一种基于无干扰的软件动态行为可信性分析方法

张 帆[1] 江 敏[2] 吴怀广[3] 徐明迪[4]
(杭州电子科技大学通信工程学院 杭州 310018)[1] (厦门大学智能科学系 厦门 361005)[2]
(郑州轻工业学院计算机与通信工程学院 郑州 450002)[3]
(武汉数字工程研究所系统软件部 武汉 430079)[4]

摘 要 软件动态行为度量是可信计算必须要解决的核心问题之一。解决这个问题有两个关键步骤:第一,对软件动态行为进行行为建模;第二,对建模后的软件动态行为进行行为可信性分析。针对第二步,即建模完成后的行为可信性分析问题进行了研究,提出一种基于无干扰的软件动态行为可信性分析方法,并从理论上给出了行为可信性判定定理。

关键词 可信计算,完整性度量,无干扰,信息流
中图法分类号 TP309 **文献标识码** A

Approach for Trust Analysis of Software Dynamic Behavior Based on Noninterference

ZHANG Fan[1] JIANG Min[2] WU Huai-guang[3] XU Ming-di[4]
(Communication School, Hangzhou Dianzi University, Hangzhou 310018, China)[1]
(Department of Cognitive Science, Xiamen University, Xiamen 361005, China)[2]
(School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China)[3]
(System Software Department, Wuhan Digital Engineering Institute, Wuhan 430072, China)[4]

Abstract Software Dynamic Behavior Measurement (SDBM) is one of the core issues that must be solved by the trusted computing. Tackling this issue has two main steps: one is to model the dynamic behavior of software; the other is to deduce the trust of the modeled behavior. A noninterference-based approach, which focuses on the second step, was presented, and the decision theorems for behavior trust analysis were given as well.

Keywords Trusted computing, Integrity measurement, Noninterference, Information flow

1 引言

度量功能是可信计算的3大核心功能(完整性度量、完整性存储和完整性报告)之一,能否实现完整的度量功能关系到整个可信计算平台能否正常运行——如果度量功能不可信,则存储和报告的内容也不可信,进而将导致整个可信计算平台不可信。因此,针对度量功能的研究是可信计算平台的重要基础研究之一。

目前的度量功能侧重于软件的静态度量,这也是可信计算组织TCG(Trusted Computing Group)所定义的度量方式,即首先计算软件当前的哈希值,并将计算所得的当前哈希值与软件的预期哈希值进行比较,如果两者一致,则认为软件是可信的;否则认为软件不可信。这种方案比较简单,也较为可行,但它存在着一个重要缺陷,即静态度量结果只能表明软件组件没有被篡改,而不能保证软件运行时没有恶意行为。举例来说,对于一个存在安全漏洞(不论有意还是无意造成的安全漏洞)的软件而言,即使其通过了静态度量,在运行时的动态行为也未必是可信的。原因很简单,一个攻击者完全可以在软件运行时,利用该软件固有的安全漏洞改变软件的控制流,从而迫使软件执行恶意操作。因此,软件动态行为度量是可信计算平台必须要解决的关键问题之一。

目前,软件动态行为度量研究只取得了初步的研究成果,还有很多问题需要深入研究[1]。根据TCG,从行为角度给出了可信定义:"一个实体是可信的,如果它的行为总是以所期望的方式,朝着预期的目标。",这意味着软件动态行为度量研究需要从两方面入手:(1)寻求合适的软件动态行为建模方法,对软件动态行为进行行为建模;(2)对建模后的软件动态行为进行推演,判定软件动态行为的可信性。在文献[2]当中,针对(1)展开了研究,提出了一种基于系统调用和进程代数CCS的行为建模方法;本文将针对(2)展开研究,提出一种基于无干扰的软件动态行为可信性推演方法。

本文的贡献如下:

到稿日期:2011-02-13 返修日期:2011-06-01 本文受国家自然科学基金(61003014/F020101),浙江省自然科学基金杰出青年团队项目(R1090138),武器装备预研基金(9140A15040211)资助。

张 帆(1977-),男,博士,讲师,主要研究方向为恶意代码分析、可信计算,E-mail:zhangfan-cixjy@163.com;**江 敏**(1976-),男,博士,助理教授,主要研究方向为形式化方法、人工智能;**吴怀广**(1976-),男,博士,讲师,主要研究方向为形式化方法、软件工程;**徐明迪**(1980-),男,博士,工程师,主要研究方向为信息安全、服务计算。

(1) 提出了一种基于无干扰的软件动态行为可信性推演方法,并从单进程行为和多进程行为两方面给出了研究结果;

(2) 本文提出的方法能够与相关软件动态行为建模方法相结合(如文献[2]等),构成一个完整的软件动态行为度量架构。

2 软件动态行为建模[2]

文献[2]给出了一种基于系统调用和进程代数 CCS 的软件动态行为建模方法。其基本结论如下。

定义 1(原子行为) 用 $Action$ 表示一个原子行为简写为 Act。一个原子行为可以表示为:

$$Act=S\rightarrow O:SC(Param_1,\cdots,Param_n)$$

式中,S 表示调用该系统调用的主体,通常就是调用该系统调用的进程;O 表示对应的客体;SC 是系统调用名;$Param_1$,…,$Param_n$ 是该系统调用的参数。

定理 1 存在一个算法,能够将所有形如 $Act=S\rightarrow O:SC(Param_1,\cdots,Param_n)$ 的原子行为转化为 SC_{ij} 的形式,即 $Act\Leftrightarrow SC_{ij}$。其中,$i$ 是原子行为 Act 当中主体 S 的进程编号;j 是从传值 CCS 到纯 CCS 转换时的枚举编号。

定义 2 行为的递归定义如下:(1)原子行为是行为;(2)行为通过以下 5 种运算之后得到的仍然是行为;(a)顺序行为:$E_1.E_2$;(b)非确定性选择行为:E_1+E_2;(c)并行行为:$E_1|E_2$;(d)屏蔽行为:$E_1\backslash L$;(e)隐藏行为:E_1/L。

进一步,利用 CCS 从操作语义可以证明,当系统运行后,上述行为(b)、(c)会退化为顺序行为(a)的形式[2]。因此有下面定义 3。

定义 3(系统行为 SCSS, System Call Sequences of System) 可以证明,对外部观察者而言,一个系统的行为表现为系统调用的顺序执行序列,即 $SCSS=SC_{11}\cdots SC_{ij}\cdots SC_{mn}.\underline{0}$,其中 $1\leqslant i\leqslant m;1\leqslant j\leqslant n;i,j\in N$。

定理 2 利用系统行为 $SCSS$,结合 CCS 基本操作符,能够获取系统中任意软件的动态行为。

结合定义 3 和定理 2,就能够对一个系统内的任意软件行为进行建模表达,即首先根据定义 3 获取系统行为 $SCSS$,然后,利用 CCS 操作符(参见文献[3,4]和定义 2)对 $SCSS$ 进行行为变换,从而获取系统内任意软件(软件组)的任意行为。

本节更详细的建模过程和示例请参见文献[2]。

3 基于无干扰的软件动态行为可信性分析

3.1 无干扰基本定义

本节给出利用无干扰模型,对第二部分建模得到的软件动态行为进行可信性分析的方法。在介绍之前,先依据 Rushby[5] 的工作给出基本无干扰定义,为了理解的方便,本文基本遵从其符合定义而不做大的变动。

定义 4 一个机器 M 由以下部分构成:(1)一个状态集 S,初始状态是 s_0。(2)一个动作集 A。动作集中的每一个动作都是原子动作 SC_{ij}。(3)一个输出集 O。(4)一个单步函数 $step:S\times A\rightarrow S$。(5)一个输出函数 $output:S\times A\rightarrow O$。(6)运行函数 $run:S\times A^*\rightarrow S$。其中,$A^*$ 是动作的序列,能够把 run 表示成右递归的形式:

$$\begin{cases} run(s,\Lambda)=s, \\ run(s,a\circ\alpha)=run(step(s,a),\alpha) \end{cases}$$

式中,Λ 表示空动作序列,$\circ$ 表示连接操作。(7)一个域集 D。指明了动作 a 所属的安全域。(8)一个安全域到动作的映射函数为 $dom:A\rightarrow D$。(9)一个自反标记 $\rightsquigarrow$,用来表示安全策略。$\rightsquigarrow$ 的补关系 $\not\rightsquigarrow$ 称为无干扰关系。

定义 5 定义具有结构化状态的机器如下:(1)一个名字集 N,给出了机器所有存储单元的名字集合。(2)一个值集 V。每一个存储单元 $n\in B$ 的单元值集合构成了值集。(3)一个内容函数 $contents:S\times N\rightarrow V$。在状态 s 内存单元 n 的值为 v。(4)一个观察函数 $observe:D\rightarrow P(N)$。(5)修改函数 $alter:D\rightarrow P(N)$。观察函数和修改函数给出了某个特定的安全域 $u\in D$ 所能观察和修改的机器内存单元的集合,其中 P 是幂集函数。

定义 6(引用监视器假设 RMA, Reference Monitor Assumption)

(1)假设 1:等价关系在 RM 下的解释。两个状态如果在安全域 u 等价,则对安全域 u 所能观察到的任何机器单元,其单元值都必须相等:

$$s\overset{u}{\sim}t \text{ iff } (\forall n\in observer(u).\ contents(s,n)=contents(t,n))$$

(2)假设 2:两个状态和如果等价 $s\overset{u}{\sim}t$,则 s 和 t 在更改 n 时必须把 n 更改为同样的值:

$$s\overset{dom(a)}{\sim}t\wedge[contents(step(s,a),n)\neq contents(s,n)\vee contents(step(t,a),n)\neq contents(t,n)]\supset contents(step(s,a),n)=contents(step(t,a)n)$$

(3)假设 3:如果动作 a 修改了名字 n 的值,则 a 所属的安全域 $dom(a)$ 必须对名字有修改权限,令 $N=alter(dom(a))$,有:

$$contents(step(s,a),n)\neq contents(s,n)\supset n\in N)$$

下面的 3.2 节和 3.3 节分别从单进程和多进程的角度给出了软件动态行为可信性分析方法。

3.2 单进程动态行为分析

单进程行为考察的是单个进程的动态行为。显然,单进程的信息传递都是在该进程内部进行的,因此,单进程行为本质上是一种传递无干扰关系。换句话说,假设有 3 个安全域 $u,v,w\in D$,若有安全策略 $u\rightsquigarrow$ 和 $v\rightsquigarrow w$,则显然 $u\rightsquigarrow w$ 也是成立的,即信息具有传递性。

首先,假设已经从 $SCSS$ 获得了单进程行为 E_{P_1}:$E_{P_1}=SCSS/L$,其中 $L=\{SC_{ij}\mid i\neq 1,j\in N\}$。接下来,对 E_{P_1} 进行行为可信性判定。

定义 7 纯化函数 $purge(\alpha,u)$ 如下:

$purge(\Lambda,u)=\Lambda$(Λ 为空动作,下同)

$$purge(\alpha\circ a,u)=\begin{cases} a\circ purge(\alpha,u), & \text{if } dom(a)\rightsquigarrow u \\ purge(\alpha,u), & \text{otherwise} \end{cases}$$

$purge(\alpha,u)$ 在安全策略 $\rightsquigarrow$ 控制下,将对安全域 u 应该无干扰($dom(a)\not\rightsquigarrow u$)的动作 a 进行删除得到新的动作序列。

定义 8 满足如下特性的等价关系 $\overset{u}{\sim}$:

(1)输出一致性:

$s\overset{u}{\sim}t\supset output(s,a)=output(t,a)$,其中 $a\in u$;

(2)单步一致性:

$s\overset{u}{\sim}tstep(s,a)\overset{u}{\sim}step(t,a)$,其中 $dom(a)\rightsquigarrow u$。

文献[5]中使用符号⊐表示蕴含关系↛，本文也使用同样的符号表达。

定义 9 传递无干扰完整性判定等式。

$$output(run(s_0,\alpha),a)=output(run(s_0,purge(\alpha,dom(a))),a) \quad (1)$$

根据可信计算组织 TCG 的定义，要判定软件行为是否可信，需要考察两个行为：软件实际执行行为和软件的预期执行行为。其中，式(1)左边表达了软件从初始状态开始，实际执行行为 α 以后的结果；式(1)右边表达了软件在安全策略控制下的预期执行行为。如果等式成立，则说明两个行为是一致的，根据 TCG 定义，软件动态行为可信；否则不可信。这就是式(1)的含义。

定理 3 如果一个单进程行为满足如下 3 个条件：

(1)输出一致性：

$s \overset{u}{\sim} t \sqsupset output(s,a)=output(t,a)$

(2)单步一致性，当 $dom(a) \rightsquigarrow u$ 时有：

$s \overset{u}{\sim} t \sqsupset step(s,a) \overset{u}{\sim} step(t,a)$

(3)局部干扰性，当 $dom(a) \not\rightsquigarrow$ 时有：

$dom(a) \not\rightsquigarrow u \sqsupset s \overset{u}{\sim} step(s,a)$

则任意一步执行都符合定义 9 当中的完整性无干扰等式，换句话说，该单进程动态行为是可信的。

证明：只需要证明式(2)成立，再利用输出一致性即可证明完整性无干扰等式成立：

$$run(s_0,\alpha) \overset{u}{\sim} run(s_0,purge(\alpha,dom(a))) \quad (2)$$

上面的式(2)又可以转化为证明式(3)成立：

$$s \overset{u}{\sim} t \sqsupset run(s,\alpha) \overset{u}{\sim} run(t,purge)(\alpha,u)) \quad (3)$$

如果式(3)成立，令 $s=t=s_0$ 即可得式(2)成立。而要证明式(3)，只需要对 α 的长度做归纳即可。具体证明略。

定理 4 将一个单进程行为表达为结构化机器 M，若 M 满足定义 6 的 RMA，且符合如下两个条件，则该单进程行为是动态可信的。

(1)$u \rightsquigarrow u \sqsupset observe(u) \subseteq observe(v)$；

(2)$n \in alter(u) \wedge n \in observe(v) \sqsupset u \rightsquigarrow v$。

证明：只要证明该单进程动态行为满足定理 3 的 3 个基本条件即可。

首先，证明满足输出一致性，这由 M 符合 RMA，并利用 RMA 假设 1 立即可得。

其次，证明满足单步一致性。根据 RMA 假设 1，单步一致性可以等价改写为：

$$\forall n \in oberve(u).[contents(s,n)=contents(t,n)] \sqsupset \forall n \in observe(u).[contents(step(s,a),n)=contents(step(t,a),n)] \quad (4)$$

然后分情况讨论证明式(4)成立。在证明过程中，要依次用到 RMA 假设 2、假设 3 和假设 1，以及式(4)的前条件。

最后证明满足局部干扰性。只要证明逆否命题即可：$s \overset{u}{\not\sim} step(s,a) \sqsupset dom(a) \rightsquigarrow u$。利用式(3)的前条件，以及 RMA 的假设 3 即可得证。

具体证明略。

3.3 多进程动态行为分析

多进程行为表达的是不同进程间的交互行为。多进程行为可以认为是一种非传递无干扰关系。假设有 3 个进程 p_1、p_2 和 p_3，如果安全策略规定 p_1 和 p_2 之间可以有信息流，p_2 和 p_3 之间可以有信息流，则 p_1 和 p_3 之间是否可以有信息流是不确定的，这取决于实际的安全策略。如果 p_1 和 p_2 之间可以有信息流，那么这就退化为 3.2 节当中的传递无干扰；如果 p_1 和 p_2 之间没有信息流，那么这就是非传递无干扰情形。本节将对其进行讨论。

在非传递无干扰的条件下，定义 7 当中的纯化 $purge(\alpha,u)$ 函数太过严格了。若有安全策略 $u \rightsquigarrow v$ 和 $v \rightsquigarrow w$，则 $purge(\alpha,u)$ 实际上将 $u \rightsquigarrow v$ 也变成了 $u \not\rightsquigarrow v$。因此，要定义更为弱化的弱纯化函数 $ipurge(\alpha,u)$。在定义弱纯化函数之前，先给出域集函数的定义，域集函数 $sources$ 用来抽取计算 $ipurge(\alpha,u)$ 时所有具有信息流传递的安全域，即若有 $u \rightsquigarrow v_1, v_1 \rightsquigarrow v_2, \cdots, v_{n-1} \rightsquigarrow v_n, v_n \rightsquigarrow w$ 及 $u \not\rightsquigarrow w$，则抽取上述信息流传递的所有安全域 $sources=\{u,v_1,\cdots,v_n,w\}$。

定义 10 域集函数 $sources: A^* \times D \to P(D)$。

$sources(\Lambda,u)=u$

$$sources(a\circ\alpha,u)=\begin{cases}\mathcal{D}_1, & \text{if } \exists v.\, v \in sources(\alpha,u) \wedge dom(a) \rightsquigarrow v \\ \mathcal{D}_2, & \text{otherwise}\end{cases}$$

式中，$\mathcal{D}_1=sources(\alpha,u) \cup \{dom(a)\}$，$\mathcal{D}_2=\text{sources}(\alpha,u)$。

定义 11 $ipurge(\alpha,u): A^* \times D \to A^*$。

$ipurge(\Lambda,u)=u$

$$ipurge(a\circ\alpha,u)=\begin{cases}\mathcal{A}_1, & \text{if } dom(a) \in sources(a\circ\alpha,u) \\ \mathcal{A}_2, & \text{otherwise}\end{cases}$$

式中，$\mathcal{A}_1=a\circ ipurge(\alpha,u)$，$\mathcal{A}_2=ipurge(\alpha,u)$。

定义 12 非传递无干扰完整性判定等式。

$$output(run(s_0,\alpha),a)=output(run(s_0,ipurge(\alpha,dom(a))),a) \quad (5)$$

定义 13 弱单步一致：在 $dom(a) \rightsquigarrow v$ 的条件下：$s \overset{dom(a)}{\sim} t \wedge s \overset{v}{\sim} t \sqsupset step(s,a) \overset{v}{\sim} step(t,a)$。

接下来，类似于定理 3 和定理 4，在定理 5 和定理 6 中对应给出多进程行为的无干扰判定定理。

定理 5 如果多进程动态行为满足如下 3 个条件，则该多进程软件动态行为是可信的。

(1) 输出一致性：

$s \overset{u}{\sim} t \sqsupset output(s,a)=output(t,a)$

(2)弱单步一致，当 $dom(a) \rightsquigarrow u$ 时有：

$s \overset{dom(a)}{\sim} t \wedge s \overset{u}{\sim} t \sqsubset step(s,a) \overset{u}{\sim} step(t,a)$

(3)局部干扰性，当 $dom(a) \not\rightsquigarrow u$ 时有：

$dom(a) \not\rightsquigarrow u \sqsupset s \overset{u}{\sim} step(s,a)$

证明：引入域集等价关系 $\overset{C}{\approx}$ 如下：

$s \overset{C}{\approx} t$ iff $\forall u \in C.\, s \overset{u}{\sim} t$。利用 $\overset{C}{\approx}$ 证明在弱单步一致性和局部干扰性条件下，下面的式(6)成立：

$$s \overset{sources(\alpha,u)}{\approx} t \sqsupset run(s,\alpha) \overset{u}{\sim} run(t,ipurge(\alpha,u)) \quad (6)$$

令 $s=t=s_0$，代入式(6)，再利用输出一致性即可得。

而要证明式(6)成立，只要对 α 的长度做归纳，并利用如下两个结论即可证明：

(下转第 114 页)

[5] Julien C, Payton J, Roman G C. Reasoning About Context-Awareness in the Presence of Mobility [A]// Proc. of the 2nd International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA 2003) [C]. Marseille, France, 2003: 259-276

[6] Roman G C, Julien C, Payton J. A Formal Treatment of Context Awareness [C] // Wermelinger M, Margaria-Steffen T, eds. FASE 2004. Vol. 2984 of LNCS. Springer Heidelberg, 2004: 12-36

[7] Yan L, Sere K. A Formalism for Context-Aware Mobile Computing [A]// Proceedings of the IEEE 3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks [C]. Cork, Ireland, 2004: 14-21

[8] Cafezeiro I, Haeusler E H. Semantic interoperability via category theory[C]// John Grundy, Sven Hartmann, Alberto H. F. Laender, Leszek Maciaszek, and John F. Roddick, eds. Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling . Vol. 83 (ER '07). Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2007: 197-202

[9] Milner R. The Space and Motion of Communicating Agents [M]. Cambridge: CambridgeUniversity Press, 2009

[10] Krivine J, Milner R, Troina A. Stochastic bigraphs [A]// Bauer A, Mislove M, eds. Proceedings of the 24th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2008) [C]. volume 218 of Electronic Notes in Theoretical Computer Science, Elsevier, 2008: 73-96

[11] Milner R. Pure bigraphs: Structure and dynamics [J]. Information and Computation, 2006, 204 (1): 60-122

[12] Leifer J J, Milner R. Transition systems, link graphs and Petri nets [J]. Mathematical Structures in Computer Science, 2006, 16 (6): 989-1047

[13] Milner R. Axioms for bigraphical structure [J]. Mathematical Structures in Computer Science, 2005, 15(6): 1005-1032

[14] Jensen O H. Mobile processes in bigraphs [D]. Dep. of Computer Science, Aalborg University, 2006

[15] Birkedal L, Debois S, Elsborg E, et al. Bigraphical models of context-aware systems [A] // Aceto L, Ingólfsdóttir A, eds. Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'06) [C]. Lecture Notes in Computer Science. vol. 3921, Springer-Verlag, 2006: 187-201

[16] Birkedal L, Debois S, Elsborg E, et al. Bigraphical Models of Context-aware Systems [R]. 74. IT Univ. of Copenhagen, 2005

[17] Milner R. Seminar Notes on developments in Bigraphs. Universities of Cambridge and Edinburgh [EB/OL] http://www.cl.cam.ac.uk/archive/rm135/uam-theme.html, 2009

(上接第 103 页)

$$s \overset{sources(a,u)}{\approx} t \supset step(s,a) \overset{sources(a,u)}{\approx} step(t,a)\, dom(a) \notin sources$$

$$(a \circ \alpha, u) \supset s \overset{sources(\alpha,u)}{\approx} step(s,a)$$

具体证略。

定理 6 将一个多进程行为与一个机器 M 对应，如果 M 满足定义 6 的 RMA，且满足条件：

$$n \in alter(u) \wedge n \in observe(u) \supset u \rightsquigarrow v$$

则该多进程动态行为是可信的。

证明：利用定理 5 来证明定理 6。

首先证明满足输出一致性。输出一致性由定义 6 的 RMA 假设 1 即可得。

其次证明满足弱单步一致性。弱单步一致性根据定义 6 的 RMA 假设 1，可以改写为：

$$s \overset{dom(a)}{\sim} t \wedge \forall n \in observe(u).\ contents(s,n) = contents(t,n) \supset \forall n \in observe(u).\ contents(step(s,a),n) = contents(step(t,a),n) \quad (7)$$

分情况讨论上述式(7)成立即可。在证明的过程中，要依次用到 RMA 假设 2 以及式(7)的前条件。

最后证明局部干扰性成立。证明逆否命题 $s \overset{u}{\not\sim} step(s,a) \supset dom(a) \rightsquigarrow u$ 即可。利用式(5)的前条件，以及 RMA 的假设 3 立即可得。

具体证略。

结束语 软件动态行为度量是可信计算必须要解决的关键问题之一，也是目前国内外的研究热点。但是，由于软件形式的多样性和攻击方式的复杂性，使得目前这方面的研究还处于初步阶段，还有很多问题需要深入研究[1]。

软件(或系统)在运行的过程中必然要涉及到信息的流动，因此，无干扰模型作为一种重要的信息流模型，在可信计算相关的研究中，特别是可信链的研究当中，获得了广泛的应用。本文也尝试以无干扰模型为基础，对建模后的软件动态行为进行分析，提出了一种基于信息流的软件动态行为可信性分析方法，并从单进程行为和多进程行为的角度分别给出了软件动态行为可信性的判定定理。本文的结论对于建模完成后的软件动态行为可信性推演具有参考意义，也能够与相应的软件动态行为建模方法相结合从而构成一个完整的软件动态行为度量框架。

参 考 文 献

[1] 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)：139-166

[2] Zhang F, Xu M D, You L. A Behavior Modeling Method based on System Call and Algebra Process CCS[J]. Proceedings of the 5th China Trusted Computing and Information Security(CTCIS'11), Journal of Wuhan University (Science Edition)

[3] Milner R. Communication and Concurrency[M]. London: Prentice-Hall, 1989

[4] Focardi R, Gorrieri R. Classification of Security Properties: Part I Information Flow[J]. Foundations of Security Analysis and Design, LNCS, 2001, 2171: 331-396

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