基于沙盒技术的恶意程序检测模型

陈丹伟 唐 平 周书桃
(南京邮电大学计算机学院 南京 210003)

摘 要 计算机恶意程序引发的犯罪活动越来越多,因此,恶意程序的有效检测成为了人们研究和关注的焦点。基于沙盒技术的恶意程序动态分析检测方法成为了目前研究的热点。利用改进的QEMU进程虚拟机,以获取更高的仿真响应时间和完整的API序列流为目的,基于改进的攻击树提出了一个行为分析算法,并用实例加以说明。实验结果证明提出的检测方法是可行的、有效的。

关键词 沙盒,恶意程序检测,动态分析

Malware Detection Model Based on the Sandbox

CHEN Dan-wei TANG Ping ZHOU Shu-tao
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract The increasing of computer malware criminal leads researchers to pay attention on the effective detection of malware. The dynamic analysis detection method based on sandbox technology becomes the research spot. This paper proposed a behavior analysis algorithm based on improved attack tree which uses the improved QEMU process virtual machine to obtain a shorter response time and a complete API sequences flow. And the experiment results demonstrate effective and feasible of this detection method.

Keywords Sandbox, Malware detection, Dynamic analysis

1 引言

在当今的信息时代,互联网的发展与广泛应用深刻改变了人们的生活方式,大大加快了人类的工作进程、信息获取,使其自由交流沟通越来越简单。然而,针对这些的攻击和犯罪层出不穷。

在现有的系统中,大多数用来维持安全性的技术通常作为独立的程序来实现,但是在实现时把用来加强安全性和监视系统活动的技术与系统被攻击部分隔离开也很重要,这就是沙盒技术的来源。

现在,传统的恶意程序分析方法已经得到了最大化的研究,基于检测的攻防方法能够快速准确地对已经出现的恶意程序进行检测。但其缺乏一定的灵活性,而且对于未知的恶意程序无能为力。而近些年来,黑客的地下发展、团队协作、智能化等让这种检测越来越无能为力。因此,提出了新的终端防护技术要求,而沙盒技术正是符合此要求的 可以解决这些问题的技术。

2 沙盒技术及恶意程序检测方法

2.1 沙盒技术

沙盒在计算机领域主要是指一个严格受控的环境,程序在其中运行时的状态、所使用或访问的资源都受到严格的记录和控制。沙盒根据指令对全系统进行模拟,对一个程序的访问资源、执行环境按照规则赋予一定的访问权限。这样程序只能在建立的沙盒里面进行相关操作,限制了恶意程序的危害。每个程序在自己的受保护的沙盒之中运行,不会影响到其他程序的运行,同样,这些程序的运行也不会影响操作系统的正常运行。

2.2 特征码法

特征码识别技术借助反汇编引擎对二进制程序进行静态反汇编,对得到的静态代码进行一些关键性的代码判断比较,包含此代码就被识别为恶意程序。特征码法早期应用于SCAN、CPAV等著名病毒检测工具中。国外专家认为特征码法是检测已知病毒的最简单、开销最小的方法。

2.3 基于行为的检测法

它是利用病毒的特有行为特征来监测病毒的一种方法。当程序运行时,监视其行为,如果发现了病毒行为,立即报警,另外行为特征识别通常需要使用类神经网络一类方法来训练分析器,并能够准确地用形式化的方法来定义恶意代码的特征。

2.4 基于特征函数的检测方法

恶意代码要实现特定功能,必要使用系统的API函数(包括内核级和用户级的),因此如果某个程序调用了危险的特定函数集合,我们有理由怀疑其可能是恶意代码。在程序加载之前,对于引入的任何程序文件,我们扫描其代码获得其系统函数集合(这个过程可能需要代码逆向技术和虚拟机配合),并将其与我们对多个恶意代码分析后设置好的一系列特征函数集合做交集运算,这样就可以知道该程序文件使用了

本文受国家自然基金(61073188)资助。

陈丹伟(1970—),男,博士生,教授,主要研究领域为信息安全及计算机应用;唐 平(1986—),男,研究生,主要研究领域为计算机应用;周书桃(1987—),男,研究生,主要研究领域为信息安全。

哪些危险的函数，并大致可以估计其功能和属于哪种类型。

3 基于改进攻击树模型的行为算法分析

3.1 攻击树模型的改进及构建

通过对大量的恶意程序行为的分析，我们总结了恶意程序最常用的 API 调用序列。本节在攻击树模型的基础上进行改进，定义了一种新型的改进攻击树模型，该模型在传统的攻击树的基础上，对每个节点加入了参数集合 *Parameter*(*T*)、权值集合 *Weight*(*T*)，改进的扩展攻击树表示为 $T=\{V,E,Parameter,Weight\}$。其中：

Root(*T*)：根节点，表示最终的行为。

V(*T*)：以与节点和或节点为集合元素的非空有限集合。

E(*T*)：树中边的集合。

Parameter(*T*)：节点参数信息集合，由三元组(*state*，*time*，*pmlist*)组成，其中 *state* 表示当前节点的状态，*time* 表示为节点行为的时间序编号，*pmlist* 记录参数调用列表。此集合与每个行为节点相关联，对于叶子节点，此节点数据在行为获取模块已经实例化，对于非叶子节点，此值为空。

Weight(*T*)：与每个行为节点相关联，由二元组(*danger*，*weight*)表示，其中 *danger* 标识当前的危险级别，*weight* 表示当前的权重。节点的 *danger* 值由子节点的 *danger* 和 *weight* 值计算得到。*weight* 值由其调用的 *API* 以及参数等决定，对于关注的 *API* 行为集合中重点关注的 *API* 不同的参数，其对应的 *weight* 也不同。

我们根据恶意程序中常见的 API 行为序列、序列达成的目的及序列间的依赖关系，依照构建一棵最大改进攻击树(AT)的目的，构建出若干棵改进的最大改进攻击树(AT_1，AT_2，…，AT_n)，它们的集合构成了一个森林 $S=\{AT_1,AT_2,\cdots,AT_n\}$。另外建立一个节点 ROOT，将所有最大改进攻击树的根节点作为 ROOT 节点的子节点，从而将森林组成一棵树，记作 *PT*。作为匹配依据，*PT* 的构造是不变的，称为原始改进攻击树。同一个 API 行为可能出现在不同的行为序列中，达成不同的行为目的。因此在 PT 中，出现在不同子树中的同名节点可以视为“同名但不同义”的节点。叶节点不同的 API 以及参数，其权重 weight 不同；同 API 下，重点关注的高于未重点关注的。初始 danger 置为 1。非叶节点的 danger 由子节点计算而来，初始化为 0。权重 weight 为≥1 的固定值。与节点的 weight 比或节点权重高。所有节点初始化状态都为 false。

3.2 基于改进攻击树的行为分析算法

算法的实现如下：

算法 1 匹配算法 Match(PT，A)

输入：原始改进攻击树 PT，行为集合 $A(T)=\{A_1,A_2,\cdots,A_n\}$。

输出：所有标记的改进攻击树 PT′。

1)i←0；

2)如果 i≥n，节点标记完毕，转 6)；

3)在 PT 中查找 Ai；

4)如果找到：①把树中对应的叶节点标记为高亮，即把 state 参数置为 true，②如果在树中多处出现，则将所有同名叶结点都进行①的操作；

5)i=i+1，转步骤 2，匹配序列中下一个行为；

6)调用 Mark(V)函数对非叶节点进行标记。

Mark(V)函数是一个递归函数，按规则标记所有非叶节点的 state 值，并返回标记后节点 Node 的状态 state(V)。

算法 2 标记算法 Mark(V)

算法思想是：如果是叶结点则直接返回其状态 state(V)，否则先通过递归调用确定的所有下一层子结点 $V_1,V_2,\cdots,V_n$，再根据 V 的属性确定并返回 V 的状态 state(V)。具体实现如下：

输入：以节点 V 为根的改进攻击树 PT

输出：标记以 V 为根的改进攻击树 PT 中所有非叶节点的状态，并返回 state(V)

1)如果 V 是也节点，直接返回该节点的状态 state(V)；

2)如果 V 是与节点，对 V 的所有下一层子节点 V_i 进行与操作。state(V)=state(V) and Mark(V_i)，然后返回 state(V)；

3)如果 V 是或节点，对 V 的所有下一层子节点 V_i 进行或操作。state(V)= state(V) or Mark(V_i)，然后返回 state(V)。

对 PT 的所有节点标记完成后改进攻击树记为 PT′，PT′中每棵高亮子树 T_i 都对应一个恶意行为攻击的完成。

每个程序 T 都对应一个静态危险指数 *D*，记作 *danger*(*T*)，其通过对程序的计算得出。此值可以抽象地量化程序 *T* 与恶意程序的相似程度。此值越高代表危险系数越大，越接近恶意程序。危险指数公式为：

$$danger(Parent)=weight(Parent)*\sum(danger(child)*weight(child))$$

其中，*danger*(*Parent*)：当前 *Parent* 节点的危险值。

weight(*Parent*)：当前 *Parent* 节点给定的权重值。

danger(*child*)：当前节点下所有的子节点值。

weight(*child*)：当前节点下该节点的权重值。

漏报和误报是恶意程序检测系统必然碰到的问题，选择合理的阈值能够有效地提高行为监测反病毒系统的性能，反之不合理的阈值会影响恶意程序检测系统的性能。阈值过高，某些病毒不能被检测出来，会出现漏报；而阈值过低，一般程序也会被判定为病毒程序，出现误报。故需要使用大量白样本和大量的恶意程序样本进行分析，根据计算出来的危险指数分布情况来确定阈值。

4 性能评估

作者对 9917 个恶意程序(原始黑样本)和 9964 个白样本的行为进行了自动化搜集，把所有的实验黑白样本分成独立的两组，分别用于建模训练和测试，如表 1 所列。

表 1 训练测试数据集定义

样本类型	建模	测试
白样本	O'_{2821}	T'_{7143}
黑样本	O_{8223}	T_{1694}

4.1 仿真时间

从仿真时间等方面展开改进的 QEMU 和未改进的 QEMU 的对比，如图 1 所示。

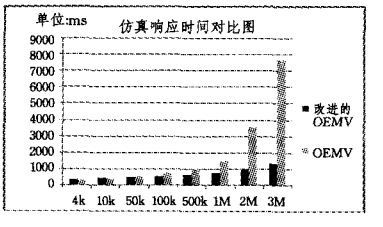

图 1 仿真响应时间对比图

从图 1 可以看出，在仿真很小的程序时，由于改进的 QEMU 有程序预处理等模块，预处理时间加上在仿真初始化额外需要为映射表、索引表申请内存的时间，相对来说比程序翻译仿真时间长，故显示出来的小程序的仿真响应时间大于 QEMU，大概在 40k 时达到平衡。时间增长的趋势一般是线

性变化的，而未改进的 QEMU 的响应时间随着程序大小的增加呈现抛物线的增长。

4.2 程序的危险系数

根据算法，对恶意程序的分析需要确定一个阈值，危险系数超过该阈值的都会报恶意程序。本次对网络上一些典型的恶意程序进行分析和计算（初始置权值和危险值为 1 的倍数），得出了总体的危险系数大于 90，又对网络上一些典型的白名单程序进行分析和计算，得出了总体白名单的危险系数都小于 30，故 $30<x<90$ 的值可以作为分析的阈值，权值和危险值都设置为 1 的倍数结果图如图 2 和图 3 所示。

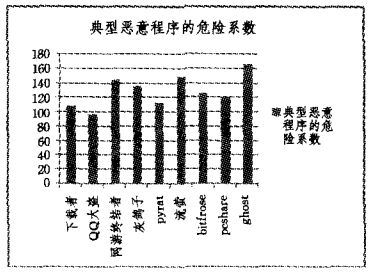

图 2 典型恶意程序的危险系数

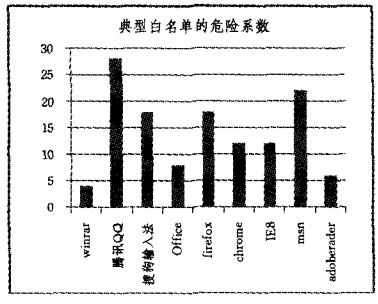

图 3 典型白名单的危险系数

5 恶意程序检测结果

采用两组独立恶意程序样本集进行完全独立的实验。实验一与实验二采用同样的模型，对样本全集进行 8 次不同的阈值测试后的实验结果如表 2 所列。

表 2 两次实验结果

阈值	实验一 $O_{2821} \cup O_{8223}$		实验二 $T_{1694} \cup T'_{7143}$	
	检出率	误报率	检出率	误报率
30	97.3%	11.3%	98.8%	19.3%
40	95.9%	9.1%	97.4%	7.8%
50	95.7%	8.9%	97.4%	8.3%
60	95.2%	8.5%	97.3%	9.1%
70	95.3%	6.8%	95.9%	11.2%
80	91.8%	8.9%	93.7%	12.0%
90	87.3%	9.7%	90.8%	13.4%

从表 2 可以看出：

(1)对于大量的恶意样本和少量的白样本，恶意样本中每个都有许多的恶意行为，对于不同的阈值，都出现了很高的检出率，但是在阈值较低的情况下，如阈值等于 30，有很高的误报率，这主要是由于大量的白样本的一些行为被当作恶意程序的恶意行为。同样在阈值很高的情况下，如阈值等于 100，检出率还是很高，并且还有很高的误报率，这主要是由于有很大一部分恶意程序的危险指数停留在 70～80 阶段，导致了这些恶意程序被当作白名单程序。大概在阈值等于 70 时达到了一个平衡，有着很高的检出率 95.3%，但是误报率只有 6.8%。

(2)对于少量的恶意样本和大量的白样本，总体出现了高的检出率，但是误报率很高。在阈值等于 30 的情况下，检出率有 98.8%，但是误报率高达 19.3%，这主要是由于大量的白样本的一些行为被当作恶意程序的恶意行为。而在阈值等于 40 的情况下，出现了高的检出率，却只有 7.8%的误报率。这主要是由于有大量的白样本的恶意指数在 30 和 40 之间，这些白样本在阈值的增长中被正确检出。随着阈值的增加，检出率总体减小，但是误报率还是呈增长趋势。

结束语 针对当前检测技术以及沙盒技术的不足提出了一种基于沙盒虚拟机动态分析技术，利用 QEMU 实现了一个进程仿真所需要的进程虚拟机，得到了应用程序行为序列流，对该行为序列流进行了形式化定义和分析，提出了行为分析方法。在实践中对上万个样本进行了分析和检测，检测结果表明，该系统可以有效地对恶意程序进行检测，与传统的分析方法相比，可以有效地检测出未知的恶意程序，与主流的沙盒系统相比，增加了人性化操作和易用性、可扩展性。

参 考 文 献

[1] 吴冰，云晓春，高琪. 基于网络的恶意代码检测技术[J]. 通信学报，2007，11

[2] Vimal K K. Securing communication using function extraction technology for malicious code behavior analysis[J]. Computers and Security，2009，28：77-84

[3] Shankarapani M K，et al. Malware detection using assembly and API call sequences[J]. Journal in Computer Virology，2011：107-119

[4] Sami A，Yadegari B，et al. Malware detection based on mining API calls [C]// Proceedings of the 2010 ACM Symposium on Applied Computing. Mar. 2010：1020-1025

[5] 曹跃，梁晓，李毅超，等. 基于差异分析的隐蔽恶意代码检测[J]. 计算机科学，2008

[6] 沙为超，谢荣传. 一种基于本地化特征的恶意代码检测系统设计[J]. 电脑知识与技术：学术交流，2007

[7] 刘艳萍. 恶意代码分析与检测研究现状[J]. 微电脑世界，2009

[8] 王海峰，夏洪雷，孙冰. 基于程序行为特征的病毒检测技术与应用[J]. 计算机系统应用，2006

[9] 史培培，吕林，张明威，等. 基于行为的恶意程序监测研究[J]. 计算机时代，2007

(上接第 11 页)

[4] Killmann W，Schindler W. AIS 31：Functionality classes and evaluation methodology for true (physical) random number generators，version 3.1，Bundesamt fur Sicherheit in der Informationstechnik (BSI). Bonn，2001

[5] Bucci M，Luzzi R. Digital Post-processing for Testable Random Bit Generators[M]. IEEE Circuit Theory and Design (ECCTD 2007). 2007

[6] Sugahara TT，Inoue T. Statistical Properties of Modulo-2 Added Binary Sequences[J]. IEICE Trans. Fundamentals，2004，E87-A (9)：2267-2273

[7] Sarwate D V，Pursley M B. Crosscorrelation properties of pseudorandom and relater sequencesproc[J]. IEEE，68(3)：593-617