文章编号:1001 - 9081(2010)01 - 0207 - 03

改进的进程行为检测模型及实现

唐彰国,李焕洲,钟明全,张　健
(四川师范大学 网络与通信技术研究所,成都 610066)
(tangzhangguo@ sicnu. edu. cn)

摘　要:为了检测恶意程序,分析了现有各类检测机制的不足,重新界定了进程行为概念的外延,提出了差量对比与进程动态行为分析的检测模型,给出了关键技术和实现方法。测试结果表明该检测模型在通用性和有效性方面优于传统检测方法。

关键词:恶意程序;差量对比;检测模型;API 函数
中图分类号: TP309　　**文献标志码**:A

Improved model of process behavior detection and implementation

TANG Zhang-guo, LI Huan-zhou, ZHONG Ming-quan, ZHANG Jian
(*Institute of Computer Network and Communication Technology, Sichuan Normal University, Chengdu Sichuan* 610066, *China*)

Abstract: To detect malicious program, the disadvantages of current detection mechanism were analyzed. The extension of process behavior concept was redefined. A detection model of difference comparison and process dynamic behavior analysis was proposed. The critical technology and realization were given. The experimental results indicate that the detection model excels traditional detection method in versatility and effectiveness.

Key words: vicious procedure; difference comparison; detection model; API function

0　引言

当前,恶意程序(如木马等)越来越泛滥。恶意程序使用的技术主要有:用户模式系统调用劫持、核心模式系统调用劫持、核心模式数据篡改以及核心模式中断处理程序劫持。恶意程序检测方法分为静态检测和动态行为检测。其中动态行为检测方法主要有挂钩函数检测、隐藏进程检测、可执行路径分析检测(EPA)、交叉检查差量检测和内存完整性检测。目前,对进程检测的研究国内外主要集中在进程行为的实现上。例如挂钩函数检测法通过检测系统内存中操作系统模块和进程中被挂钩的函数实现;执行路径分析法则检测某些关键系统函数执行时所用的指令个数来判定系统中是否存在恶意程序;而交叉检查差量检测的原理是比较操作系统不同层次 API 函数的查询结果是否一致来判断是否存在 API Hook[1-3]。

普通的基于进程动态行为特征的恶意程序检测技术存在一定局限。如文献[4]提出的基于 APC 机制的远程线程注入技术实现的进程隐藏由于未对内核对象等系统敏感信息进行修改(未挂钩系统服务函数、修改服务函数执行路径),使得系统完整性检测、EPA 执行路径分析、挂钩检测等检测技术失效;文献[5]认为基于截获系统调用(Hook System Call,HSC)的进程隐藏检测技术能完整且可靠地建立进程列表,但绕过这种检测方法也很简单,如改用其他中断或是用全局描述符表中的调用门。可见,由于恶意程序的种类多,技术差异大,各种专门的检测工具很难统一成普适的检测模型,其原因是由于 Windows 是分层实现的,而 Windows 程序是事件驱动并主要基于消息投递的。因此,信息的获取和消息的拦截可在操作系统的不同层次和不同路径上实现。正是因为这个原理,恶意程序与检测工具在技术上表现出有针对的对抗性,使得以上类型的检测技术不能面面俱到,无法构筑一个全方位的进程行为监控体系。

综合分析以上进程动态行为检测技术发现,共同的缺点是把检测的重点只放在进程行为的过程上,而忽视了行为结果以及进程状态变化等相关信息,从而就导致了漏检率和误检率较高。本文将重新审视进程行为的定义,并据此提出一种改进的进程行为检测模型:差量对比与进程行为动态分析。

1　进程行为的定义及模型

1.1　进程行为模型

进程行为由主体、客体、操作集合、行为输入、行为输出、行为状态和行为属性等组成。以进程生命周期的全局来看,可从如下三个方面对进程行为加以界定。

1)进程行为的内涵:操作系统内核把系统调用看作是进程的行为。在基于行为的进程检测技术中,进程被看作一系列系统调用的有序组合。

2)进程行为的外延:一个行为与另一个行为的区别在于进程行为的四个要素(主体、客体、所使用的操作以及状态的迁移)。主体指进程本身,客体指主体操作的对象及使用的资源,操作是指系统(API)调用。状态的迁移指进程行为引起进程状态的变化。

3)进程行为的属性:按时序度量,分为行为创建、行为过程和行为结果。

文献[6]对进程行为进行了定义,但没考虑进程状态及进程时间属性等重要信息,本文改进如下:

收稿日期:2009 - 07 - 09;**修回日期**:2009 - 08 - 19。　**基金项目**:四川省应用基础研究项目(07JY029-011)。

作者简介:唐彰国(1978 -),男,广西桂林人,讲师,硕士,主要研究方向:信息安全;　李焕洲(1974 -),男,四川阆中人,副教授,博士,主要研究方向:网络监控、可信计算;　钟明全(1975 -),男,四川内江人,讲师,硕士,主要研究方向:网络通信、信息安全;　张健(1975 -),女,四川宜宾人,讲师,博士研究生,主要研究方向:网络安全。

定义 1 进程行为：

ProActions = { action = s applies (f) to (obj) :S→S′| t }

其中：s 表示行为的主体；f 表示施用函数；obj 表示行为的客体；S 表示进程状态；S→S′表示状态的迁移；t 表示时间域。进程行为模型可理解为：在生命周期任意一时间点上，主体期望对客体施加的操作并由此引起的进程状态迁移。

1.2 进程行为的检测对象

根据进程行为的定义，进程行为检测的对象是某主体的行为轨迹，即以发生的时间顺序用串的格式刻画出的同一主体从事的行为集合，包括生命周期中发生的系统调用、进程从创建到终止的状态迁移、进程访问的对象以及引起系统资源的变化。

定义 2 检测对象：

ObjToDet = {(S,R,F,A)| P}

它包括状态因子 S、资源因子 R、操作因子 F、属性因子 A 和程序因子 P。状态因子表示进程所处的状态，资源因子表示进程占用的系统资源，如通信端口、CPU 等，操作因子指进程的 API 调用，属性因子表示进程行为时间、出现的频率等信息。程序因子表示启动进程的程序，即进程的运行主体。

2 差量对比与进程动态行为分析的检测模型

2.1 检测思想

恶意程序与合法程序的区别就在于行为的隐蔽性，包括植入、启动、运行和通信环节。但恶意程序在运行时会使得目标系统的文件、文件夹、注册表、函数等与运行前发生变化，在通信时会使得网络流量、CPU 及内存等系统资源使用率急剧提高[7]。以上种种变化说明了恶意程序的执行会导致目标系统与没有恶意程序时的某些特征具有了差量。这种差量可以用来判断恶意程序的性质及功能。

定义 3 差量对比法。启动可疑程序后，实时监测系统的某些特征变量，依据前后的差量来界定恶意程序的功能和行为性质，即合法性。

差量对比法主要界定进程行为的两个要素，即恶意进程的客体对象以及行为的结果。然而，对于内核级的进程隐藏和远程线程技术，它们的行为结果在宏观表象上可能不会出现任何异常，如新一代的木马启动后无进程、无端口、无 DLL、无文件、无启动项等，从而导致差量对比法出现漏检。这就需要进一步检测进程动态行为另外两个要素：进程行为的执行过程及进程状态的迁移。

定义 4 进程动态行为分析。基于可疑程序的行为识别，将某主体启动后的行为轨迹以发生的时间顺序用串的格式刻画出来，以此来分析可疑文件的功能和行为性质，即合法性。

根据进程行为的定义，这里提出一种改进的进程行为检测模型：差量对比与进程动态行为分析。其中，差量对比采用了黑箱思想，不关心进程行为的内部过程，只关心行为的输出和结果，这在原理上屏蔽了技术的差异，增强了模型的通用性。进程动态行为分析采用白箱思想，与前者互为补充。该模型的优点是，尽管各种恶意程序采用不同技术和思路，但最终的目的和期望达到的效果是不变的[7]，据此来检测恶意程序的行为和功能是抓住了恶意程序的本质属性，不易被恶意程序的各种检测对抗技术所绕过。

2.2 模型架构

该检测模型的工作原理如图 1 所示。首先准备一个干净的检测环境，让被检测程序在检测环境中运行并实时监视：监视其对检测环境的影响并与原始环境进行差量对比；监视其行为的过程以及行为产生的结果。然后对两种方法得到的数据进行行为层次的综合并输出恶意行为报告。

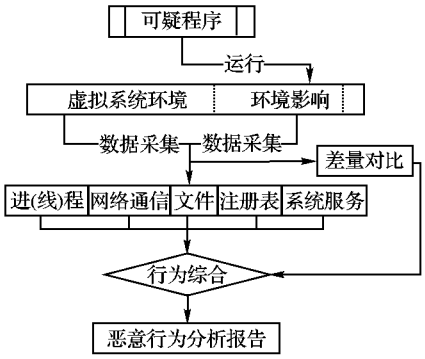

图 1 模型架构

3 关键技术

3.1 差量对比技术

差量对比法首先准备一个干净的虚拟检测环境，在恶意程序进程启动前对虚拟检测环境的各项特征进行快照，这些特征变量包括注册表的项、键值、数据，系统文件的数目、大小、创建时间，系统目录内文件的数目、大小、创建时间，系统服务的数目，硬件的各项指标等。然后启动恶意程序并实时监控其对注册表的访问，对系统目录、系统文件和系统服务的改变，网络通信流量及硬件资源占用率的变化等，进一步提取系统特征的差量作为后期判断和处理的依据。

3.2 进程动态行为分析技术

恶意程序的各种行为在程序代码上表现为不同的 API 调用，检测出这些 API 调用即可识别出对应的动态行为。为了有效应对恶意程序的各种模式劫持方式，检测模型中的各个模块均采用了相应的内核态驱动开发技术。

进程的动态行为分析模块关系如图 2。其中，可疑程序行为轨迹检测按照进程行为的定义检索出以可疑程序进程名（或 PID）为主语的按时序先后归纳的行为集合。其叙述方式形如：a. exe 启动后隐藏了自身进程，启动了 2 个新进程 b. exe 和 c. exe，创建了远程线程并注入到 d. exe，受感染的进程 d. exe向外发起了 5 次连接并发送了电子邮件。恶意行为分析主要总结为以下几类：对文件的异常访问行为，如自删除行为、创建压缩文件、不断复制文件、删除或替换非自有文件等；对存储器的异常访问行为；对网络的异常访问行为，如反向连接、端口复用、发送电子邮件或使用 ICMP 协议等；对应用程序的恶意控制；对系统配置的更改行为，如自启动项、新增系统服务等；对资源的恶意占用行为；对自身的隐藏行为，如创建远程线程；与安全软件的对抗行为，如杀死防火墙和杀毒软件。

3.3 虚拟环境技术

1）虚拟干净检测环境：通过构造虚拟计算机的寄存器表、指令对照表和虚拟内存，让待测程序在虚拟机中运行一段时间，使该程序完成其全部行为动作，记录其对系统环境的影响，从而完成差量检测。具体实现时可采用虚拟机软件（如 VMware）及其快照恢复功能。需要注意的是，为了避免对差量对比结果的干扰以及由此带来的误报问题，检测前需对操作系统和应用程序进行相关设置，例如禁止操作系统和应用软件的自动更新、自动升级以及定时自动杀毒和备份等功能。

2）行为诱发技术：恶意程序为了隐藏和躲避查杀，通常

会遍历当前活动进程,找出并杀死各类安全软件进程,如杀毒软件、防火墙、IDS 及其他专用的检测工具等。为了触发恶意程序的这种行为,需要在活动进程列表中出现这些安全软件进程。但一台主机往往不能同时安装多个杀毒软件和防火墙,解决这个矛盾的方法是虚拟出各种安全软件进程并在运行恶意程序前启动。这些虚拟的进程仅仅与真实的安全软件进程同名而已,内部不执行任何有意义的操作,只作为触发恶意程序行为的诱饵。

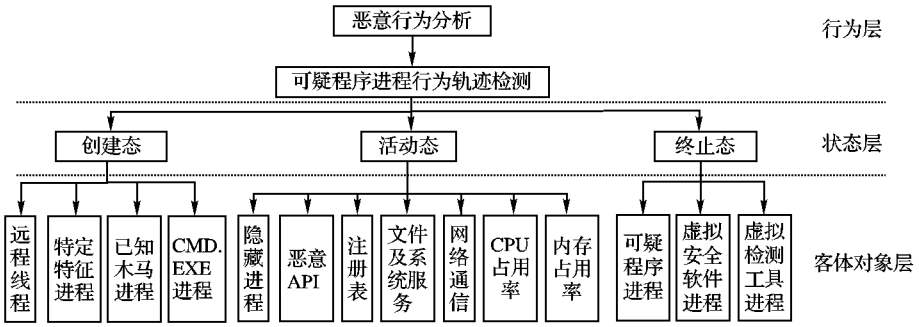

图 2　进程的动态行为分析模块关系

3.4　恶意行为识别技术

程序的行为具有模糊特性,即某些行为既属于合法程序的行为特征,也属于恶意程序的行为特征。如修改注册表项、远程线程注入等行为。为降低误报,在检测恶意程序行为特征的同时,还需提取出合法程序区别于恶意程序的行为特征。

另外,差量比较虽具有所谓的“全特征”特性,能有效解决漏报问题。但随着恶意程序本身的不断地发展,某些行为特征可能会减少甚至消失,同时出现新的行为特征。因此各行为特征属性的重要性是存在偏序关系的。

基于程序行为的模糊性和特征属性的偏序性,为有效降低误报率和漏报率,这里采用基于加权的统计行为特征识别技术:实时地对待测程序进行多种行为特征的检测,将多种行为特征联合起来作为判别某程序是否合法的规则,对于出现的各项行为特征通过一定的权系数相加起来,达到一定的阈值后根据设定的策略输出与恶意程序的相似度。

4　测试及结果分析

由于差量对比与进程动态行为分析模型检测的行为特征变量较为全面,限于篇幅,这里只给出了几种典型恶意行为的对比测试结果(如表 1 所示),采用的对比工具为著名的挂钩函数检测工具 VICE 以及隐藏进程检测工具 Klister。

表 1　差量对比与进程行为动态分析检测模型通用性对比测试结果

被检测程序	本文模型					VICE					Klister				
	进程隐藏	杀死进程	设置自启动	挂钩函数	发电子邮件	进程隐藏	杀死进程	设置自启动	挂钩函数	发电子邮件	进程隐藏	杀死进程	设置自启动	挂钩函数	发电子邮件
熊猫烧香	有	有	有	有	无	无	无	无	有	无	有	无	无	无	无
灰鸽子 2009VIP 版	有	有	有	有	有	无	无	无	有	无	有	无	无	无	无
工行钓鱼木马	有	无	有	有	有	无	无	无	有	无	有	无	无	无	无
He4Hook 内核级木马	无	无	有	有	无	无	无	无	有	无	无	无	无	无	无

从表 1 中可以看出,相比传统的进程行为检测工具本文模型检测的特征要素更为全面,能够有效应对恶意程序的各种检测对抗技术。以“熊猫烧香”为例,该恶意程序会终止几乎所有杀毒软件进程和 IceSword 等安全分析工具进程,修改注册表键值导致不能查看隐藏文件和系统文件,删除杀毒软件在注册表的启动项和服务。“熊猫烧香”的这些行为又可分为两种:一是其他恶意程序同样具有的;二是特有的。对于第一种行为本文模型中的进程动态行为分析模块由于对恶意行为的归纳较为全面能给出准确的报告,对于第二种行为差量对比模块发挥出了“技术无关性”的优势。

表 2 给出了模型的误报率测试结果,其中合法程序被判定为与恶意程序高相似度的个数为 0,这表明该模型能够有效区分合法程序和恶意程序。其原因是模型考虑了合法程序区别于恶意程序的行为特征,如呈现可视化界面。合法程序被判定为“中”的含义是程序出现了自删除(如 Word 关闭后会删除其创建的临时文件)、自启动(如杀毒软件在注册表中创建自启动项)和远程线程(如金山词霸的取词功能)等行为,而这些行为还不足以成为判定其为恶意程序的充要条件。模型中的恶意行为识别算法通过合理地设置各种行为的权值有效解决了可能由此到来的误报问题。

表 2　差量对比与进程行为动态分析检测模型误报率测试结果

被检测程序	与恶意程序相似度的个数分类统计		
	高	中	低
100 个常用合法程序	0	8	92

可见,差量对比与进程动态行为分析两种方法互为补充,共同组成了一个全方位的进程行为监控体系,使得该模型具备了通用性好、有效性强等特点。

5　结语

针对现有恶意程序检测技术的不足,本文重新界定了进程行为概念的内涵和外延。在此基础上提出了差量对比与进程动态行为分析的恶意程序检测模型。测试结果表明该检测模型在通用性和有效性方面优于传统检测方法。

下一阶段的工作中需进一步研究的问题有:1)进一步提高检测的效率。由于进程行为的客体对象较多,检测的时间开销较大。2)恶意行为的可视化输出。本文的原型系统实现了文本方式的恶意程序进程行为轨迹描述,为了更直观更具可读性,后期改进拟采用图形化的方式标识出进程行为的时间属性和进程间的逻辑关系。

(下转第 223 页)

是加入树的引入。当成员加入时,加入树充当类似于缓存区的作用。由于加入树较小,成员加入加入树显然需要较小的开销,而批量更新开销平均到每一个成员来说显然是很小的。在 WJT 和 JET 方案中,因为总的移动开销被分摊到每个成员,因此每个成员的平均批量移动开销是恒定的。而 WJT 通过将成员加入加入树根部,进一步缩短了成员加入加入树的时间,而在 JET 中,成员总是被插入加入树中深度最浅的最右边节点,因此所需的加入开销大于 WJT 方案。

3.2　有成员正确离开信息时的组密钥更新

在本节,假设成员连续离开安全组,并可以获得成员在组中的离开概率。当主树的规模大于 16 时,离开树被激活,而在 JET 中,离开树在主树大于 256 时被激活。图 4 比较了三种方案 WJT、JET 和 TGDH 在成员连续离开时所需的平均离开时间。

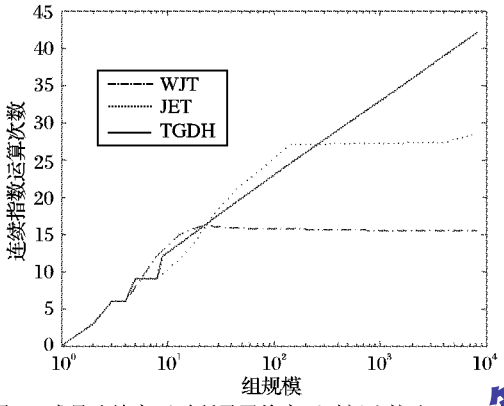

图 4　成员连续离开时所需平均离开时间比较 ($\rho = 0.5$)

从图 4 中可以看出,WJT 与 TGDH 和 JET 在组小于 4 时获得同样的性能。当组规模大于 4 时,WJT 中的加入树被激活时,WJT 的性能先优于 TGDH 和 JET,然后低于 TGDH 和 JET,最后在组规模大于 16 时优于它们。这主要是由于在离开树激活前,加入树的激活首先降低了密钥树的高度,然后由于相对较小的加入树容量增加了密钥树的高度。对于 JET,情况也是相同的。同时也注意到 WJT 的平均离开时间并没有随着组规模的增大而增加,而其他两个方案都随着组规模的增大而逐渐增大。这就证实了在成员动态信息可知时,通过离开树的引入,WJT 在成员离开时密钥更新的渐近性能上界为 $O(1)$。

引入离开树作为一个临时缓存区是 WJT 和 JET 相比 TGDH 方案获得较好性能的一个原因。在 WJT 和 JET 方案中,成员离开的开销包括两部分:从主树移动到离开树的开销和从离开树离开的开销。批量移动的开销在两种方案中都是恒定的,因为批量移动开销被分配给了每个移动的成员。但是通过使用权重离开树,WJT将成员从离开树离开的开销降低为常数,而在 JET 方案中,成员离开的开销为 $O(\log(\log n))$。

4　结语

本节在 JDH 方案的基础上,引入了基于权重离开树的贡献型组密钥管理方案 WJT 来获得较好的成员离开时的密钥更新性能。WJT 利用加入树和离开树充当新加入成员和即将要离开成员的暂时缓存区,主要包括加入算法和离开算法。加入算法与 JDH 方案中的加入算法相同,将新成员加入到加入树根部,当成员从加入树移动到主树中时,将可获得的新成员的离开概率加入离开队列。离开算法是假设成员在组中的离开概率可以预先获得。当批量移动条件满足时,将那些即将要离开的成员从主树移动到离开树。由于批量移动的开销依赖于被移动成员的数目,而不是离开树的结构。因此将离开树根据成员的离开概率组织为一个权重树,它的高度与所包含的成员个数相等。改进的算法可以将成员加入和成员离开时的平均密钥更新时间的上界降低为 $O(1)$。仿真和实验说明 WJT 在密钥建立和成员加入时,优于现有方案,在成员动态信息可知时的离开开销也要优于其他方案。

值得注意的是,WJT 的性能在成员的离开概率不确定时会下降。因此 WJT 的应用应该限于那些可以预先获得成员停留时间的组播应用,例如按次付费,远程会议等。

参考文献:

[1] KIM Y, PERRIG A, TSUDIK G. Tree-based group key agreement [J]. ACM Transactions on Information and System Security, 2004, 7(1): 60 – 96.

[2] MAO YILLIAN, SUN YAN, WU MIN, *et al*. JET: Dynamic join-exit-tree amortization and scheduling for contributory key management[J]. IEEE/ACM Transactions on Networking, 2006, 14(5): 1128 – 1140.

[3] GU XIAOZHUO, YANG JIANZU, YU JING, *et al*. Join-tree-based contributory group key management[C]// 2008 10th IEEE International Conference on High Performance Computing and Communications. New York: IEEE, 2008: 564 – 571.

[4] DIFFIE W, HELLMAN M. New directions in cryptography [J]. IEEE Transactions on Information Theory, 1976, 22(6): 644 – 654.

[5] DONDETI L R, MUKHERJEE S. DISEC: A distributed framework for scalable secure many-to-many communication[C]// Proceedings of 5th IEEE Symposium on Computer and Communications Security. Washington, DC: IEEE Computer Society, 2000: 693 – 698.

[6] STEINER M, TSUDIK G, WAIDNER M. Diffie-Hellman key distribution extended to group communication[C]// Proceedings of 3rd ACM Conference Computer and Communications Security. New York: ACM, 1996: 31 – 37.

[7] BECKER K, WILLE U. Communication complexity of group key distribution[C]// Proceedings of 5th ACM Conference on Computer and Communications Security. New York: ACM, 1998: 1 – 6.

(上接第 209 页)

参考文献:

[1] 康治平, 向宏, 傅鹏. 基于 APIHOOK 技术的特洛伊木马攻防研究[J]. 信息安全与通信保密, 2007(2): 145 – 148.

[2] 李焕洲, 唐彰国, 钟明全, 等. 基于行为监控的木马检测系统研究及实现 [J]. 四川师范大学学报, 2009, 32(3): 386 – 389.

[3] FISKIRAN A M, LEE R B. Runtime execution monitoring (REM) to detect and prevent malicious code execution[C]// Proceedings of the IEEE International Conference on Computer Design. Washington, DC: IEEE Computer Society, 2004: 452 – 457.

[4] 何志, 范明钰, 罗彬杰. 基于远程线程注入的进程隐藏技术研究[J]. 计算机应用, 2008, 28(6): 92 – 94.

[5] 何志, 范明钰. 基于 HSC 的进程隐藏检测技术[J]. 计算机应用, 2008, 28(7): 1772 – 1775.

[6] 陈京浩. 进程检测模型及相关技术的研究与实现[D]. 重庆: 重庆大学, 2007.

[7] 米渊. 使用差异比较法对木马程序进行防杀的技术探讨[D]. 呼和浩特: 内蒙古大学, 2008.