一种基于污点数据传播和无干扰理论的软件行为可信性分析模型

陈 曙[1] 叶俊民[1] 张 帆[2]

(华中师范大学计算机学院 武汉 430079)[1] (杭州电子科技大学通信工程学院 杭州 310018)[2]

摘 要 以无干扰理论为基础,提出了一种基于污点数据流的软件行为可信分析模型。该模型通过跟踪程序外部输入的污点数据,提取可能引发系统不可信的关键系统调用,并建立污点传播调用序列。利用完整性条件下的污点信息流无干扰模型来判定构成污点传播的系统调用序列执行时是否可信,并给出了调用序列可信性判定定理。

关键词 可信计算,污点分析,无干扰

中图法分类号 TP309 **文献标识码** A

Taint Trace with Noninterference Based Approach for Software Trust Analysis

CHEN Shu[1] YE Jun-min[1] ZHANG Fan[2]

(School of Computer Science, Huazhong Normal University, Wuhan 430079, China)[1]

(Communication School, Hangzhou Dianzi University, Hangzhou 310018, China)[2]

Abstract A model for software trust analysis was proposed based on taint data trace and noninterference theory. This approach extracts core operation system APIs that may cause un-trusted behaviors by tracing taint data imported from outside of software environment. These APIs forms a taint dependency behavior model and imports information flow model to analyse whether it is trusted. Theorem for the trust determinant is also improved.

Keywords Trusted computing, Taint analysis, Noninterference model

1 引言

软件可信性度量是信息安全领域一个重要的研究方向。传统的静态分析法通过比较软件的哈希校验码来判断软件的可信性。该方法虽然能在一定程度上判断软件自身是否被恶意篡改,但是却无法发现自身的漏洞以及软件在执行过程中作为宿主而引发的恶意行为,例如攻击者可以利用缓冲区溢出、格式化字符串等软件自身的脆弱性,来达到破坏系统,获得未经授权的访问控制的目的。鉴于此种观点,研究者们提出了动态的软件可信性分析方法,即从软件运行时所展现的行为来判断软件是否可信。也正是在这种背景下,可信计算组织(TCG, Trusted Computing Group)提出了"可信计算"概念,从软件动态行为的角度给出软件实体可信的定义,并强调行为的可预测性和可控性,认为"当一个实体始终沿着预期的方式(操作或行为)达到预期的结果,那么该实体就认为是可信的"[1,2]。

截至目前为止,基于软件动态行为的可信性分析方法的研究只取得了初步的成效,还有许多问题需要深入研究,比如缺少对于软件动态行为的有效提取和分析方法,或缺少通用的可信验证理论体系等[2]。而根据 TCG 组织提出的可信定义,一个完整的系统行为可信分析模型至少需要两个部分:(1)合理的系统动态行为模型;(2)针对行为模型的可信推演理论和方法。本文针对软件动态可信分析中存在的问题展开研究,提出一种基于污点数据传播和无干扰理论的软件行为可信分析模型。本文的主要贡献如下:

(1)提出一种基于污点数据追踪的软件动态行为分析方法,该方法以污点数据依赖的方式表示软件在执行中所使用的可能产生恶意行为的关键系统调用,从而建立软件动态行为模型,能够剔除正常的无关行为,缩小分析范围,并从一定程度上降低了行为混淆对分析的影响。

(2)以无干扰模型为基础,提出针对(1)中软件动态行为模型的行为可信性推演方法。

(3)综合(1)和(2),本文提出的方法构成了一个完整的软件行为可信性分析模型。

本文第 2 节介绍相关研究工作;第 3 节介绍基于污点数据追踪的软件动态行为分析方法;第 4 节介绍软件动态行为模型的行为可信性推演方法;第 5 节简述实现方法;最后是总结。

2 相关研究

污点分析方法是由加州大学伯克利分校的宋晓东研究小组[3,4]提出的一种软件动态行为分析方法。污点数据是指引

到稿日期:2012-07-28 返修日期:2012-11-23 本文受中央高校自主科研基金(CCNU11A01012, CCNU11A02007),湖北省自然科学基金(2010CDB04001)资助。

陈 曙(1981—),男,博士,讲师,主要研究方向为软件工程、可信软件、信息安全;叶俊民(1965—),男,博士后,教授,主要研究方向为可信软件工程;张 帆(1978—),男,博士,讲师,主要研究方向为软件工程、可信计算。

起软件产生不安全因素的数据。污点数据的来源称为污点源，包括键盘输入、网络套接字等。系统调用是应用程序与系统交互的接口，应用程序的行为目标几乎都是通过系统调用实现的。文献[5,6]认为大多数由恶意代码造成的软件不可信均是通过攻击者引导程序访问攻击者指定的内存空间执行指定的系统调用，即非法调用转移，例如缓冲区溢出攻击、格式化字符串攻击等。利用污点数据进行分析，可以描绘和重现这些攻击过程，跟踪恶意行为所篡改的内存区域以及发现恶意行为如何利用相关系统调用来达到攻击的目的，且可依此构建更严格、更具有针对性的安全策略规则。因此，监控关键系统调用，或者说，监控污点数据的传播方式，能够在一定程度上消解恶意代码混淆对分析的影响，可以使得恶意代码的行为分析更准确，更有针对性[7]。例如文献[3,4]利用污点分析方法来描述系统运行时的关键信息流传播特征，文献[5,6]利用污点传播原理来消除恶意行为混淆，而上述研究均局限在特定的情况下，缺乏一个通用的可信验证理论体系作为支撑。

从系统运行的角度来看，系统调用以及返回、调用参数的传入、输入等均可看成是系统执行的信息流传递。而由 Rushby 等人提出的无干扰模型[8-11]是一种典型的信息流传递模型。无干扰模型被广泛用于基于访问控制策略和通道控制的系统可信系统描述中，例如文献[12]尝试在完整性条件下使用无干扰模型来解释系统运行时信息流释放的可信问题。

综合以上分析，本文根据污点传播思想建立污点数据依赖的系统调用结构图作为系统动态行为模型，利用无干扰思想对模型的污点信息流传播进行可信性分析，从而构成一个完整的软件行为可信性分析模型。

3 基于污点数据流的软件动态行为分析

本文约定，大写字母如 A 表示系统调用序列，小写字母如 a 表示单个系统调用。将系统调用定义为一个三元组 $a=\{ret, par_i, par_o\}$，其中 ret 表示系统调用的返回值，par_i 表示传入该系统调用的参数列表，par_o 表示该系统调用传出的参数列表，并对外部输入的敏感参数标记为污点。

本文构造二次筛选算法 TF(Twice Filter)来建立基于污点数据依赖的系统调用结构图，利用该图作为污点信息流无干扰模型的输入来进行行为可信分析。TF 的主要思想描述如下。

第一轮筛选：跟踪系统调用数据源，如果 $a.par_o$ 或 $a.ret$ 传播至 $b.par_i$，则记录系统调用 b 数据依赖于 a。重复上述过程，直到污点数据不再传播或程序执行结束。第一轮筛选生成污点传播调用依赖图 $G=\{in,out,A,E\}$，其中 in 为依赖图的入口节点，out 为出口节点，A 为依赖图中所有系统调用的执行标记集合，E 为所有边，表示为 $A\times A$。对于系统 S 而言，由于在分析中可能有多个污点数据输入源，从而可能出现多个依赖图，因此将这些图表示为 $GS=\{G_1,G_2,\cdots,G_n\}$。

第二轮筛选：对于执行序列 GS 中的每个污点信息依赖图 $G_i\in GS$，从 G_i 的出口节点开始逆向深度优先扫描，对于其每个系统调用执行 $a_i=\{ret,par_i,par_o\}$，假如 par_i 被标记为污染，则将其存入集合 S 中。逆向扫描 G_i。对于每个系统调用 $a\in G_i$，如果 a 定义了某个内存地址 $l\in S$，则标记调用 a 为污点依赖，并将 l 从 S 中移除，另外将 a 所使用的内存地址加入到 S 中。重复上述过程，直到 S 为空或到达 G_i 的入口节点，并删除 G_i 中所有未被标记为污点依赖的节点及其与该节点相关联的边。第二轮筛选算法的目的是当某个系统调用 a 访问某个污染数据 m 时，算法将会找到在 a 执行前且最靠近 a 的定义或改写污染数据 m 的系统调用，从而忽略掉 m 的构造过程。

按照二次筛选算法构建的依赖图表示为 $G=\{in,out,A,E\}$。对于任意 $G_i\in GS$，由于其包含一个入口和一个出口，根据图的可达性可知，G_i 可看作是数条由 in 到 out 的系统调用序列 A_{ij} 的复合。将 G_i 拆解为可能具有重叠的系统调用序列集合表示为：$G_i=\{A_{i1},A_{i2},\cdots,A_{in}\}$。根据 TCG 组织对软件系统动态可信的定义，对于一个软件系统 S 而言，如果所有 $G_i\in GS$ 中所有的 $A_{ij}\in G_i$，在同等环境中执行的情况与预期相符，即可认为 A_{ij} 所展现的行为是可信的，那么可认为系统 S 是可信的。下面的工作便是对系统调用序列 A_{ij} 做行为可信推演，判断其是否可信。

4 基于污点信息流无干扰的行为可信性推演方法

根据可信计算组织 TCG 所作的定义，判断一个系统是否可信，就是看它的行为是否与预期一致。对于系统调用执行序列 A 而言，如果 A 与预期的执行结果相符，则可认为 A 的行为是可信的。我们假设序列 A 执行于安全的结构化环境中，并借助于 Rushby 的无干扰模型思想[8-11]，将利用污点数据进行的非法调用转移看作是一种干扰。如果污点依赖调用序列 A 的执行违背了结构化的无干扰安全策略，则认为序列 A 利用了污点数据产生了恶意的行为。基于上述原理，本节给出系统调用序列的行为可信性分析推演方法。

4.1 基于污点信息流无干扰模型的行为可信推演方法

假设调用序列 A 在结构化机器 M 中运行，首先给出如下几个定义。

定义 1 结构化机器 M 被定义为一个八元组 $M=\{S,Call,O,D,P,do,out,process\}$，其中 $Call$ 为系统调用集；O 为输出集；D 为系统调用所在进程所属的安全域集，且满足对于所有 $a\in Call$，有 $dom(a)\in D$；P 为安全策略集，定义为 $p:D\mapsto D$，其中 $p\in P$，二元关系 $\mapsto$ 满足自反，表示两个安全域之间有污染信息流传递，其补关系 $\not\mapsto$ 表示域间无污染信息干扰；N 为机器 M 运行的名字空间；V 为与 M 对应的值集；S 为状态集，每个 $s\in S$ 表示为 N 和 V 的有序对集合；do 为单步执行函数，表示状态的迁移，定义为 $do:S\times Call\to S$；$output$ 表示执行某个系统调用后的返回值，定义为 $out:S\times Call\to O$；$process$ 表示执行一个系统调用序列后产生的状态，定义为 $process:S\times A*\to S$。令 τ 为空调用，$\circ$ 为连接操作，则执行序列与单步执行函数满足如下递归关系：

$$\begin{cases} process(s,\tau)=s \\ process(s,a\circ A)=process(do(s,a),A) \end{cases}$$

定义 2 污染信息流定义为，某个域 $d_1\in D$ 中的行为被允许写访问内存区域 n，另一个域 $d_2\in D$ 中的行为被允许读访问 n，如果 n 被标记为污染，则称域 d_1 到 d_2 在域间无干扰策略 $d_1\not\mapsto d_2$ 的控制下有污染信息流存在。形式化描述如下：

$$\exists n\in Taint: n\in write(d_1)\wedge n\in read(d_2)\Leftrightarrow d_1\not\mapsto d_2$$

其中，由污染信息流引发的未授权行为也称为非法污染

控制流，污染信息流所对应的系统调用序列称为污点依赖调用序列，其中每个系统调用称为污点传播调用。

定义 3 对于结构化机器 M，将其运行时的存储空间定义为一个五元组 $R=\{N,V,read,write,observe\}$，其中 N 为存储空间每个存储单元的标识集，V 为对应的值集，$read$ 和 $write$ 分别表示读和写操作，$observe$ 表示观察。

定义 4 定义函数 $origin:Call^* \times D \to P(D)$，其表示获取污点依赖调用序列中所有具有污染信息流传递的安全域：

$$\begin{cases} origin(\tau,u)=u \\ origin(a\circ A,u)=origin(A,u)\cup dom(a), \text{if } \exists v\in origin(A,u)\wedge dom(a)\vdash\!\!\!\rightarrow u \\ origin(a\circ A,u)=origin(A,u), \text{else} \end{cases}$$

定义 5 如果机器 M 的环境是多视点可观察的，则对于所有安全域 $u\in D$，存在状态等价关系 $s[\approx u]t$，其表示安全域 u 中观察状态 s 和 t 的环境取值是一致的。形式定义为：

$$s[\approx u]t \Leftrightarrow \forall n\in observe(u):s(n)=t(n)$$

定义 6 定义函数 $filter:Call^* \times D \to Call^*$，其表示在某污点依赖调用序列中过滤掉所有直接或间接与指定安全域有污点信息干扰的系统调用后得到的子调用序列：

$$\begin{cases} filter(\tau,v)=\tau \\ filter(a\circ A,u)=a\circ filter(A,u), \text{if } dom(a)\in origin(A,u) \\ filter(a\circ A,u)=filter(A,u), \text{else} \end{cases}$$

污点依赖调用序列及其数据依赖序列记录了软件在执行关键系统调用时，被污染数据的传播过程，这些被污染的数据有可能被攻击者利用。按照可信计算组织 TCG 对软件可信的定义："如果软件的所有实际运行行为均与预期方式相符合，则可认为该软件是可信的"，我们认为，对于某个软件而言，如果其在安全环境中运行，且可能产生恶意行为的关键系统调用的执行与预期方式相符，换句话说，如果在运行时污染数据的信息流传播与预期相符，则可认为该软件在安全环境中是可信的，反之则不可信。

定义 7 在结构化环境中污点依赖调用序列 A 对于污点信息无干扰策略是可信的，当该序列的执行环境满足分层屏蔽，且安全域间无非法污染控制流存在：

$$out(process(s_0,A),a)=out(process(s_0,filter(A,dom(a))),a)$$

定义 7 给出了判定系统动态安全的准则，其中左式表示污点依赖调用序列 A 实际的运行输出，右式表示在无污染信息干扰的策略作用下期望的输出。如果左右两式结果相同，则说明污点依赖调用序列 A 没有产生非法行为，因此是可信的。

由于定义 7 中 A 为序列，不便于分析，因此将其扩展为污点传播调用单步执行时的情形。

定理 1 污点依赖调用序列 A 是可信的，当且仅当其满足如下 3 个性质：

(1)单步输出屏蔽性

$$s[\approx u]t \Rightarrow out(s,a)=out(t,a)$$

(2)状态等价一致性

$$s[\approx u]t \wedge s[\approx dom(a)]t \Rightarrow do(s,a)[\approx u]do(t,a)$$

(3)污点传播无干扰

$$dom(a)\vdash\!\!\!\not\rightarrow u \Rightarrow s[\approx u]do(s,a)$$

$$\forall n\in observe(u):s(n)\neq do(s,a)(n)\Rightarrow dom(a)\vdash\!\!\!\rightarrow u$$

$$dom(a)\vdash\!\!\!\rightarrow u \Rightarrow \exists n\in observe(u):s(n)\neq do(s,a)(n)$$

证明：根据单步输出屏蔽性可看出，定义 7 成立的充要条件是证明式(1)的成立：

$$\begin{matrix} process(s_0,A)[\approx dom(a)] \\ process(s_0,filter(A,dom(a))) \end{matrix} \quad (1)$$

将式(1)进一步归纳为式(2)：

$$process(s_0,A)[\approx u]process(s_0,filter(A,u)) \quad (2)$$

要证明式(2)成立，需要对 A 的长度进行数学归纳。当 $A=\tau$ 时，式(2)必然成立，则证明目标即归纳为假设 A 的长度大于 0 时，式(3)成立，则 $a\circ A$ 时，式(4)也成立。

$$s[\approx u]t \Rightarrow process(s,A)[\approx u]process(t,filter(A,u)). \quad (3)$$

$$s[\approx u]t \Rightarrow process(s,a\circ A)[\approx u]process(t,filter(a\circ A,u)) \quad (4)$$

证明：式(4)可根据假设 $dom(a)\in origin(i\circ A,u)$ 和 $dom(a)\notin origin(i\circ A,u)$ 两种情况分别对序列 A 做长度归纳，利用反证法即可得证，由于篇幅限制，具体过程省略。

本节根据无干扰理论建立了针对污点传播调用序列的可信性推演方法。为了更有利于系统平台实现，下面给出在结构化环境中具体的解释。

4.2 结构化环境下的解释

令函数 $content:S\times N \to V$ 表示状态 s 下存储标记 n 的取值，其中 n 表示在安全域 u 内所能观察到的机器的状态，包括进程状态、寄存器状态等。因此可将状态等价改写为如下形式：

$$s[\approx u]t \Leftrightarrow \forall n\in observe(u):content(s,n)=content(t,n)$$

定义函数 $shift:Call\times D \to P(D)$ 表示污点传播调用 a 在安全域 u 下所能跳转的安全域集，并满足如下性质：

$$\forall a\in Call:u\in shift(a,dom(a))\Leftrightarrow ring(dom(a))\searrow ring(u)\wedge(dom(a)\vdash\!\!\!\rightarrow u)\in P$$

其中函数 $ring(u)$ 表示安全域 u 的安全等级。

定义 8(域内一致性) 域内一致性表示在同一安全域内执行污点传播调用，执行后生成的状态对该安全域仍保持一致。即满足如下蕴含式：

$$\forall n\in observe(dom(a)):content(s,n)=content(t,n)\Rightarrow \forall n\in observe(dom(a)):content(do(s,a),n)=content(do(t,a),n)$$

定义 9(域间单向性) 域间单向性表示污染数据跨域传递时，必须按照域间层级关系有序分层流动。即满足如下蕴含式：

$$\forall n\in observe(u):content(s,n)=content(t,n)\wedge ring(dom(a))\searrow ring(u)\Rightarrow \forall n\in observe(u):content(do(s,a),n)=content(do(t,a),n)$$

定义 10(域间一致性) 表示污染数据在域间传递时，应通过具有权限的统一方式来进行。即满足如下蕴含式：

$$\forall n\in observe(u):content(do(s,a),n)\neq content(s,n)\Rightarrow n\in write(dom(a))\wedge u\in shift(a,dom(a))$$

定理 2 说明了在结构化环境下，如果污点依赖调用序列不满足域内一致性，或违背域间污点数据传播单向性或产生未经授权的域间转移行为，则该污点依赖调用序列不可信。

定理 2 给定污点依赖调用序列 A，假设 A 在结构化环境中运行，如果满足初始环境安全，但不满足定义 8－定义 10 中的任意一条，则说明序列 A 不可信。

证明：假设序列 A 是可信的，即满足定理 1 的 3 条性质。

(1)假设序列 A 不满足域内一致性，即有式(5)存在：

$$\forall n\in observe(dom(a)):content(s,n)=content(t,n)\Rightarrow \forall n\in observe(dom(a)):content(do(s,a),n)\neq content(do(t,a),n) \quad (5)$$

根据状态在安全域下等价的定义。由于满足 $content(s,n)=content(t,n)$，因此有 $s[\approx dom(a)]t$。由于根据定理 1 的单步输出屏蔽性有 $out(s,a)=out(t,a)$，因此有 $s[\approx dom(a)]t$。根据状态等价一致性即有 $do(s,a)[\approx u]do(t,a)$。根据状态在安全域下等价的定义，即有 $content(do(s,a),n)=content(do(t,a),n)$，与假设不符。即若序列 A 不满足域内一致性，则不满足单步输出屏蔽性和状态等价一致性，因此序列 A 是不可信的。

(2)假设序列 A 不满足域间单向性，即有式(6)存在：

$$\forall n\in observe(u):content(s,n)=content(t,n)\wedge ring(dom(a))\searrow ring(u)\Rightarrow \forall n\in observe(u):content(do(s,a),n)\neq content(do(t,a),n) \quad (6)$$

根据定理 1 的单步输出屏蔽性和状态等价一致性，由于满足 $content(s,n)=content(t,n)$，且在 $dom(a)$ 和 u 所属的层级间存在污染信息传递，但是满足层间有序，即有 $dom(a)\nvdash u$，即调用 a 的执行在 u 中不可观察，根据定理 1 的污点传播无干扰性质，有 $s[\approx u]do(s,a)$，且有 $s[\approx u]do(t,a)$，因此有 $do(s,a)[\approx u]do(t,a)$，且对于 $\forall n\in observe(u)$，有 $content(do(s,a),n)=content(do(t,a),n)$，与假设不符。即若序列 A 不满足域间单向性，则不满足输出屏蔽性，状态等价一致性和污点传播无干扰性，因此序列 A 是不可信的。

(3)假设满足式(7)：

$$\forall n\in observe(u):content(do(s,a),n)\neq content(s,n) \quad (7)$$

由于有 $content(do(s,a),n)\neq content(s,n)$，根据定理 1 污点信息传播的无干扰性质，有 $dom(a)\nvdash u$，根据定义 2，有 $n\in write(dom(a))$。如果假设 $u\notin shift(a,dom(a))$，由于有 $dom(a)\nvdash u$，根据域转移函数定义有 $dom(a)\nvdash u\notin P$，表示产生不满足策略集的非法转移，因此序列 A 是不可信的。证毕。

5 系统实现

为了说明模型的可用性，在 Ubuntu 9.04-32bit 下实现了基于污点信息无干扰的软件行为可信分析系统的部分原型，系统总体结构如图 1 所示。

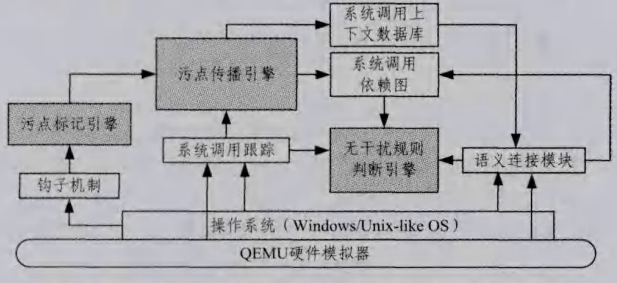


图 1 基于污点信息无干扰的软件行为可信分析系统架构

该系统基于 QEMU 硬件模拟器，通过加载不同的操作系统镜像模拟不同的操作系统环境，实现跨平台分析。系统主要由 3 部分构成：污点标记引擎、污点传播引擎和无干扰规则判断引擎。其中污点标记引擎针对不同的分析需要，将感兴趣的数据(例如键盘输入数据、磁盘输入输出数据、网卡输入输出数据等)通过 Hook 机制打上污点标记记录为污点数据，污点传播引擎从系统调用级别记录污点数据在系统内的传播和转染其它正常数据为污点数据的过程，并利用二次筛选法输出关键系统调用依赖图。系统调用上下文数据库通过链表方式记录调用过程中的系统关键状态信息、包括寄存器信息、系统调用所属的进程信息，系统调用所属的进程安全域、访问的资源所属的安全域级别等。域间无干扰规则判断引擎根据污点传播引擎生成的系统调用依赖图，结合系统调用上下文所表示的系统调用语义信息，通过判断是否满足域内一致性、域间单向性和域间一致性来判断图中是否存在不可信的污点信息传播，如果有，则报警，并将其上下文信息记录在日志中。限于篇幅，对该系统的详细介绍将在后续文献中展开。

结束语 软件动态行为的可信分析为软件可信度量研究提出了新的方向，目前也是可信计算的研究热点之一。然而由于软件构造形式以及攻击方式的多样化以，尤其是攻击者在恶意代码中加入混淆机制，以及恶意软件变种的繁多，使得对于软件关键行为的提取以及行为可信分析面临新的难题。本文尝试使用污点分析技术，结合二次筛选算法提取可能造成软件不可信的关键系统调用，生成系统动态行为模型，能够在一定程度上解决代码混淆以及变种问题，并结合污点信息流无干扰模型，在其基础上进行行为可信分析，从而形成一个统一的可信性分析模型框架。本文所提出的方法和结论对于构造新的软件动态可信分析平台具有一定的借鉴意义。尽管如此，由于信息流无干扰模型以域的概念来描述和解释访问控制和通道，而在目前主流操作系统平台(例如 Windows，Linux 等)中很难划分用户域边界，且对于进程所属的域缺乏统一的隔离机制[11]，因此在今后的工作中将进一步深入研究更加合理的刻画进程及其用户环境的方法。此外，如果待分析系统较为复杂，则生成的系统调用依赖图可能包括数量庞大的调用依赖节点，因此针对可信分析平台的实现部分仍需要做进一步优化。

参 考 文 献

[1] Challener D, Catherman R. A practical Guide to Trusted Computing [M]. 北京：机械工业出版社，2009

[2] 沈昌祥，张焕国，王怀民，等. 可信计算的研究与发展[J]. 中国科学：信息科学，2010，40(2)：139-16

[3] Newsome J, Song D. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software[C]// Proceedings of the Network and Distributed Systems Security Symposium(NDSS). Feb 2005：56-73

[4] Kang M G, Camant S M, Poosankam P, et al. DTA++：Dynamic Taint Analysis with Targeted Control-Flow Propagation [C]// Proceedings of the 18th Annual Network and Distributed System Security Symposium(NDSS). SanDiego, CA, 2011：67-81

[5] Coogan K, Lu Gen, Debray S. Deobfuscation of Virtualization Obfuscated Software A Semantics-Based Approach [C]// CCS 2011. Chicago, Illinois, USA, 2011：17-21

[6] 王蕊，冯登国，杨轶，等. 基于语义的恶意代码行为特征提取及检测方法[J]. 软件学报，2012，23(2)：378-39

[7] Li Y, Zuo ZH. An overview of object code obfuscation technologies[J]. Journal of Computer Technology and Development,

2007,17(41):125-127

[8] Ushby J. Noninterference, Transitivity, and Channel-Control Security Policies[R]. Computer Science Laboratory, SRI International, 2005

[9] Haigh J T, Yong W D. Extending the noninterference model of MLS for SAT [J]. IEEE Transaction On Software Engineering, 1987,2(13):141-150

[10] Goguen J A, Meseguer J. Security policies and security models [C]// Proc. of the 1982 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, April 1982:11-20

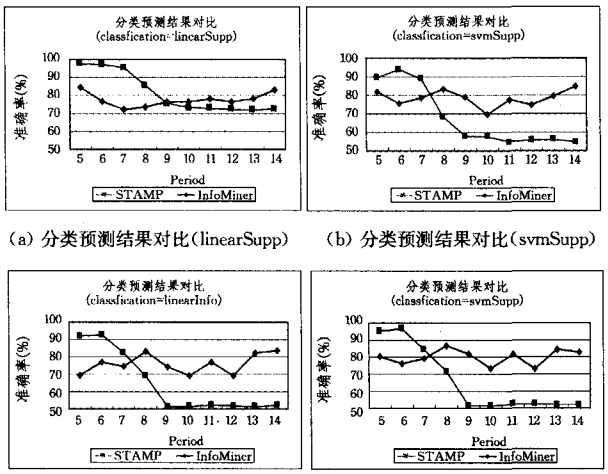
[11] 赵佳,沈昌祥,刘吉强,等. 基于无干扰理论的可信链模型[J]. 计算机研究与发展,2008,45(6):974-980

[12] 张帆,陈曙,桑永宣,等. 完整性条件下无干扰模型[J]. 通信学报,2011,32(10):11-19

[13] Bellard F. QEMU documentation [OL]. http://wiki.qemu.org/Main_Page,2011

(上接第 167 页)

定,会在一个较小的区间内波动。



(a) 分类预测结果对比(linearSupp) (b) 分类预测结果对比(svmSupp)

(c) 分类预测结果对比(linearInfo) (d) 分类预测结果对比(svmInfo)

图 8 分类预测结果对比

对所有数据集进行实验,InfoMiner 算法分类结果、STAMP 算法分类结果以及 KDD09 原始分类结果的最终对比如图 9 所示。

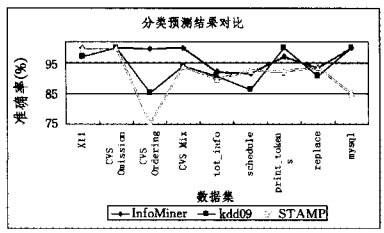


图 9 分类预测结果对比(InfoMiner,STAMP,KDD09)

从图 9 中可以看出,InfoMiner 算法的预测准确率高于 STAMP 和 KDD09,对于序列长度相对较短的模拟数据集 mysql,InfoMiner 算法的预测准确率能达到 100%;对于长序列数据集 replace 和 schedule,STAMP 算法结果与 InfoMiner 持平,且都高于 KDD09,而其他数据集 STAMP 结果是 3 种算法最差的。

实验结果表明,InfoMiner 和 STAMP 算法在挖掘模式上各具特点,InfoMiner 算法随着周期的增加,所耗费时间呈指数增长,而 STAMP 算法比较稳定。InfoMiner 算法挖掘出模式能够良好体现序列数据的特征,虽然挖掘模式的效率比较低,但其分类预测的结果很好,几乎超过了文献[11]的结果;而 STAMP 算法主要专注于序列数据中的部分周期模式,因此这方面其预测效果比 InfoMiner 差。

结束语 本文在研究了基于统计策略的两种算法 InfoMiner 和 STAMP 算法的基础上,运用卡方检验来测试模式与序列之间的相关度,并用支持度或信息增益值作为描述序列数据的特征,利用基于软件系统的历史数据和已知缺陷数据,构造了一个对已知缺陷进行分类的分类器,并以此来预测发现未知的软件缺陷。

从实验结果可以看出,利用 SVM 分类器可以取得较好的准确率,但在其训练过程所耗费的时间也占用了较大的比例。因此本文的进一步研究可以从改进序列模式挖掘算法效率以及改进分类算法效率或者利用其他分类算法两方面着手。

参 考 文 献

[1] Agrawal R, Srikant R. Mining sequential patterns [C]// Proceedings of the Eleventh International Conference on Data Engineering. Washington DC, USA: IEEE Computer Society, 1995: 3-14

[2] Chen Yuan, Shen Xiang-heng, Du Peng, et al. Research on Software Defect Prediction Based on Data Mining[C]// The 2nd International Conference on Computer and Automation Engineering. Singapore: ICCAE, 2010:563-567

[3] Yang Jiong, Wang Wei, Yu P S. Infominer: mining surprising periodic patterns [C]// Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'01). New York, USA: ACM, 2001:395-400

[4] Yang Jiong, Wang Wei, Yu P S. STAMP: on discovery of statistically important pattern repeats in long sequential data[C]// Proceedings of the Third SIAM International Conference on Data Mining(SDM'03). San Francisco, CA, USA: SIAM, 2003: 224-238

[5] 张小康. 基于数据挖掘和机器学习的恶意代码检测技术研究[D]. 合肥:中国科学技术大学,2009

[6] 周聚. 基于网络信息审计的文本过滤的研究与实现[D]. 苏州:苏州大学,2010

[7] 杨明,张载鸿. 决策树学习算法 ID3 的研究[J]. 微机发展,2002,12(5):6-9

[8] Quinlan J R. C4.5: Programs for Machine Learning[M]. San Francisco: Morgan Kaufmann Publishers, 1993

[9] 眭俊明,姜远,周志华,等. 基于频繁项集挖掘的贝叶斯分类算法[J]. 计算机研究与发展,2007,44(8):1293-1300

[10] Han Jia-wei, Kamber M. Data Mining: Concepts and Techniques [M]. San Francisco: Morgan Kaufmann Publishers, 2006

[11] Lo D, Cheng Hong, Han Jia-wei, et al. Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach [C]// Proceedings of the 15th ACM SIGKDD (KDD'09). New York, USA: ACM, 2009:557-565