第 32 卷 第 9 期 计 算 机 工 程 2006 年 5 月
Vol.32 № 9 Computer Engineering May 2006
·安全技术· 文章编号：1000—3428(2006)09—0151—03 文献标识码：A 中图分类号：TP309.5

一种基于行为分析的反木马策略

商海波[1]，蔡家楣[1]，胡永涛[2]，江 颉[1]
(1.浙江工业大学软件开发环境重点实验室，杭州 310032；2.公安部第三研究所，上海 200031)

摘 要：阐述了反木马策略的研究现状及其发展趋势——木马的行为分析，分析并归纳了木马在安装阶段的行为特征，并在该基础上提出了一种针对 Windows 木马的反木马策略，可在木马安装阶段有效地拦截与查杀木马。
关键词：木马；安装；行为特征；拦截；查杀

An Anti-trojan Strategy Based on Behavioral Analysis

SHANG Haibo[1], CAI Jiamei[1], HU Yongtao[2], JIANG Jie[1]
(1. Key Lab of Software Development Environment, Zhejiang University of Technology, Hangzhou 310032;
2. 3rd Research Institute of Public Security Ministry, Shanghai 200031)

【Abstract】 This article expounds current research condition and development trend of anti-trojand strategy——behavioral analysis of Trojan, analyzes and summarizes trojans' behavioral characteristics of their installation, and puts forward an anti-trojan strategy at Windows platform, which can block and remove trojans effectively during their installation.
【Key words】 Trojan horse; Installation; Behavioral characteristic; Block; Remove

1 反木马策略的研究现状及发展趋势

现有的反木马策略可分为两类：

(1) 通过监控发现网络通信的异常，阻断木马的网络通信。

采用这类策略的有防火墙、入侵检测系统等，它们对通信端口及网络连接作了严格的限制和严密的监控，以“天网”防火墙为例，它能发现并拦截任何未经用户允许的网络连接或者通信端口的使用，并向用户报警。此外，入侵检测还能够自动探测网络流量中潜在的入侵和攻击。

但是，目前许多木马趋向于采用无连接的网络通信协议，并且采取一定的技术使通信端口不易被发现，甚至能做到无端口通信，同时，对通信流量也作了控制，所以使用该策略已经越来越难检测和阻断木马的通信。另外，目前的防火墙、入侵检测系统等一般都不内置检测病毒、木马和蠕虫的模块，在发现网络通信的异常后，无法进一步查找异常的来源。

(2)利用木马特征码检查文件，判断并清除木马。

目前，采用这类策略的主要有基于特征码的静态扫描、虚拟机[2]和实时监控[2]。

这类策略无法检测特征码未包含于特征码库中的木马，即使是已知木马，也可通过加壳或者变形等技术躲避该类策略的检测，使木马服务器有机会进行与客户端的通信，带来信息泄露、系统破坏等损失。另外由于新木马及各类木马变种产生的速度越来越快，特征码的数量也迅速增加，使用这类策略必然需要更大的时间开销，导致检测的效率不断下降。

为克服这些缺陷，许多国内外的研究人员开始从行为角度考虑应对方法，即根据程序的行为特征(如修改注册表等)判断其是否可疑，这种方法也称行为分析。McAfee 公司在它的一份白皮书[3]中指出行为分析将成为基于特征码查杀方法的强有力的补充。某些知名的反木马软件(如“木马猎手”)的开发者也正在考虑在软件中加入行为分析模块。但同其它较为成熟的反木马技术相比，使用行为分析的反木马技术的研究很大程度上仍停留于理论阶段。

2 在木马安装阶段拦截与查杀木马

木马入侵计算机的过程可分为 3 个阶段：(1)木马被投放到计算机；(2)木马程序运行，安装木马服务器；(3)木马服务器与木马客户端进行通信。

选择木马安装阶段拦截与查杀木马是因为：

(1)木马在通信阶段的行为主要包括发送数据和接收数据，而这些行为是许多正常的网络通信程序(如聊天程序、邮件发送程序、浏览器等)都具有的，并不适合作为木马区分正常程序的行为特征。而木马在安装阶段具有显著的不同于一般正常程序的行为特征，比在通信阶段更容易辨别；

(2)在该阶段，木马行为作用的主要对象之一是木马程序本身，可在拦截木马行为的同时，获得木马程序的信息，如程序文件名、文件路径等，从而准确定位木马，并将其清除；

(3)可拦截木马安装时的行为，如修改注册表、修改系统文件等，从而保护这些重要的文件不被破坏，不必在清除木马的同时再对这些文件进行修复。

而“拦截”具体是指对系统中所有当前正在运行的程序进行监控，一旦发现程序具有木马的行为特征，便阻止该程序继续运行；“查杀”则是指根据拦截过程中收集的信息定位与程序相关的文件，对其隔离或者清除。

2.1 木马在安装阶段的行为特征

木马的安装通常有两个步骤：(1)隐藏木马程序； (2) 进行木马服务器的自启动设置[4]，使木马服务器在计算机每次开机或者重启时都自动运行。

如果从木马行为特征的角度观察，可以归纳出上述两个步骤的各种实现途径对应的木马行为及行为作用的对象，如

作者简介：商海波(1981—)，男，硕士生，主研方向：计算机网络安全等；蔡家楣，教授；胡永涛，硕士；江 颉，讲师
收稿日期：2005-05-09 **E-mail：**bobo@trimps.ac.cn

表 1 所示。

表 1 木马安装阶段的行为特征

步骤	实现途径	木马行为	行为作用的主要对象
隐藏木马程序	将木马程序拷贝到系统目录下	拷贝文件	木马程序、系统目录路径
进行木马服务器自启动设置	修改注册表(例如在注册表设置木马服务器的启动项)	打开注册表项、读写注册表项、关闭注册表项	木马程序的路径、注册表
	修改系统文件 win.ini、system.ini 等	打开文件、读写文件、关闭文件	木马程序的路径、系统文件
	将木马服务器程序拷贝至自启动目录下	拷贝文件	木马程序、自启动目录路径
	将木马服务器程序捆绑到系统现有的正常的自启动文件	打开文件、读写文件、关闭文件	木马程序、系统现有的自启动文件
	拷贝 Autorun.inf 文件至某根目录路径下	拷贝文件	文件 Autorun.inf、某根目录路径
	将木马服务器注册为系统服务	注册系统服务	木马程序的路径

以上关于木马安装阶段行为特征的描述，是在对大量已知的流行木马的了解和分析的基础上得出的，是对绝大多数木马在该阶段行为特征的归纳。

2.2 Windows API 钩子技术在本策略中的应用

木马的上述行为在木马程序上表现为不同的 API 调用，所以，如果能拦截这些 API 调用，就达到了拦截木马的目的。

利用 Windows API 钩子技术[5]可拦截 API 调用——预先自定义一个钩子函数，并在系统安装针对某个 API 调用的钩子，便可在真正 API 调用发生前，先调用钩子函数。而在钩子函数内部可以预先自定义逻辑流程，使其结合截获的函数参数作出判断，结果可能是调用真正的 API、退出程序或者执行其它代码。而且，钩子能够作用于系统中所有的进程和线程。所以，该技术可在本策略中应用。

以拦截“拷贝程序到自启动目录”的行为为例：

Windows 平台下实现文件拷贝的 API 可以用伪代码表示为 COPY(SOURCE，DEST)，其中 SOURCE 表示现有文件的路径和文件名，DEST 表示拷贝所得的新文件的路径和文件名。

定义一个与 COPY(SOURCE,DEST)具有相同名称和参数表的钩子函数，并在系统安装针对 COPY 的钩子。之后，任何进程或者线程对 COPY 的调用都会被拦截，实际调用的是钩子函数，而且 SOURCE 和 DEST 两个参数也被钩子函数截获，若 DEST 中的文件路径为自启动目录(很可能是木马程序为了实现自启动，试图将木马服务器程序拷贝到自启动目录下)，则通知系统当前的用户，若用户确认为异常，便不再进行真正的 COPY 调用，甚至终止程序的运行，从而达到“拦截”的目的；另外，SOURCE 中包含异常程序的完整路径和文件名，可用它准确定位文件，并隔离或者删除该文件，从而达到“查杀”的目的。

对其它木马行为的拦截过程也与此类似。

2.3 策略实施过程

本策略在实施过程中，要用到两个关键模块：

(1) 木马行为特征数据库(简称“行为特征库”)。它存储的信息主要包括各行为对应的 API 名称、行为作用的对象和关于行为的描述信息及警告信息。以“拷贝文件到自启动目录”的行为为例，上述 3 项信息分别为 COPY、自启动目录的路径和这样的文字描述“有程序试图拷贝文件到自启动目录下，这很可能是一个木马，是否拦截？”。对所有其它的木马行为特征都可作类似的整理和归纳，并存入数据库。这样便于直接查询，也便于添加新的行为特征。

(2)行为拦截模块。它主要由拦截各个木马行为的 API 钩子组成，分为 3 个子模块：

1)注册表保护模块

该模块拦截所有修改注册表的行为，并向用户报警。这是由于很少有正常软件或者程序会修改注册表，而用户直接修改注册表的情况更是极少发生，也就是说，如果有程序即将修改注册表，那么它很可能是木马。

2)文件保护模块

系统中存在着大量的文件和文件目录，其中绝大多数不能被木马用来实现隐藏和自启动，而且和正常的注册表操作相比，正常的文件操作要频繁得多，所以，文件保护模块不仅判别出即将发生的文件操作行为，还将行为与其作用的对象相联系，只有当行为作用的对象可被木马用来实现隐藏或者自启动(如系统文件、系统文件目录、自启动目录等)时，才拦截该行为，并通知用户。

3)系统服务保护模块

注册系统服务的操作极少在正常软件或程序中出现，所以，系统服务保护模块拦截所有注册系统服务的行为，并向用户报警。

行为拦截模块的各个子模块相互独立，可方便地添加新的行为拦截子模块，因而具有较强的灵活性和较好的扩展性。

如图 1 所示，系统每次启动时，行为拦截模块将拦截木马行为的 API 钩子安装到系统，然后，各个钩子读取行为特征库中的信息进行初始化。

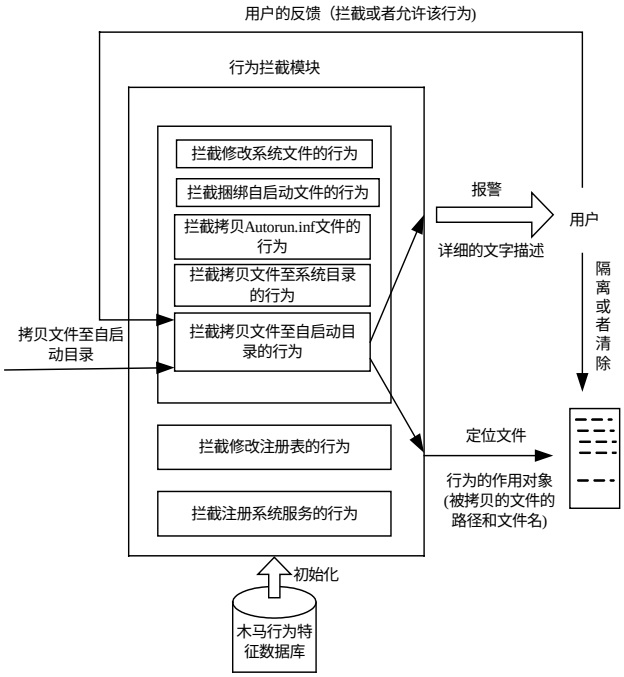

图 1 在木马安装阶段拦截和查杀木马的反木马策略的实施过程

行为拦截模块一旦发现任何进程或线程具有木马的行为特征，便立即通知用户，通知中包含详细的文字描述和警告信息(由于这些行为在正常程序中均极少出现，通知都会建议用户拦截这些行为)，因而用户作出判断并不困难。该模块根据用户的反馈进行下一步操作：

(1)若用户确认行为异常，则阻止该行为，暂停异常程序，并根据截获的参数准确定位与程序相关的文件，再次询问用户，根据用

户的反馈，隔离或者删除该文件，最后终止异常程序；

(2)若用户允许该行为(如某些防火墙的安装程序也修改注册表、用户正在拷贝文件到自启动目录等)，则钩子直接调用真正的 API，不再做其它操作。

2.4 本策略的优点

本策略能阻断木马的安装，保护重要的系统文件、注册表等不被破坏，而且，在拦截木马后能进一步查杀木马，这些都是网络监控策略无法做到的。另外，现有的特征码修改技术[2]如变形、多态等都已经发展得较为成熟，而且具体应用时又有很多变化，而木马安装阶段的程序隐藏及自启动设置的实现途径有限，不同于特征码修改往往只需要考虑木马本身，它需要结合具体操作系统的特性，某些在旧版本的操作系统中可用的技术在新版本中便会失效，所以新的程序隐藏或者自启动设置途径的发现有更大的技术难度。因此，从行为特征角度讲，木马的“行为特征库”比“特征码库”变更的频率要小得多，更具有稳定性。

同时，由于木马安装阶段的程序隐藏及自启动设置的实现途径有限，导致大量木马在这个阶段的行为具有很大的相似性，甚至完全相同，有人对 22 种流行木马及其不同的版本(共计 42 个木马)作过分析[5]，其中，16 种(共计 25 个)木马都通过修改注册表实现自启动，3 种(共计 9 个)木马修改系统文件 system.ini 或者 win.ini，1 种(共计 2 个)捆绑文件；而木马的特征码则一般是某一项特征码只对应某个木马，这也导致了木马的行为特征在数量上比特征码少得多。

基于以上两点原因，木马的“行为特征库”更容易被归纳总结出来，同时，由于不会像“特征码库”那样占用大量空间并且频繁增长，因此更容易维护，开销更小，用在反木马策略中检测木马的效率也更高。

木马特征码与其行为之间没有必然的联系，不管其特征码是否已知，只要其行为包含于现有的“行为特征库”中，就会被本策略拦截。尤其是新出现的木马和特征码改变的已知木马，其特征码未知，使用实时监控、特征码扫描等策略是无法检测的；但由于新的隐藏和自启动设置途径的发现有很大的技术难度，木马通常不得不使用已知途径中的一种或者几种，也就是说其行为特征往往是已知的，因此能被本策略成功地拦截和查杀。同理，对某些隐藏通信端口、无连接的木马，使用网络监控很难发现时，却很可能根据行为特征检测出这些木马。

3 结束语

当前的木马种类层出不穷，使得反木马工作越来越复杂，单靠一两种反木马的策略已越来越难应对，只有把各种不同的策略综合运用，从各个不同角度拦截或者查杀木马，才能够最大限度保证网络或者个人计算机免受木马的侵扰。本文从行为分析的角度提出了在木马安装阶段拦截和查杀木马的反木马策略，它可作为现有策略的有力补充，构成更加牢固的反木马防线。

参考文献

1 金山公布 2004 年度十大“木马”病毒[EB/OL]. www.bjbusiness.com.cn, 2004.

2 Hume. 病毒和网络攻击中的多态、变形技术原理分析及对策[EB/OL]. http://www.xfocus.net/projects/Xcon2003_hume.pdf, 2003.

3 McAfee. Best Behavior-making Effective Use of Behavioral Analysis [M]. Network Associates, 2002.

4 与特洛伊木马过招[EB/OL]. http://member.netease.om/~netsurfe/inet/safe_tlym.htm, 2000.

5 Ivanov I. API Hooking Revealed[EB/OL].www.codeproject.com, 2002.

（上接第 150 页）

(3)如果检测到违背规则的情况则作出相应处理。

检查模块与解释器、目标代码的关系以及执行流程如图 2 所示。

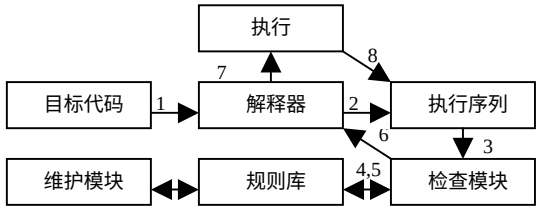

图 2 检查模块与解释器、目标代码的关系以及执行流程

图 2 中各流程的含义如下：

流程 1~2：目标代码通过解释器生成执行序列；

流程 3~5：检查模块根据规则库对当前操作进行检查；

流程 6~7：通过检查的操作送入解释器执行；

流程 8：当前操作执行结束，进入下一个流程 3~8 的循环中。

5 小结

本文提出了对代码进行动态检查的模式识别方法，主要解决时序安全特性和竞争条件两类问题。通过对代码与规则进行实时匹配、检查，动态地检测代码的执行是否存在（潜在的）安全问题。并且给出了该方法在 Perl 解释器中的实现。

使用动态方法对代码进行检查，可以很好地解决静态方法无法解决的问题：

(1)动态方法的一个明显优势就是更为准确、时效性好。动态地进行检查可以掌握更多更准确的信息，这些实时的信息都是静态检查所无法掌握的，因此可以作出更准确的判断；

(2)由于是实时检查，因此，不会存在未及时对程序做出检查的问题；

(3)因为在动态检查中过程内与过程间没有区别，动态检查关注的只是程序的执行过程，所以可以很好地对过程间规则进行检查；

(4)动态检查也不会受到无限循环问题的困扰。即便程序因为某些原因而进入无限循环状态中，也不会影响到对其他程序的检查，因为解释器总是为每个程序分配一个独立的子解释器，所以对各个程序的检查也是相互独立互不影响的。

今后的工作主要包括两方面：进一步丰富规则库，使得检测可以覆盖更多的安全问题，降低漏报率；进一步完善规则库，提高模式描述的准确性，降低误报率。

参考文献

1 Ball T, Rajamani S K. Automatically Validating Temporal Safety Properties of Interfaces[C]. Proceedings of the 8th International SPIN Workshop on Model Checking of Software, 2001:103-122.

2 Chen Hao, Wagner D. MOPS: An Infrastructure for Examining Security Properties of Software[C]. Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 1992-11.

3 Engler D, Chelf B, Chou A, et al. Checking System Rules Using System-specific, Programmer-written Compiler Extensions[Z]. OSDI, 2000.