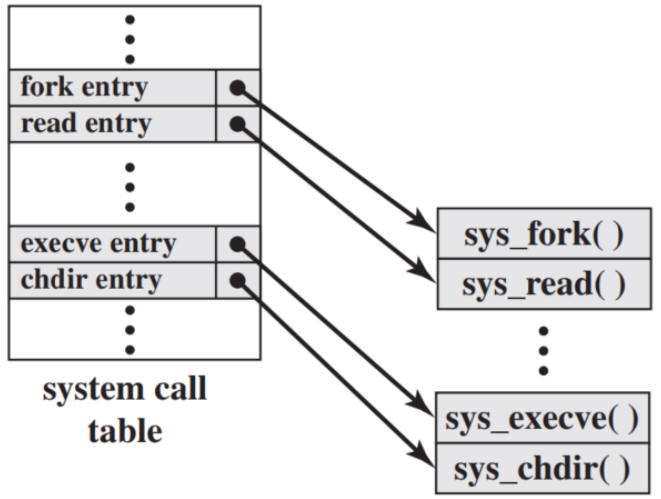
# 内核中的Rootkit

下一代 rootkit 从 user mode 向下移动了一层，在内核内部进行更改并与操作系统代码共存，以使其检测更加困难。
它们主要的 hook 目标：系统调用

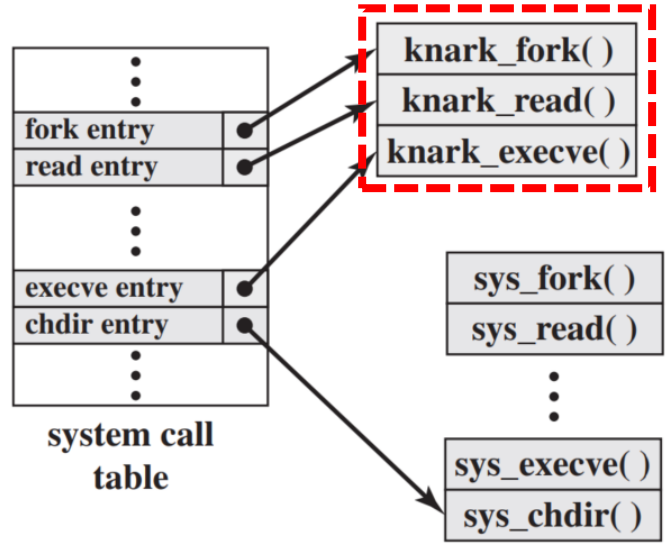
**修改系统调用表：**
攻击者修改存储在系统调用表中的选定系统调用地址。
这使 rootkit 能够将系统调用从合法例程引导到 rootkit 的替换。

**修改系统调用表目标（如右图示例）：**
攻击者用恶意代码覆盖选定的合法系统调用例程。
系统调用表没有改变。

**重定向系统调用表：**
攻击者将对整个系统调用表的引用重定向到新内核内存位置中的新表。



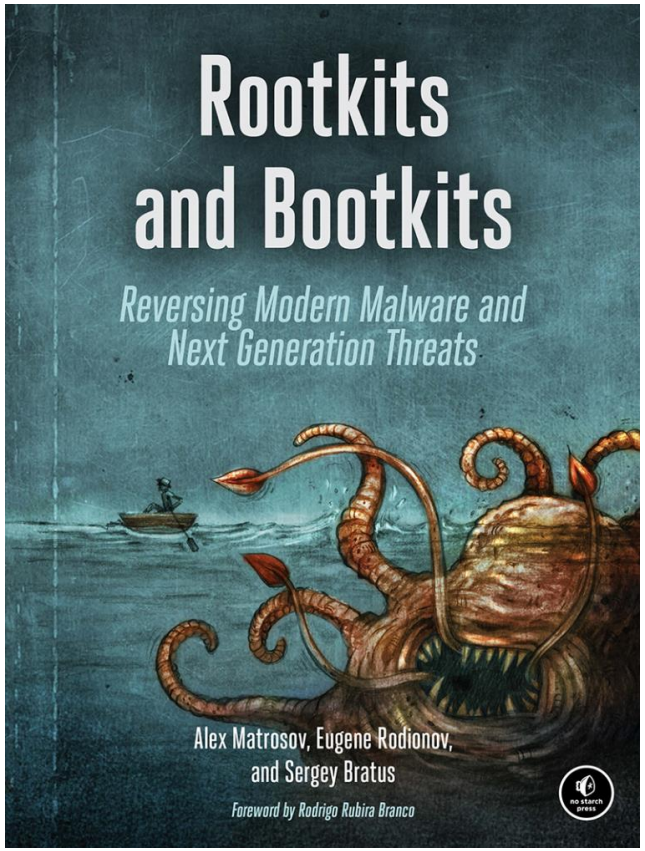

(a) Normal kernel memory layout




(b) After knark install

Computer System Security CS3312

# Rootkit + APT




## BRIEF CONTENTS


# 04

# 恶 意 代 码 隐 藏

Malicious Code Concealment

# 代码变形

- **Polymorphic Code (多态)**
- Metamorphic Code (变形/态)

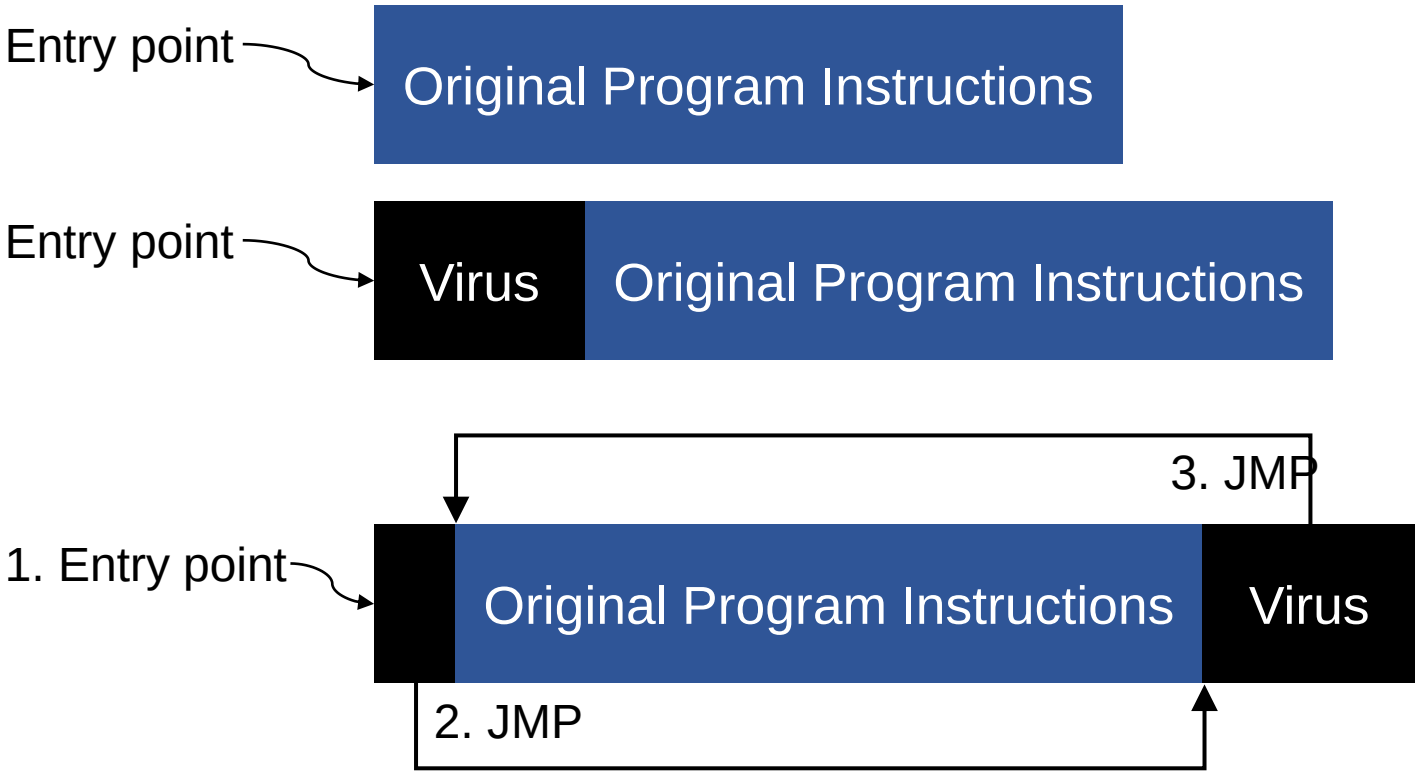
# 代码的多态 Polymorphism

一些用于创建与原始数据形态上毫无关联的数据表示的技术，例如加密技术，编码技术。

Idea：每次你的恶意软件传播时，它都会插入一个新的“转换”，例如 自身的加密副本
显然，加密需要改变
每次使用不同的密钥
或通过包含一些随机的初始填充（如 IV）

Note：多数场合弱加密算法够用（如移位、异或等），不需要强加密算法（如AES等）。
当注入的代码运行时，它会解密以获得原始功能

# Polymorphic Code




- Application the user runs
- Run-time library / routines resident in memory
- Disk blocks used to boot OS
- Auto-run file on USB device

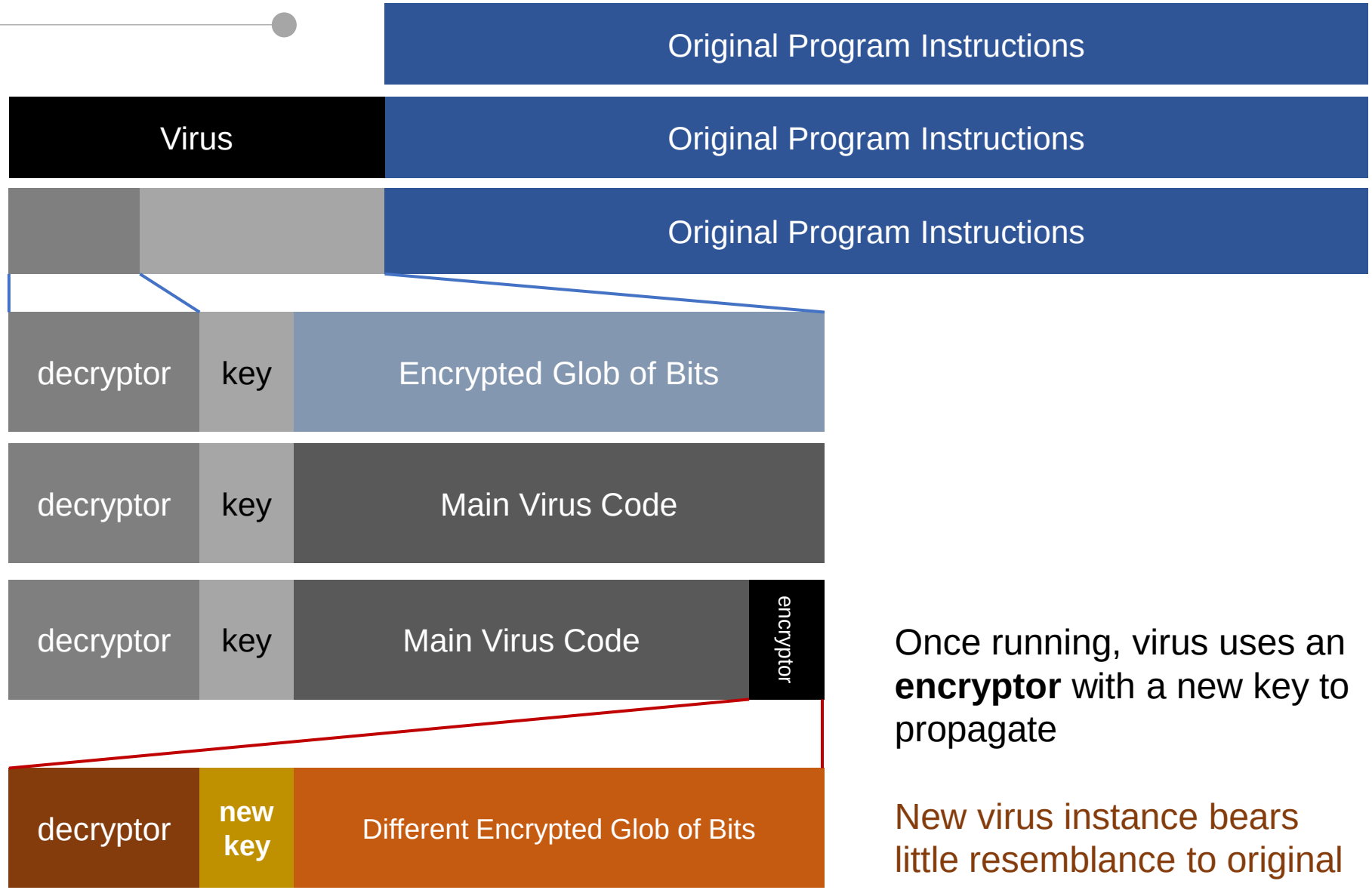
# Polymorphic Code

- Some technology for creating a representation of some data that appears completely unrelated to the original data, such as **encryption**.

- **Idea**: every time your malware propagates, it inserts a newly "transformed" , e.g. encrypted copy of itself
  - Clearly, encryption needs to vary
    - either by using a different key each time
    - or by including some random initial padding (like an IV)
  - Note: weak crypto algorithm works fine, no need for truly strong encryption

- When injected code runs, it decrypts itself to obtain the original functionality

# Polymorphic Code




Once running, virus uses an **encryptor** with a new key to propagate

New virus instance bears little resemblance to original

# Polymorphic Code的检测

- 鉴于多态性，我们如何检测恶意软件?
- 思路一：使用窄信号。——以解密器为目标

问题?

- 更少的代码匹配 → 更多的误报
- 恶意代码体中存在病毒作者编写的解密器

# Polymorphic Code的检测

- 鉴于多态性，我们如何检测恶意软件?
- 思路二：部分执行（或静态分析）可疑代码，看是否包含解密步骤

问题?

- 合法的“打包者”执行类似的操作（解压缩）
- 你让新代码执行多长时间?
- 如果解密器仅在经过长时间的合法执行后才起作用，则很难发现

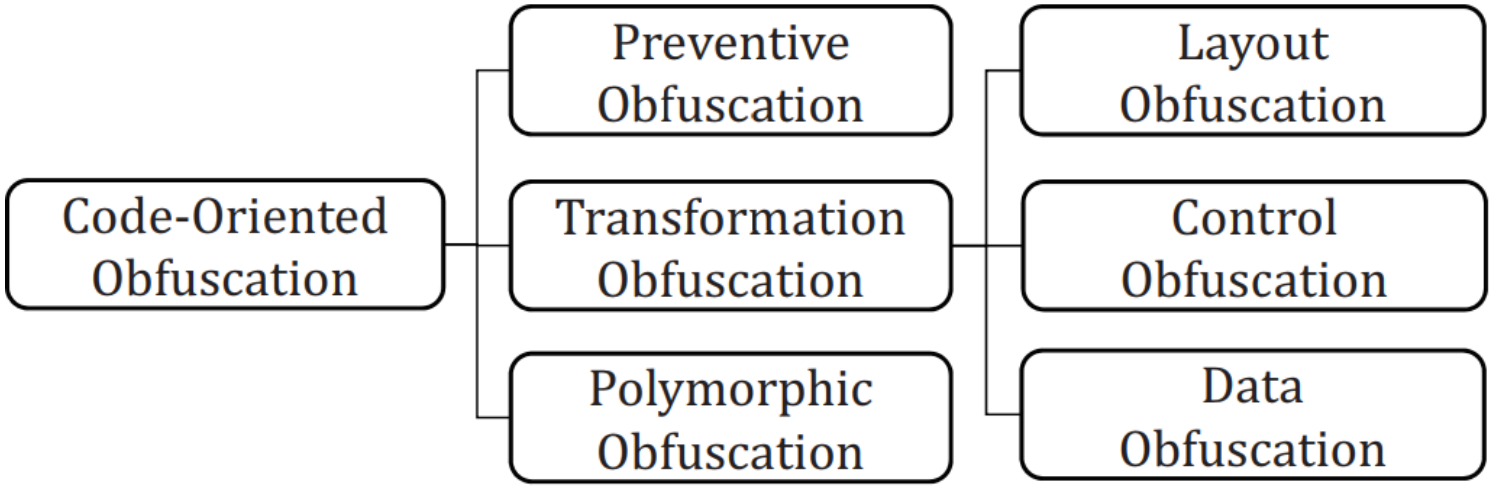
# 代码变形

- Polymorphic Code (多态)
- **Metamorphic Code (变形/态)**

# Metamorphic Code

Idea：每次病毒传播时，生成语义不同的版本！

However：仅在执行的immediate level有不同的语义； 更高级别的语义保持不变

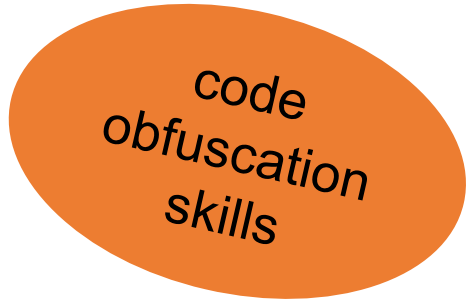


# Metamorphic Code

技术实现：在病毒中包含代码重写器，如，检查自己的代码，生成随机变体等

具体的有：

- 重新编号寄存器
- 更改条件代码的顺序
- 重新排序操作不相互依赖
- 将一种低级算法替换为另一种
- 删除一些 “无操作” 填充并用不同的 “无操作” 代码/方式填充替换它



# Metamorphic Code的检测

如何检测metamorphic code

需要分析执行行为
从语法（指令的外观）到语义（指令的效果）的转变

两个阶段：
Anti-v公司分析新病毒以寻找行为特征
终端系统上的反 v 软件分析可疑代码以测试与签名的匹配

病毒作者的对策：
通过花费很长时间来表现行为来延迟分析
长时间 = 等待特定条件，甚至只是时钟时间
检测在分析的环境中执行是否发生，如果是，则行为不同
例如，测试是在调试器中运行，还是在 VM 中运行


# 05

# 恶 意 代 码 缓 解

Malware Mitigation

# 基于主机的扫描程序和基于签名的反恶意软件技术

四代anti-malware历程：

- 第一代：简单的扫描仪
  - 恶意软件的结构和位模式； 节目长度
- 第二代：启发式扫描仪
  - 代码片段； 完整性检查； 加密哈希函数
- 第三代：活动陷阱
  - 捕获恶意活动的一小部分行为
- 第四代：全功能保护
  - 所有的混合

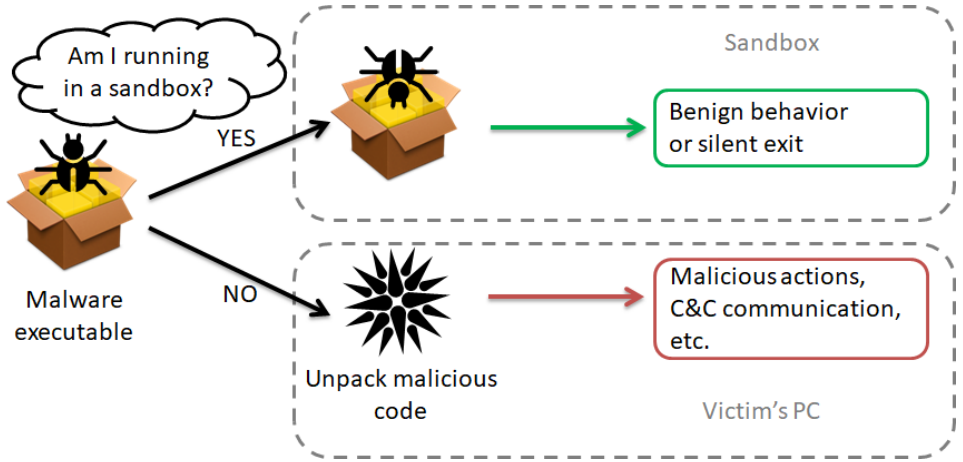
# 沙盒分析（针对第三代恶意软件）

一种检测和分析恶意软件的方法涉及在模拟沙箱或虚拟机上运行潜在的恶意代码。

- 模拟目标系统的内存和 CPU
- 运行潜在的恶意软件
- 检测复杂的加密、多态或变形恶意软件

限制：

- 确定每次解释运行多长时间
- 沙盒分析只有有限的时间和资源可用
- 一些恶意软件会检查它是否在沙箱或虚拟机中运行
- 很难抓住逻辑炸弹



# 基于主机的动态恶意软件分析

动态恶意软件分析或行为阻止软件与主机的操作系统集成，并实时监控程序行为以发现恶意行为。

该软件监控可能的恶意代码的行为，寻找潜在的恶意行为，类似于沙盒系统，但不是。
该技术的目标是尽量在恶意行为影响目标系统之前阻止它们。

- 尝试打开、查看、删除和/或修改文件
- 尝试格式化磁盘驱动器和其他不可恢复的磁盘操作
- 修改可执行文件或宏的逻辑
- 修改关键系统设置，例如启动设置
- 编写电子邮件和即时消息客户端脚本以发送可执行内容
- 启动网络通信

# Rootkit 缓解

Rootkit 可能非常难以检测和中和，特别是对于内核级 Rootkit。

- 基于签名的检测
- 查找行为，例如拦截系统调用

如果检测到内核级 rootkit，唯一安全可靠的恢复方法是在受感染的计算机上执行全新的操作系统安装。


Virustotal is a **service that analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. More information...

1 VT Community user(s) with a total of 1 reputation credit(s) say(s) this sample is goodware. 6 VT Community user(s) with a total of 8 reputation credit(s) say(s) this sample is malware.

File name: **4.doc**
Submission date: **2011-04-19 07:19:30 (UTC)**
Current status: **finished**
Result: **27 /42 (64.3%)**

**VT Community**

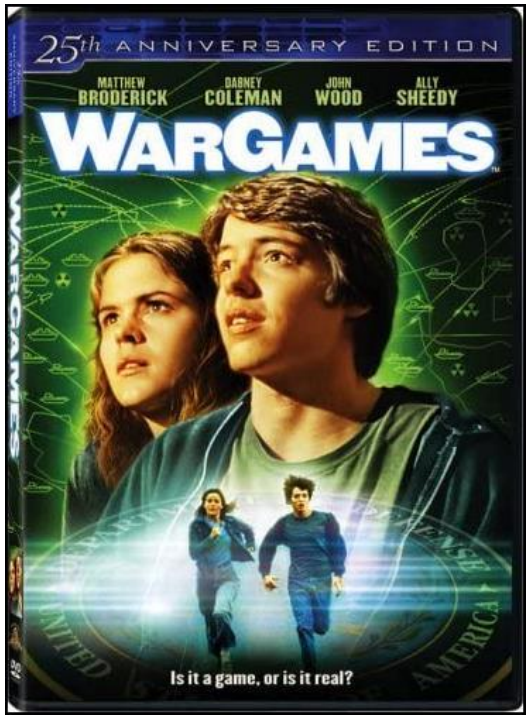
**malware**
Safety score: 11.1%

Compact

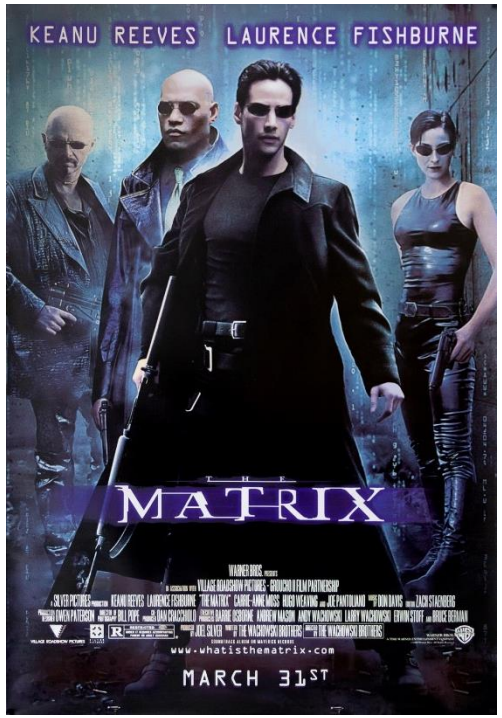
Print results

| Antivirus | Version | Last Update | Result |
|---|---|---|---|
| AhnLab-V3 | 2011.04.19.01 | 2011.04.19 | Dropper/Cve-2011-0611 |
| AntiVir | 7.11.6.177 | 2011.04.19 | EXP/CVE-2011-0611 |
| Antiy-AVL | 2.0.3.7 | 2011.04.18 | Exploit/SWF.CVE-2011-0611 |
| Avast | 4.8.1351.0 | 2011.04.18 | SWF:CVE-2011-0609-C |
| Avast5 | 5.0.677.0 | 2011.04.18 | SWF:CVE-2011-0609-C |
| AVG | 10.0.0.1190 | 2011.04.18 | - |
| BitDefender | 7.2 | 2011.04.19 | - |
| CAT-QuickHeal | 11.00 | 2011.04.19 | - |
| ClamAV | 0.97.0.0 | 2011.04.19 | - |
| Commtouch | 5.3.2.6 | 2011.04.19 | MSWord/Dropper.B!Camelot |
| Comodo | 8396 | 2011.04.19 | UnclassifiedMalware |
| DrWeb | 5.0.2.03300 | 2011.04.19 | Exploit.Wordbo.12 |
| Emsisoft | 5.1.0.5 | 2011.04.19 | Exploit.SWF.CVE-2011-0611!IK |

# 恶意软件 之 文艺洗脑




1983年6月3日




1999年3月31日


2020年8月12日

# 本章要点

- 恶意软件
  - 病毒
  - 蠕虫
  - 僵尸网络
  - 其他恶意软件
- 代码变形
  - Polymorphism
  - Metamorphism
- 缓解方法
  - 签名扫描
  - 动态扫描