第41卷第7期 中国科学技术大学学报 Vol. 41, No. 7
2011年7月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jul. 2011

文章编号:0253-2778(2011)07-0626-10

一种面向软件行为可信性的入侵检测方法

王新志,孙乐昌,陆余良,张 旻

(电子工程学院,安徽合肥 230037)

摘要:针对现有入侵检测方法的问题,面向软件行为可信需求,提出了一种新的静态检测方法.首先讨论并给出了软件行为可信性的定义和形式化描述,并以指令序列形式进行表示;然后,提出了检测方法和流程,通过数据挖掘方法对恶意软件和正常软件进行行为知识发现,利用发现的行为知识对未知软件进行行为可信性判定;最后,对方法进行了实现,对一些行为模式使用选定的样本进行了实验验证.实验结果表明,该方法能够依据软件行为可信策略检测未知软件中的恶意行为,检测成功率高.

关键词:行为可信性;入侵检测;软件行为;序列模式发现;静态检测

中图分类号:TP393 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2011.07.010

Intrusion detection approach towards software behavior trustworthiness

WANG Xinzhi, SUN Lechang, LU Yuliang, ZHANG Min

(*Electronic Engineering Institute, Hefei* 230037, *China*)

Abstract: According to the problems of current intrusion detection methods, a new static detection approach towards software behavior trustworthiness was presented. Firstly, software behavior trustworthiness was discussed and defined formally, and was then described with instruction sequences. Secondly, a detection approach and its process were presented. Malicious behavior knowledge obtained through data mining on malware was organized as trustworthiness policy and used to detect and judge unknown software. Thirdly, the approach was implemented and verified by some behavior patterns on chosen samples. The experimental results show that the approach can detect malicious behavior in unknown software with a high success rate.

Key words: software trustworthiness; intrusion detection; software behavior; sequential pattern discovery; static detection

0 引言

恶意软件(malware)是目前计算机安全的一个主要威胁.近年来,恶意代码技术发展迅速,新型的恶意代码利用计算机系统的漏洞,传播迅速、行踪隐秘,影响日益严重.与此同时,受经济利益驱使,一些免费甚至商业软件中隐藏恶意行为,对用户隐私和数据安全带来严重威胁.现有的杀毒技术和入侵检测方法已经难以对恶意软件形成有效抑制,用户利益不断受到侵害.软件行为的安全性,尤其是未知软件的行为可信任性越来越受到重视.

可信计算发展迅速,是信息安全领域的新思路.

收稿日期:2011-04-28;**修回日期**:2011-06-23

基金项目:国家自然科学基金(60972161)资助.

作者简介:王新志(通讯作者),男,1978年生,博士生/讲师.研究方向:计算机安全.E-mail:xinzhi_wang@163.com

关于“可信”的定义有多种[1-4]，总的来看，现有可信计算观点主要强调实体行为的可预测性、可控性和行为的安全性.目前可信计算的研究主要是可信计算平台(trusted computing platform, TCP)[1]，已经形成了较为完整的规范，并有以可信平台模块(trusted platform module, TPM)为代表的产品得到应用.但现有可信计算研究主要以信息系统的静态完整性度量为主，不能保证经过度量的软件其所有行为是可信的，也不能保证信息系统的动态可信性，并不能从根本上解除安全威胁.

入侵检测根据规则对计算机系统中的误用操作和异常行为进行检测，其中误用检测主要通过系统日志分析[5]，异常检测主要根据通过数据挖掘等方法获得的软件行为特征[6].已有较多的入侵检测研究围绕软件行为展开[7-9].入侵检测系统一般以零散的软件行为知识作为检测规则，对已知的单点恶意行为具备较强的检测能力，但缺乏系统化的软件行为策略定义，难以对目标软件的整体安全性进行检测.典型的例子，某些声明安全的商业软件，其中的隐藏行为会对用户隐私和数据安全带来影响.可信计算要求软件行为符合可信原则，按照预期方式可知、可控地发生，将软件行为发生方式等诸多属性限定在可信范围内.借鉴可信计算的思路，以软件行为的可信任性为检测目标，根据系统预定的行为可信性策略进行入侵检测，能够在一定程度上克服现有入侵检测的问题.

本文提出了一种面向软件行为可信性的入侵检测方法，以软件行为的预期性为检测依据，对未知软件的行为进行检测.软件行为预期性以软件行为可信策略形式体现，是软件行为特性的知识，这些知识部分可以通过人工定义，部分可以通过对已有恶意软件的行为进行学习自动形成.本文采用数据挖掘中的序列模式发现方法，基于预定的支持度，发现恶意软件样本集中存在且在常规软件样本集中不存在的行为序列，自动化地形成软件行为预期性检测依据.然后利用该依据对目标软件进行检测，给出目标软件是否可信的结论.该方法在一定程度上避免了前述方法的不足，实验结果表明该方法能够依据软件行为可信策略正确地检测出含有恶意行为的软件，检测成功率高.

1　软件行为可信性

目前，关于“可信”的定义，学术界和产业界有多种观点.可信计算组织(TCG)[1]用实体行为的预期性来定义可信，认为如果一个实体是可信的，那么它的行为总是以预期的方式达到预期的目标.从这个定义出发，行为可信任是实体可信任的充分条件，行为可信任的关键在于行为应该按照预期的方式发生，并完成预定的功能.可信计算会议(PRDC)和我国科学家[4]持一种由容错计算发展而来的可信计算观点，主要从可靠和安全的角度进行研究，认为可信计算系统是能够提供可靠性、可用性、信息和行为安全性的计算机系统.综合上述两种观点，可信计算是从软件行为的预期性和行为安全角度来衡量软件的可信性.虽然目前产业界的可信计算平台尚不足以保证系统全寿命周期的可信，但是根据软件行为的预期性进行软件可信性判别这个基本观点是正确的，目前有大量的研究正围绕此展开[10-11].

1.1　软件行为可信性的定义

软件行为的预期性具体体现在行为性质、行为输入输出、行为过程、行为属性等方面是否符合须遵守的约定，是否能够为用户接受，是关于软件行为特性的知识[3].

首先给出软件行为可信性的一般性定义：

定义 1.1　对于给定的软件，其所有行为或行为踪迹均属预期行为，则称该软件的行为是可信的，或称该软件满足行为可信性要求.

类似于访问控制，软件行为的预期性通常由可信策略定义并由一组行为规约表示.可信策略的具体内容取决于可信需求，可信需求可包括保密性、完整性、隐私保护等具体方面.

行为可信的定义可形象化地表示为图 1.其中 A 是所有行为的集合；V 是所有预期行为的集合；V′是所有非预期行为的集合；P_t 表示可信策略.可信软件的行为必须符合可信策略 P_t，如 V_t 所示；经验证不可信的软件必然存在不符合可信策略的行为，如 V_m 所示.

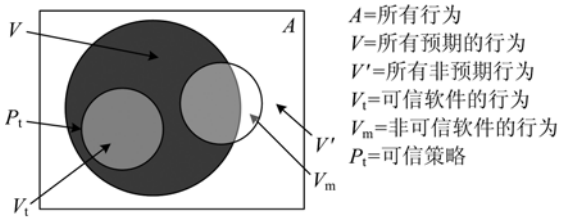

图 1　软件行为可信性

Fig. 1　Software behavior trustworthiness

现实中软件总是在一个给定的环境中运行，不同环境中对可信性的要求是不一样的. 考虑到环境因素，我们给出更贴近真实需求的软件行为可信性定义.

定义 1.2 给定软件集合和系统运行环境，对于软件集合中的任意一个或多个软件构成的子集，如果其所有行为或行为踪迹都符合系统运行环境中预定义的行为可信规约，则该子集中的软件在此系统运行环境中是行为可信的.

该定义在考虑软件运行环境的同时，还关注了实际应用中多个软件可能同时或交叉运行的情况，能够真实反映客观事实，但是缺乏对问题的统一描述，使问题变得复杂难以表示. 我们借鉴文献[12]中关于安全的“自然归纳性”原理，给出如下定义：

定义 1.3 对于任意软件，如果其每种行为及行为的集合都是可信的，则该软件是行为可信的；如果系统内的所有软件都是行为可信的，且所有软件之间是相互独立无干扰的，则该系统是行为可信的.

在软件运行过程中，多个软件的行为可能发生相互干扰，产生新的行为轨迹. 本文以单个软件的行为为研究对象，是一种静态检测方法，无法考察软件运行期间多软件运行时相互干扰产生的新行为. 因此我们以单个软件的行为为研究对象，假定各软件之间行为是相互独立的. 多个软件之间行为的相互干扰情况，必须采用动态检测方法，将另行研究.

1.2 软件行为可信性的形式化描述

软件行为是软件运行时作为主体，依靠其自身的功能对客体的施用、操作或动作[3]. 软件的行为是一系列操作的有序集合，每种行为可以看作一系列操作的轨迹. 软件行为可抽象地描述如下：

给定软件集合 S，对于任意软件 $s\in S$，定义 i 为 s 的一次操作，则 s 的一次行为可表示为一条操作轨迹 t，t 可以用偏序集合$\langle I, A\rangle$表示，其中 $I=\{i_1, i_2, \cdots, i_n\}$，$n\geqslant 1$，表示行为内容；A 是绝对时间序，表示了操作的先后顺序.

此时各操作按照绝对的先后次序发生.

软件 s 的行为集合 T_s 可通过操作轨迹集合表示为

$$T_s=\{t_1, t_2, \cdots, t_k\}=$$

$$\{\langle I_1, A\rangle, \langle I_2, A\rangle, \cdots, \langle I_k, A\rangle\}, 1\leqslant k<\infty$$

行为集合 T_s 的幂集 $P(T_s)$包含了软件 s 的所有行为及行为的所有集合，$P(T_s)$的元素表示为 T'_s.

给定行为可信规约 $Z=\{z_1, z_2, \cdots, z_n\}$(其中 z 表示一条具体的规约)，行为可信性判定函数 $F(x, y)$，单个软件 s 满足行为可信要求的条件是：软件 s 的任意行为及任意行为集合都满足所有可信规约的情况下，单个软件 s 才是行为可信的.

上述条件可形式化表示如下：

$$(\forall T'_s\subseteq P(T_s), \forall z_j\in \mathbf{Z}) \mid F(T'_s, z_j) \quad (1)$$

行为可信性判定函数是一个多值函数，定义如下：

$$F(x, y)=\{\text{true}, \text{false}, \text{error}, ?\} \quad (2)$$

式中，true 表示行为符合规约；false 表示不符合；error 表示规则间存在冲突等情况导致无法作出判定；? 表示无对应规则可供作出判定.

1.3 软件行为可信性的指令序列表示

软件最基本的行为是计算机指令系统中的一条指令. 由一系列指令组合而成的软件作为主体，实现对信息、资源等客体的施用、操作或动作. 从程序设计的角度，可以把完成某个特定功能的一系列程序语句或系统调用看作一个软件行为. 程序经过编译后，生成二进制可执行文件，是由机器指令序列和数据组成的实体，这是计算机软件的最主要表现形式，其中的机器指令序列体现了该软件的行为.

本文的目的主要是检测单个软件中自带的恶意行为，方法是首先将二进制可执行文件进行反汇编，得到由汇编指令序列和数据组成的汇编语言文本，然后以此为对象分析、检测原软件的行为. 这是一种静态检测方法，其优点是可以检测软件的所有行为，漏报率低，不需要执行软件，降低了风险，检测性能高；缺点是不能检测因执行环境注入导致的恶意行为，以及多软件同时执行时相互干扰产生的新行为. 二进制可执行文件经过反汇编后得到的汇编文本是线性序的汇编语言指令序列，在保持大部分行为特性的同时，忽略了多线程特性，降低了分析的复杂度.

我们将汇编指令序列构成的所有有意义的软件行为定义为集合 A. 对于任意的二进制可执行文件，我们假定都可以通过反汇编方法转换为由汇编指令序列形式组成的软件行为集合 T，则 T 是 A 的子集.

给定软件 s，对应节 1.2 中软件行为的形式化描述，以汇编指令序列表示的软件行为要素可实例化如表 1 所示.

表 1　软件行为要素的汇编指令序列表示

Tab. 1　Software behavior elements described in assemble instruction sequence

要素	汇编指令序列表示
软件 s	将其反汇编之后得到的汇编指令文本
操作 i	汇编指令文本中的一条汇编指令
行为 t	完成具体功能的一个汇编指令序列
所有行为 T_s	所有完成具体功能的汇编指令序列集合

带有恶意行为的软件为达到某种目的，必然有正常软件没有的某些行为方式. 如图 2 中根据条件设置挂钩的操作，绝大多数正常的软件在安装和使用阶段不会发生挂钩行为，更不会根据不同条件执行对不同对象的挂钩，这可以作为判断是否存在恶意行为的凭据之一. 类似地，在文件创建、网络访问、进程属性等方面，恶意软件都会与正常软件有所不同. 表现在汇编指令序列上，恶意软件中某些汇编指令片段的指令构成、排列方式与正常软件会有较大区别，甚至在正常软件中不存在类似片段.

```
mov   eax, hmod
push  esi
mov   esi, ds:SetWindowsHookExA
push  0          ; dwThreadId
push  eax        ; hmod
push  offset fn  ; lpfn
push  2          ; idHook
call  esi ; SetWindowsHookExA
mov   ecx, hmod
push  0          ; dwThreadId
push  ecx        ; hmod
push  offset sub_10001460 ; lpfn
push  5          ; idHook
mov   hhk, eax
call  esi; SetWindowsHookExA
mov   dword_1000BDD4,  eax
mov   dword_1000BDC8,  1
pop   esi
```

图 2　键盘拦截行为反汇编代码片段

Fig. 2　Disassemble code of keyboard intercept behavior

定义所有恶意行为的汇编指令序列集合为 T_M，所有正常行为的汇编指令序列为 T_N. 以汇编指令序列描述软件行为的情况下，该方法中对软件的行为可信性具体定义如下：

定义 1.4　一个软件经反汇编处理后，生成的汇编指令序列集合中所有元素都属于正常行为集合 T_N，且没有元素属于恶意行为集合 T_M，则该软件的行为满足可信预期，称该软件是行为可信的.

此时行为可信规约 Z 可描述为

$$\{\forall T'_s \subseteq T_s \mid (T'_s \subset T_N) \cap (T'_s \not\subset T_M)\}; \quad (3)$$

式中，T'_s 的定义与式(1)相同.

结合式(1)～(3)，可以完整地对单个软件的行为可信性作出判定，形成一套完整的面向行为可信性的入侵检测方法. 定义 1.4 并未对正常行为和恶意行为给出限定，节 2 中我们将详细描述通过数据挖掘方法如何得到正常行为和恶意行为的知识，并给出检测系统的工作流程和算法.

2　面向软件行为可信性的入侵检测方法

2.1　方法的基本原理

为达到检测目标，入侵检测器需有两个输入：①恶意特征；②需要检测的对象，一般是进程或文件. 恶意特征有特征码等多种形式，本文选取恶意行为知识作为恶意特征，其内容是汇编指令序列模式.

方法主要包括软件的汇编指令序列生成、恶意行为知识发现算法、行为可信性判定算法 3 部分，如图 3 所示.

①汇编指令序列生成. 将恶意软件样本和正常软件样本通过反汇编生成汇编指令序列，对汇编指令序列进行预处理分别生成恶意指令序列样本集和正常指令序列样本集.

②恶意行为知识发现算法. 从恶意指令序列样本集开始处理，通过恶意行为知识发现算法阶段 A 发现该样本集中存在的满足支持度 1 的频繁指令序列模式集合；接着将该频繁序列模式集合应用到正常指令样本集中，通过恶意行为知识发现算法阶段 B，去除在正常指令样本集中存在且满足支持度 2 的序列模式，剩余的序列模式可视作恶意行为知识.

③行为可信性判定. 将未知软件反汇编为汇编指令序列，用获得的恶意行为知识对其进行检测，判定其是否满足行为可信性要求，给出判定结果. 判定的基本原则遵从式(3).

2.2　汇编指令序列样本集的生成

反汇编技术目前已经比较成熟，能够将大多数可执行文件成功反汇编为有意义的汇编指令序列. 某些软件为了防止破解，会采取加密手段处理，很多恶意代码为了对抗安全员的分析更是采取了代码替代、条件跳转、寄存器重分配等混淆手段. 虽然文献[13]从理论上证明了代码混淆的不可能性，但代码混淆对实际的反汇编过程带来了影响. 在本文的研究中，样本软件和目标软件的选择以 IDA 可直接反汇编以及通过解密程序处理后可反汇编的二进制可

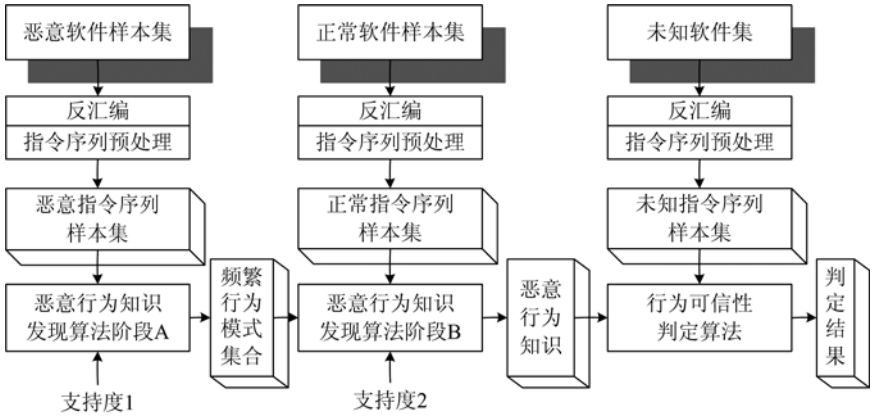

图 3 方法的流程图

Fig. 3 Flow chart of the approach

执行文件为主.

普通的汇编指令由操作码和操作数构成.在现代操作系统中,通常将底层功能进行封装,以系统调用的方式提供给应用层程序访问.假定系统调用功能本身是可信的,为方便特征创建,统一将系统调用名称和地址看作是名称固定的操作数.用汇编指令序列表示的软件行为特征主要包括:

①系统调用(包括系统调用名称、系统调用参数名称、部分系统调用参数值);

②系统调用序列(系统调用信息组成的序列);

③程序控制结构(循环、条件判断).

我们将 IDA 反汇编工具进行了封装,将其作为软件行为指令序列生成的前端,通过它完成二进制代码的反汇编并以文本形式输出.由于软件行为纷繁复杂,汇编指令文本中包含大量的指令片段和数据行,需根据以下特点进行样本选择并对原始数据进行预处理,去除无用信息,以提高算法执行效率.

①同类型恶意行为的指令序列相关性较大,不同类型恶意行为相关性小,选择样本时必须做好特征子集选择.一个软件可能具备一种或多种恶意行为,一种恶意行为可能有几种不同的实现方法,但在实现技术相同的前提下通过反汇编得到的指令序列具有相似性.比如键盘记录行为,只要是通过挂钩技术实现,在汇编指令序列中会以相同或相似的形式体现.因此,每次选择具备同一种恶意行为的样本,可有效提高样本的针对性.

②不同类型的恶意行为体现在汇编指令序列上,需要关注的侧重点不同.比如键盘挂钩行为需要关注系统调用参数值(图 2 所示,idhook=2 表示对键盘挂钩),网络监听行为需要关注系统调用序列和程序控制结构.因此在对特定类型的恶意行为进行发现之前,需要对样本数据依据行为特征进行预处理,进行特征创建,提高学习效率.

2.3 恶意行为知识发现算法

根据节 2.1 的描述,恶意行为知识发现问题可形式化地描述如下:

给定一个含恶意行为的汇编指令序列集合 S_m 和指定的最小支持度 sup_1,从 S_m 中发现满足支持度大于等于 sup_1 的所有序列模式,构成中间集合 M_m;然后对于给定的一组正常行为指令序列 S_n 和指定的最小支持度 sup_2,从 M_m 中排除掉满足支持度大于等于 sup_2 的代码模式,得到最终的恶意行为模式集合 M.

对应问题描述,算法分为 A 和 B 两个阶段.阶段 A 是一个典型的序列模式发现问题[14];阶段 B 是利用另外的样本集对阶段 A 的结果进行剪枝.下面分别讨论.

阶段 A 采用一种基于序列模式发现的频繁行为挖掘算法,该算法借鉴了 PrefixSpan 算法的思想:采用分治策略,首先检查前缀子序列,只将其相应的后缀子序列投影到数据库中;不断产生序列数据库的多个更小的投影数据库,然后在各个投影数据库进行序列模式挖掘.该算法的时空效率比类 Apriori 算法有较大提高.

根据频繁行为挖掘的具体情况,定义 I 为 S_m 中所有单条指令的集合,α_k 为长度小于等于 k 的指令序列频繁项集合.算法细节见算法 2.1.

算法 2.1 频繁行为挖掘算法

初始化:

①$\alpha_1=\{i|i\in I\wedge\frac{\sigma(\{i\})}{|S_m|}\geqslant sup_1\}$　　//找到所有长度为 1 的序列模式

②$S_m|_{\alpha_1}=$createsuffix(S_m,α_1)　//生成初始投影数据库

频繁行为挖掘子过程 MB-prefixscan：

③$b_{k+1}=\{i|i\in S_m|_{\alpha_k}\wedge$prefix(i)$\in\alpha_k\wedge\frac{\sigma(\{i\})}{|S_m|}\geqslant sup_1\}$

　　//根据投影数据库生成新的序列模式

④IF $b_{k+1}\neq\varphi$　　//如果存在满足要求的新序列模式

⑤　$\alpha_{k+1}=\alpha_k\cup b_{k+1}$　　//序列模式合并

⑥　$F_m=\alpha_{k+1}$　　//满足支持度 sup_1 的中间集合

⑦　$S_m|_{\alpha_{k+1}}=$createsuffix($S_m|_{\alpha_k}$,α_{k+1})

　　//生成新的投影数据库

⑧　MB-prefixspan(α_{k+1},k+1,$S_m|_{\alpha_{k+1}}$).

　　//递归调用，找到长度为 k+1 的序列模式

⑨End IF

⑩输出 $M_m=F_m$　　//满足支持度 sup_1 的频繁行为模式集合

阶段 B 是正常行为过滤算法. 由算法 2.1 得到的频繁行为模式集合 M_m 中不但包含了恶意行为模式，还包含了正常行为模式，必须对其进行过滤. 过滤的依据是根据指定的支持度 sup_2，过滤掉在正常指令序列中频繁出现的行为模式. 该算法比较简单，参见算法 2.2.

算法 2.2　正常行为挖掘算法

①for 每条频繁行为模式 t$\in M_m$ do

②　for 每个正常指令序列 s$\in S_n$ do

　　//如果指令序列与行为模式匹配

③　　if sMATCHt

　　//行为模式 t 的支持度增加

④　　$\sigma(t)=\sigma(t)+1$.

⑤　end for

⑥end for

　　//得到满足支持度 sup2 的频繁行为模式集合，即正常行为集合

⑦$M_n=\{t|t\in M_m\wedge\frac{\sigma(t)}{|S_n|}\geqslant sup_2\}$

⑧输出 $M=M_m-M_n$　　//恶意行为模式集合

算法 2.2 最终得到的 M 是存在于恶意软件而不存在于正常软件中的行为模式集合，我们将其看作是恶意行为知识，用于判定目标软件是否存在恶意行为. 不同样本集和支持度的情况下，得到的 M 内容不同，这些恶意行为知识的组合构成了式(5)中所有恶意行为模式 T_M 的一个子集 M^+，一般情况下是真子集，即 $M^+\subset T_M$. 算法过程及结果涉及如下几个主要问题：

①M^+ 中不同恶意行为模式的组合方式问题. M^+ 中包含多种不同类型的恶意行为模式，恶意行为的组合方式会对检测结果产生较大影响. 如果行为模式以与关系进行组合，会有较低的误报，但易产生漏报，此时 $M^+=M_1\wedge M_2\wedge\cdots\wedge M_i$；如果以或关系进行组合，漏报率会降低，但误报率上升，此时 $M^+=M_1\vee M_2\vee\cdots\vee M_i$. 实际检测中，$M^+$ 的具体内容取决于软件行为可信策略，一般是多条件共同作用，表现为松散的组合关系，此时 M^+ 的一种可能的形式为$\{(M_1\wedge M_2)\vee(M_1\wedge M_3)\vee M_4\vee\cdots\vee M_i\}$.

②支持度对恶意行为知识 M^+ 的影响. 支持度 sup_1 的含义是待发现频繁行为序列模式在含恶意行为汇编指令样本集 S_m 中的最小出现概率. 支持度 sup_2 的含义是从 S_m 中发现的频繁序列模式在正常软件中出现的最小概率. sup_1 值的选取影响到能否发现恶意软件中存在的行为序列模式，该值过高会导致漏报率上升；sup_2 值的选取影响到能否过滤掉正常行为模式，该值过高会导致误报率上升. 两个支持度的选取对最终生成的恶意行为知识 M 及检测结果有很大影响. 在实践中，两个支持度值的选取没有理论依据，不同行为特征和不同样本空间情况下取值不是固定的. 实际操作中需要针对待发现的行为特征，对样本数据进行分析后选取.

③算法的效率问题. 恶意行为知识发现算法相对于 PrefixSpan 算法，在计算量上主要增加了后期去除正常代码模式的过程. 从计算复杂度上看，如果忽略恶意行为发现问题本身的一些因素(如指令长度等因素导致的字符串匹配计算量变化)，频繁行为挖掘算法的时间复杂度与 PrefixSpan 算法相同. 正常行为过滤算法的计算复杂度明显低于频繁行为挖掘算法，计算量也比之要小. 因此恶意行为知识发现算法的时间复杂度与 PrefixSpan 算法相同.

2.4　行为可信性判定

定义 1.4 和式(5)中给出了软件行为可信判定规约，这是从可信计算角度给出的定义. 但是正常行为集合 T_N 是难以穷尽的，基于正常行为特征进行可信性判定在计算上不可行，难以给出“某软件是可信的”结论. 从入侵检测的角度，检测目的是发现存在潜在危险的软件，只需给出“某软件是不可信的”结论，在恶意行为特征已知的情况下是可行的.

从入侵检测的角度，定义软件行为可信性判定

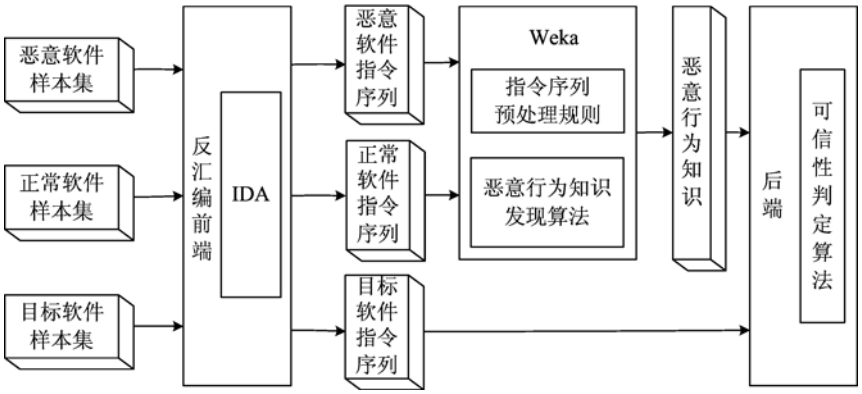

图 4 验证系统组成结构及处理流程

Fig. 4 Structure of testing system

规约如下：

$$\{\exists T'_s \subseteq T_s \mid (T'_s \subset T_M) \cup (T'_s \not\subset T_N)\} \quad (4)$$

该规则的形式与式(3)相似，但含义不同：它表示对于给定的软件 s，如果其存在属于恶意行为或者不属于正常行为的汇编指令序列，则该软件是不可信的.

考虑到正常行为集合 T_N 的不可穷尽性，恶意行为集合 T_M 通过数据挖掘只能获得其一个子集 M^+，对式(4)进行简化，获得实际的判定规则如下：

$$\{\exists T'_s \subseteq T_s \mid (T'_s \subset M^+)\} \quad (5)$$

行为可信性判定根据式(1)，(2)，(5)进行计算，算法流程较简单，主要是一个模式匹配过程，不再赘述.

3 实验验证

使用 weka 模块的形式对算法进行了实现，具体实现中使用正则表达式完成模式匹配条件限定[15]. 实验开发和运行平台硬件为联想 M6000 计算机，操作系统 Windows XP SP3，Weka 3.5，IDA 5.6，开发工具 VisualStadio 2008. 验证系统的结构组成及处理流程如图 4 所示.

选取多个具有典型恶意行为特征的软件作为恶意软件样本，经过验证系统的处理，重点针对键盘拦截行为、注册表自启动行为和网络监听行为进行了分析，选取其中的关键代码段生成恶意行为指令序列样本. 此处首先以键盘拦截行为为例给出实验验证结果.

按照节 2.1 描述的方法，针对键盘拦截行为，使用 IDA 作为前端对 ktr，keylogger 等 10 个带有键盘拦截行为的软件进行了反汇编，然后通过数据预处理得到恶意代码样本. 恶意软件样本集如表 2 所示. 选取包括 Sysinternals 工具集等软件在内的 50 个不含键盘拦截行为的正常软件进行了反汇编，作为正常行为代码样本.

表 2 恶意软件样本集

Tab. 2 Malware sample set

恶意软件名称
Trojan. PSW. Win32. QQPass. aafy
TrojanSpy. KeyLogger. ii
PSW. OnLineGames. an
PSW. OnLineGames. om
TrojanSpy. KeyLogger. fw
Trojan. Clicker. Agent. ahy
Trojan. Spy. Bancos. kcp
Trojan. Spy. Bancos. lyw
Trojan/PSW. Agent. zq
Ktr

通过恶意行为知识发现算法，得到两种键盘拦截行为序列模式. 在 sup_1 取值为 0.8，sup_2 取值 0.5 的情况下，得到一种通过 API 函数 SetWindowsHookEx 实现的行为模式，如图 5 所示，简称“X 模式”. 在 sup_1 取值为 0.4，sup_2 取值为 0.5 的情况下，得到一种通过驱动实现的行为模式，如图 6 所示，简称“Y 模式”. 图中“*”表示任意字符串，“#”表示多行.

假定行为可信策略将 M^+ 取值为 $\{X \vee Y\}$，使用该序列模式对 gamepass 等 8 个含有、mydown 等 3

```
push   *      ; *
push   *      ; *
push   *   ; *
push   2      ; idHook
call   * ; SetWindowsHookExA
```

图 5 键盘拦截行为序列模式 X

Fig. 5 Keyboard intercept behavior pattern X

表 3　测试目标软件集及测试结果 1

Tab. 3　Testing software set and test result 1

测试软件名称	行为特征	检测结果	可信性判定
Trojan/PSW. GamePass. cop	键盘拦截,注册表自启动	Y 模式	不可信
PSW. OnLineGames. yy	键盘拦截,注册表自启动	Y 模式	不可信
Trojan/Startpage. a	键盘拦截	X 模式	不可信
Backdoor/Agent. ccn	键盘拦截,注册表自启动	X 模式	不可信
TrojanSpy. KeyLogger. hg	键盘拦截	X 模式	不可信
TrojanSpy. Agent. at	键盘拦截,注册表自启动	X 模式	不可信
Win32. Troj. HuigeziHooK	键盘拦截,注册表自启动	Y 模式	不可信
Getkb. Test	键盘拦截	X,Y 模式	不可信
Trojan. win32. small. qg	注册表自启动	无	正常
Trojan. win32. mydown. che	注册表自启动	无	正常
Trojan. win32. Scar. cuzp	注册表自启动	无	正常
Passmark Keyboardtest	键盘测试	X 模式	不可信
RealVNC Viewer	远程控制	X 模式	不可信

```
push    * ; "Driver\kbdhid"
#
call    * ; "IoAttachDeviceToDeviceStack"
#
push    * ; "\Device\KeyboardClass0"
#
call    * ; "IoGetDeviceObjectPointer"
```

图 6　键盘拦截行为序列模式 Y

Fig. 6　Keyboard intercept behavior pattern Y

个不含有键盘拦截行为的恶意软件,以及 Keyboard test,RealVNC Viewer 两个含有键盘拦截行为的正常软件进行检测. 其中程序 getkb. test 是作者全新编写的键盘拦截测试程序,包括 API 挂钩和驱动拦截两种行为模式,能够拦截所有键盘输入并保存到文件,通过了卡巴斯基 2010 的杀毒检测. 测试目标软件集及检测结果如表 3 所示.

从检测结果可以看出,针对给定的样本集和目标软件,根据预定的行为可信策略,能够检测未知软件中的键盘拦截行为. 从入侵检测的角度,对 Keyboard test,RealVNC Viewer 两个正常软件的检测结果属于误报,而对 Trojan. win32. small. qg, Trojan. win32. mydown. che, Trojan. win32. Scar. cuzp 3 个恶意软件则是漏报.

增加恶意行为知识,改进行为可信策略,可有效降低漏报率和误报率. 通过另外的样本集进行恶意行为知识发现,获得注册表自启动行为模式 R(如图 7 所示)、网络监听行为模式 Z(如图 8 所示),行为可信策略将 M^+ 取值为{(X∧R)∨(Y∧R)∨(Z∧R)},获得检测结果如表 4 所示. 此时没有误报,但对于只存在键盘记录行为的恶意软件产生了漏报.

实验结果表明,该方法能够成功检测未知软件中的已知恶意行为,检测成功率高. 在获得足够的恶意行为知识的情况下,综合多种行为模式进行检测,

```
#
push    * ;"注册表自启动项字符串"
push    * hKey
call     * RegOpenKeyA
#
push    *
push    *
push    *
push    *
call     * RegSetValueExA
#
push    * hKey
call     * RegCloseKey
*
*
```

图 7　注册表自启动行为序列模式 R

Fig. 7　Auto start by registry behavior pattern R

```
loc_#
#
push    *
push    *
push    *
*
call     * recv *
#
call     *
#
jmp    * loc_# *
```

图 8　网络监听行为模式 Z

Fig. 8　Network monitoring behavior pattern Z

表 4 测试目标软件集及测试结果 2

Tab. 4 Testing software set and test result 2

测试软件名称	行为特征	检测结果	可信性判定
Trojan/PSW. GamePass. cop	键盘拦截,注册表自启动	Y,R 模式	不可信
PSW. OnLineGames. yy	键盘拦截,注册表自启动	Y,R 模式	不可信
Trojan/Startpage. a	键盘拦截	X 模式	正常
Backdoor/Agent. ccn	键盘拦截,注册表自启动	X,R 模式	不可信
TrojanSpy. KeyLogger. hg	键盘拦截	X 模式	正常
TrojanSpy. Agent. at	键盘拦截,注册表自启动	X,R 模式	不可信
Win32. Troj. HuigeziHooK	键盘拦截,注册表自启动	Y,R 模式	不可信
Getkb. Test	键盘拦截	X,Y 模式	正常
Trojan. win32. small. qg	注册表自启动	Z,R 模式	不可信
Trojan. win32. mydown. che	注册表自启动	Z,R 模式	不可信
Trojan. win32. Scar. cuzp	注册表自启动	Z,R 模式	不可信
Passmark Keyboardtest	键盘测试	X 模式	正常
RealVNC Viewer	远程控制	X 模式	正常

严格定义行为可信策略,可有效降低方法的漏报率和误报率,提高检测的准确性. 在下一步的工作中我们将增加软件样本,以期获得更多软件行为模式,同时加大目标软件样本空间,对方法的检测性能进行进一步的验证和完善.

4 结论

本文提出了一种面向软件行为可信性的静态入侵检测方法,给出了软件行为可信性的定义,介绍了方法的基本原理,设计实现了恶意行为知识发现算法及检测算法,并给予了初步的实验验证. 实验结果表明该方法可根据已知软件的行为特征有效检测未知软件中的恶意行为.

可信计算是当前研究的热点之一. 在"绝对安全"不可及的情况下,以"可信"为目标研究计算机安全问题是一种务实的思路. 自动化行为知识发现方法可用于构建软件行为可信性策略的部分内容,不能全面覆盖所有策略,却是人工进行可信性策略定义的有效补充. 本文中我们主要以静态方法研究软件的单点行为,如键盘拦截、注册表自启动等. 由于一些正常软件也存在类似行为,因此不可避免地存在漏报和误报. 下一步我们将研究运行监控情况下软件行为的可信性问题,重点研究多主体综合作用下软件行为的可信性检测与控制问题,以期克服本文静态检测方法存在的问题和不足.

参考文献(References)

[1] Trusted Computing Group. TCG Specification Architecture Overview: Revision 1. 4 [EB/OL]. [2011-02-20]. http://www. trustedcomputinggroup. org/files/resource _ files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG _ 1 _ 4 _ Architecture _ Overview. pdf.

[2] ISO. ISO/IEC 15408-1: Information technology - Security techiques - Evaluation critetia for IT security - Part I: Introduction and general model [S]. ISO, 2009.

[3] Qu Yanwen. Software Behavior [M]. Beijing: Publishing House of Electronics Industry, 2004.
屈延文. 软件行为学[M]. 北京:电子工业出版社,2004.

[4] Zhang Huanguo, Luo jie, Jin Gang, et al. Development of trusted computing research [J]. Journal of Wuhan University (Natural Science), 2006, 52(5): 513-518.
张焕国,罗捷,金刚,等. 可信计算研究进展[J]. 武汉大学学报(理学版), 2006, 52(5): 513-518.

[5] Denning D E. An intrusion-detection model [J]. IEEE Transactions On Software Engineering, 1987, 13(2): 222-232.

[6] Wenke L, Salvatore J S. Data mining approaches for intrusion detection [C]//Proceedings of the 7th conference on USENIX Security Symposium. Berkeley, CA, USA: USENIX Association, 1998.

[7] Mihai C, Somesh J, Christopher K. Mining specifications of malicious behavior [C]// Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2007). New York: ACM, 2007: 5-14.

[8] Matthew G. S, Eleazar E, Erez Z, et al. Data mining methods for detection of new malicious executables

[C]//Proceedings of the 2001 IEEE Symposium on Security and Privacy. Oakland, CA, USA: IEEE Computer Society, 2001, 38-49.

[9] Mila D, Mihai C, Somesh J. A semantics-based approach to malware detection [C]// Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. New York: ACM, 2007.

[10] Li Xiaoyong, Gui Xiaolin, Mao Qian, et al. Adaptive dynamic trust measurement and prediction model based on behavior monitoring [J]. Chinese Journal of Computers, 2009, 32(4): 664-674.
李小勇,桂小林,毛倩,等. 基于行为监控的自适应动态信任度测模型[J]. 计算机学报, 2009, 32(4): 664-674.

[11] Yang Xiaohui. Researches on Dynamic Trusted Theories and Models of Software Behavior [D]. Hefei: University of Science and Technology of China, 2010.
杨晓辉. 软件行为动态可信理论模型研究[D]. 合肥: 中国科学技术大学, 2010.

[12] Bell D E, Lapadula L J. Secure computer system: Unified exposition and multics interpretation, MTR-2997 Rev. 1 [R]. Bedford, CA: MITRE Corporation, 1976.

[13] Boaz B, Oded G, Russell I, et al. On the (im) possibility of obfuscating programs[J]. Lecture Notes in Computer Science, 2001, 2139: 1-18.

[14] Agrawal R, Srikant R. Mining sequential patterns [C]//Proceedings of 11th International Conference on Data Engineering. IEEE Computer Society, 1995: 3-14.

[15] Garofalakis M, Rastogi R, Shim K. Mining sequential patterns with regular expression constraints[J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(3):530-552.

(上接第 588 页)

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

[5] Kim Y, Perrig A, Tsudik G. Group key agreement efficient in communication[J]. IEEE Transactions on Computers, 2004, 53(7): 905-921.

[6] Tu Shanshan, Ma Chunbo, Ao Faliang, et al. The research of hierarchical group key management for ad hoc networks [C]//Proceedings of the 2010 International Conference on Intelligent Computing and Integrated Systems. Piscataway, NJ, USA: IEEE, 2010:121-124.

[7] Li Huixian, Pang Liaojun, Wang Yumin, et al. Key management scheme without secure channel for ad hoc networks[J]. Journal on Communications, 2010, 31(1):112-117.
李慧贤, 庞辽军, 王育民,等. 适合 ad hoc 网络无需安全信道的密钥管理方案[J]. 通信学报, 2010, 31(1): 112-117.

[8] Yuan Ting, Ma Jianqing, Zhong Yiping, et al. Key Management Scheme Using Time-Based Deployment for Wireless Sensor Networks[J]. Journal of Software, 2010, 21(3):516-527.
袁珽, 马建庆, 钟亦平,等. 基于时间部署的无线传感器网络密钥管理方案[J]. 软件学报, 2010, 21(3): 516-527.

[9] Zhou Fucai, Xu Jian, Xu Haifang, et al. Research of STR multicast key management protocol based on bilinear pairing in ad hoc network [J]. Journal on Communications, 2008, 29(10):117-125.
周福才, 徐剑, 徐海芳,等. Ad hoc 网络中基于双线性配对的 STR 组密钥管理协议研究[J]. 通信学报, 2008, 29(10):117-125.

[10] Kim Y, Perrig A, Tsudik G. Communication-efficient group key agreement[M]// Trusted Information: The New Decade Challenge. Norwell, MA, USA: Kluwer Academic Publishers, 2001: 229-244.

[11] Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C[M]. 2nd ed. New York, NY, USA: Wiley, 1996: 686-688.