ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.7, July 2006, pp.1601−1608
DOI: 10.1360/jos171601

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一个非确定系统的不干扰模型*

谢 钧[1+], 黄 皓[2]

[1](解放军理工大学 指挥自动化学院,江苏 南京 210007)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

A Noninterference Model for Nondeterministic Systems

XIE Jun[1+], HUANG Hao[2]

[1](Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-80828023, E-mail: xiejun73@263.net

Xie J, Huang H. A noninterference model for nondeterministic systems. *Journal of Software*, 2006,17(7): 1601−1608. http://www.jos.org.cn/1000-9825/17/1601.htm

Abstract: The noninterference concept for actions of system to information domains is proposed. On the basis of this concept, the noninterference model is extended to nondeterministic systems. The noninterference concept based on actions of system simplifies the "purge" of the action sequence of the system. As a result, this model has concise unwinding conditions which are easy to understand and use. The extended model can be used to verify not only static but also dynamic information flow policies. Finally, a dynamic label based access control model is designed, in which the concrete semantic of the actions such as read, write and execute are defined, and its security is verified by the noninterference model.

Key words: noninterference model; information security; security model; access control model; information flow

摘 要: 提出系统动作对信息域的不干扰概念,并在此基础上将不干扰模型推广到非确定系统.由于基于系统动作的不干扰概念简化了系统动作序列的提取操作,该模型的单步展开条件具有简洁的形式并易于理解和使用.推广后的不干扰模型不仅能够验证静态信息流策略,还可以验证各种动态信息流策略.最后设计了一个基于动态标记的访问控制模型,并在该模型中定义了读、写、执行等操作的具体语义,然后利用不干扰模型对其安全性进行了形式化验证.

关键词: 不干扰模型;信息安全;安全模型;访问控制模型;信息流

中图法分类号: TP309 文献标识码: A

安全模型用形式化的方法来描述信息系统的安全特性,是进行形式化安全验证的基础.访问控制模型(如著名的 BLP 模型)具有直观且易于实现和验证的优点,这类模型通过一个不能被篡改和绕过的"访问监控器"控制所有主体对客体的读写操作.但如何确定系统的主客体和读写操作,却并不像表面上那样简单.例如,当主体试图写一个不存在的文件时,如果访问监控器返回文件不存在的错误信息,则可能会导致一个隐蔽信道使得高密

* Supported by the National Natural Science Foundation of China under Grant No.60473093 (国家自然科学基金)
Received 2004-08-22; Accepted 2006-01-09

级的“木马”程序通过有规律地创建和删除一个高密级文件,将高密级信息泄漏给低密级的用户.由于访问控制模型过于简单地用主客体和读写操作来描述信息流,使得这些隐蔽信道被隐藏或忽略.一些信息流模型(如文献[1])虽然用形式化方法验证访问控制模型的信息流策略,但每个系统操作和事件所产生的信息流完全基于主观判断和假设,并未验证.而不干扰模型[2,3]将信息流理解为:如果对 a 的行为的任意更改都不影响 b 的系统视图,则没有信息从 a 流向 b.显然,不干扰模型的概念更接近信息流的实质.因此,该模型能够很好地排除各种存储隐蔽信道问题(若要排除时间隐蔽信道,则在系统动作中必须考虑时间因素).近几年,不干扰模型广泛应用于基于语言的信息流安全[4−6].

虽然不干扰模型能够很好地应用于确定系统中,但它在非确定系统中的推广要么过于简单,要么太复杂和抽象而难以应用[7].本文将不干扰概念从信息域推广到系统动作,提出了系统动作对信息域的不干扰概念,并在此基础上给出了基于单步状态的安全展开条件(unwinding conditions).该条件与其他模型相比更加简洁且易于使用.同时,推广后的不干扰模型可以验证各种动态信息流策略,如汇聚流、分发流等[1].最后,本文设计了一个基于动态标记的访问控制模型,并用不干扰模型对其安全性进行了形式化验证,为该模型的使用提供了实例.

1 一个非确定系统的不干扰模型

我们先用一个状态转换机来描述一个非确定系统,然后给出该系统的信息流策略的安全性定义以及单步展开条件.

在一个非确定系统中,对于同一输入序列在相同起始状态下可以产生不同的输出序列.在我们定义的系统中,系统执行序列不仅考虑系统的输入、输出,还包括系统内部的动作.与文献[8]类似,为简化问题的复杂性,我们对系统的非确定性因素进行了限制:系统对可执行动作的选择是不确定的,但每个动作执行的结果是确定的.由于执行同一输入序列的过程中系统内部执行动作可能是不同的,会产生不同的输出序列,因此,系统仍然是非确定的,如各种并发系统.对于系统动作执行结果不确定的状态有限的系统,可以认为系统选择了不同的内部执行路径从而产生了不同的结果,因此,可以通过引入适当的内部执行动作将其转换为系统动作执行结果确定的系统.由于系统动作执行结果确定,可以用确定状态机来描述该系统.

定义 **1**. 系统 M 包括以下元素:

- 系统状态集合 S,系统的初始状态 $s_0\in S$;
- 系统信息域集合 D,系统的安全策略由信息域上的信息流关系表示,信息域到客体和主体的映射在不同的系统内可以有不同的解释;
- 系统动作集合 A,包括系统执行的输入、输出、命令、指令等,可以外部产生也可以是内部的响应;
- 系统信息域的值域集合 V;
- 信息域取值函数 $value:S\times D\to V$,信息域取值表示系统状态在信息域的投影,可以映射为单个或多个主客体的状态,相当于其他模型中的域视图函数;
- 动作可执行关系 $ENABLED\subset S\times A$,$enabled(s,a)\stackrel{\text{def}}{\Longleftrightarrow}(s,a)\in ENABLED$,表示在状态 s 下动作 a 可执行;
- 单步状态转换函数 $step:ENABLED\to S$,$step(s,a)$表示在状态 s 下执行动作 a 后的状态,虽然系统选择执行的动作是不确定的,但每个动作执行的结果是确定的;
- 系统在状态 s 下可能的执行序列集合 T_s,该集合与系统状态有关,在不同状态下系统可能产生的执行序列是不同的:$\varepsilon\in T_s$,$a\circ\alpha\in T_s\Leftrightarrow enabled(s,a)\wedge\alpha\in T_{step(s,a)}$,$\varepsilon$表示空串,$\circ$表示串连接符;
- 系统运行偏函数 $run:S\times A^*\to' S$,当$\alpha\in T_s$时,$run(s,\alpha)$有定义:$run(s,\varepsilon)=s$,$run(s,a\circ\alpha)=run(step(s,a),\alpha)$;
- 动作执行域偏函数 $dom:A\to' D$,安全策略并不关心所有执行实体间的信息流关系,因此,dom 并不一定要求对所有动作有定义.

为引出系统信息流策略的安全性定义,我们先定义系统动作对信息域的不干扰概念.

定义 **2**. 对任意系统状态 s,若存在关系$\not\sim_s\subset A\times D$ 使系统满足$\forall i\in D,\alpha\in A^*:\alpha\in T_{s_0}\to\alpha|_{s_0,i}\in T_{s_0}\wedge value(run(s_0,\alpha),i)=value(run(s_0,\alpha|_{s_0,i}),i)$,则$\not\sim_s$是系统 M 的系统动作对信息域的不干扰关系,即 $a\not\sim_s i$ 表示动作 a 在状态 s 下执行

不会影响域 i,或者说不干扰关系$\not\triangleright_s$的定义对系统 M 是安全的.

其中,$\alpha|_{s,i}$表示可执行序列的提取(purge)操作.如果$\alpha\in T_s$,$\alpha|_{s,i}$定义如下:

$$\varepsilon|_{s,i}=\varepsilon,$$

$$(a\circ\alpha)|_{s,i}=\begin{cases}\alpha|_{step(s,a),i}, & \text{if } \not\triangleright_s\ i\\ a\circ(\alpha|_{step(s,a),i}), & \text{otherwise}\end{cases}$$ □

从该系统的某一执行序列中删除所有不影响域 i 的动作所得到的序列,应该仍然是该系统的可执行序列,并且在执行完这两个序列后,域 i 的状态应该是一样的.需要注意的是:从某一执行序列中删除部分不影响域 i 的动作所得到的序列,不一定是该系统的可执行序列,即$\neg(a\circ\alpha\in T_{s_0}\wedge a\not\triangleright_s i\rightarrow\alpha\in T_{s_0})$,因为可能某些系统动作的执行要依赖那些被删除了的动作.而删除所有不影响域 i 的动作所得到的序列,应该是该系统的执行序列.假如该序列不是可执行序列,则其中必存在某动作 a 不可执行,若是因为删除动作 b 使得 a 不可执行,显然 b 影响了 a 的执行,而 a 是影响域 i 的,则 b 会影响域 i,显然是矛盾的.

由于动作的不干扰关系与系统执行该动作的状态有关,故对执行序列的提取操作必须考虑每一步的状态.

现在,我们在系统动作的不干扰概念的基础上考虑系统信息流策略的安全性.我们用 D 上的自反传递关系$\rightsquigarrow$表示信息域间的信息流安全策略(简称为策略$\rightsquigarrow$),即 $x\rightsquigarrow i$ 表示允许信息从域 x 流入域 i.由信息流的意义,我们认为当来自域 x 的所有动作都不会影响域 i 时,则没有信息从域 x 流入域 i.因此,若不允许信息从域 x 流入域 i,则要求来自 x 的所有动作都不能影响域 i 是安全的.

定义 **3**. 若$\not\triangleright_s$是系统 M 的系统动作对信息域的不干扰关系,且系统满足$\forall x,i\in D,a\in A,s\in S:\neg(x\rightsquigarrow i)\wedge dom(a)=x\rightarrow a\not\triangleright_s i$,则系统 M 强制策略$\rightsquigarrow$,或策略$\rightsquigarrow$对系统 M 是安全的.

将不干扰模型推广到非确定系统的关键是:对系统的可执行序列进行提取操作后,还能得到该系统的一个可执行序列,若删除条件改为$\neg(dom(a)\rightsquigarrow i)$(如文献[3]),则这一要求可能得不到保证,因为有些内部执行动作并不是信息域的动作,但在某些状态下可能依赖于被删除的动作.若只考虑输入输出事件(如文献[9]),只删除所有高级输入事件,则得到的不一定是一个可接收序列,可能某高级输出事件依赖于某高级输入事件,但若删除所有高级事件,则会限制低级输入对高级输出的影响,因此面临两难.引入动作的不干扰概念可方便地删除所有无关动作,而无论它是外部随机产生的或是内部的响应动作,将证明策略的安全性转移到证明动作对域的不干扰关系定义的安全性.

为便于系统验证,现给出并证明只涉及单步状态的系统安全展开条件.

定理 **1**. 对于$\forall s,t\in S,\forall i\in D,\forall a\in A$,如果存在某等价关系$\sim^i$满足下列条件,则$\not\triangleright_s$的定义对系统 M 是安全的:

(1) $s\sim^i t\rightarrow value(s,i)=value(t,i)$;

(2) $s\sim^i t\wedge enabled(s,a)\wedge enabled(t,a)\rightarrow step(s,a)\sim^i step(t,a)$;

(3) $s\sim^i t\wedge enabled(s,a)\wedge\neg enabled(t,a)\rightarrow a\not\triangleright_s i$;

(4) $a\not\triangleright_s i\wedge enabled(s,a)\rightarrow s\sim^i step(s,a)$.

$s\sim^i t$ 的实际意义表示:对于域 i,状态 s 和 t 是完全一样的.条件(1)表示,两状态对 i 等价则这两系统状态在域 i 的投影一样;条件(2)表示,在两等价状态下执行相同动作后的状态仍然等价;条件(3)表示,对两等价状态,在一状态下某动作可执行而在另一状态下该动作不可执行,则该动作必不干扰 i.只有这样,才能保证对 i 两状态完全一样;条件(4)表示,执行不干扰 i 的动作后对 i 来说状态无变化.

证明:由于 $s_0\sim^i s_0$,若能证明 $s\sim^i t\wedge\alpha\in T_s\rightarrow\alpha|_{s,i}\in T_t\wedge run(s,\alpha)\sim^i run(t,\alpha|_{s,i})$,则定理得证.

对α的长度进行归纳证明:当$\alpha=\varepsilon$时,命题显然成立.假设当α时命题成立,现证明当 $a\circ\alpha$时命题也成立.假设

$$s\sim^i t\wedge a\circ\alpha\in T_s \tag{1}$$

对 a 分情况讨论:

情况(1):若

$$a\not\triangleright_s i \tag{2}$$

则

$$(a\circ\alpha)|_{s,i}=\alpha|_{step(s,a),I} \tag{3}$$

由公式(1),有

$$enabled(s,a) \tag{4}$$

由上述条件(4)和公式(2)、公式(4),有

$$s\sim^{i}step(s,a) \tag{5}$$

由公式(1)、公式(5),有

$$step(s,a)\sim^{i}t \tag{6}$$

由公式(1),有

$$\alpha\in T_{step(s,a)} \tag{7}$$

由归纳假设和公式(6)、公式(7),有

$$\alpha|_{step(s,a),i}\in T_t\wedge run(step(s,a),\alpha)\sim^{i}run(t,\alpha|_{step(s,a),i}) \tag{8}$$

由公式(3)、公式(8),有

$$(a\circ\alpha)|_{s,i}\in T_t\wedge run(s,a\circ\alpha)\sim^{i}run(t,(a\circ\alpha)|_{s,i}).$$

情况(2):若

$$\neg(a\succ_s i) \tag{9}$$

则

$$(a\circ\alpha)|_{s,i}=a\circ(\alpha|_{step(s,a),i}) \tag{10}$$

由条件(3)和公式(1)、公式(9),有

$$s\sim^{i}t\wedge enabled(s,a)\wedge enabled(t,a) \tag{11}$$

由条件(2)和公式(11),有

$$step(s,a)\sim^{i}step(t,a) \tag{12}$$

由公式(1),有

$$\alpha\in T_{step(s,a)} \tag{13}$$

由归纳假设和公式(12)、公式(13),有

$$\alpha|_{step(s,a),i}\in T_{step(t,a)}\wedge run(step(s,a),\alpha)\sim^{i}run(step(t,a),\alpha|_{step(s,a),i}) \tag{14}$$

根据定义,

$$\alpha|_{step(s,a),\,i}\in T_{step(t,a)}\rightarrow a\circ(\alpha|_{step(s,a),i})\in T_t \tag{15}$$

由公式(10)、公式(14)、公式(15),有

$$(a\circ\alpha)|_{s,i}\in T_t\wedge run(s,a\circ\alpha)\sim^{i}run(t,(a\circ\alpha)|_{s,i})$$

由情况(1)、情况(2)证明:当 $a\circ\alpha$时命题也成立,即定理得证. □

2 在动态信息流策略上的推广

由于系统动作对信息域的不干扰关系与系统状态有关,因此该模型的信息流策略的安全性定义可以方便地推广到与系统状态相关的动态信息流策略.

定义 **4**. 若$\succ_s$是系统 M 的动作对域的不干扰关系,且 M 满足$\forall x,i\in D,a\in A,s\in S:\neg(x\rightsquigarrow_s i)\wedge dom(a)=x\rightarrow a\succ_s i$,则策略$\rightsquigarrow_s$对系统 M 是安全的.$x\rightsquigarrow_s i$ 表示在状态 s 下允许信息从域 x 流入 i.

在实际应用特别是在商业领域中,需要一些动态的信息流策略,即信息能否从 A 流入 B 取决于当时的系统状态.典型地如汇聚策略:A 和 B 的信息不能汇聚于 C.而广泛应用于商业领域的“中国墙策略”就是一种汇聚策略.用动态标记方法可以实现这样的策略:

- 域互斥表 $M:D\rightarrow\boldsymbol{P}(D\times D),(a,b)\in M(c)$表示 a 和 b 的信息不能汇聚于 c;
- 动态标记 $l:S\times D\rightarrow\boldsymbol{P}(D),\forall a\in D:l(s_0,a)=\{a\}$,动态标记记录实体包含的信息类;
- 当满足下列条件时才允许执行导致信息从 a 流入 b 的动作:

$$\neg(\exists x,y:x\in l(s,a)\wedge y\in l(s,b)\wedge(x,y)\in M(b));$$

- 当执行了导致信息从 a 流入 b 的动作后必须满足:

$$l(s',b)=l(s,a)\cup l(s,b),s'\text{为执行动作后的状态;}$$

- 信息流策略:

$$a\rightsquigarrow_s b \underline{\underline{\text{def}}} \neg(\exists x,y:x\in l(s,a)\wedge y\in l(s,b)\wedge(x,y)\in M(b)).$$

虽然用动态标记法可以很方便地表达这类策略,但设计者必须保证对系统动作的信息流假设的正确性,这可以由我们的不干扰模型来验证.而这类安全策略用其他不干扰模型都无法进行验证.

3 一个动态标记访问控制模型及其安全性验证

为了说明该不干扰模型的易用性,我们设计了一个简单、可行的操作系统访问控制模型,并用该不干扰模型对其安全性进行了验证.

该访问控制模型主要用于保证信息的机密性,强制高密级的输入信息不能流向低密级的输出,将各种输入、输出作为一类客体,系统内部还包括其他临时客体和处理输入、输出的进程.由于机密的泄漏必须通过输入、输出客体,因此,该模型的信息流策略不考虑内部实体间的信息流动.模型允许进程的访问控制标记动态改变,而客体的访问控制标记与客体名称绑定.虽然模型简单,但具有一般操作系统的很多特性,如创建和终止进程等.与 BLP 等访问控制模型的最大不同在于:该模型对系统读、写、执行等操作定义了具体的语义,从而可以有效地排除各种存储隐蔽信道.

定义 **5**. 系统 M'在定义 1 的基础上增加以下元素:

- 系统实体集合 E_s,与系统状态相关,包括客体和主体,实体可动态创建与删除;
- 系统进程集合 P_s,$P_s\subset E_s$;
- 系统进程名集合 N_p,$P_s\subseteq N_p$,为描述简单起见,用实体名表示实体;
- 系统客体集合 O_s,$O_s\subset E_s$,除 I/O 客体外还包括系统内部动态客体,如文件、内存单元等,系统满足:

$$(E_s=P_s\cup O_s)\wedge(P_s\cap O_s=\varnothing);$$

- 系统 I/O 集合 IO,$IO\subset O_s$,I/O 实体不能动态创建与删除,如网络接口、用户终端、外设等,由于只关心 I/O 间的信息流,IO 与信息域对应:$IO=D$;
- 系统客体名集合 N_o,$O_s\subseteq N_o$,为描述简单起见,用实体名表示实体;
- 系统实体值域集合 V',$V\subseteq V'$,$N_p\subseteq V'$,v_0,OK,ERROR$\in V'$,表示初值和成功/错误信息;
- 实体取值函数

$$val:S\times E_s\rightarrow V',\forall i\in IO:value(s,i)=val(s,i),$$

在这里和域的取值函数兼容,但不是必须的;

- 动作执行域函数

$$dom':A\rightarrow IO\cup P_s,\forall a\in A:dom'(a)\in IO\rightarrow dom'(a)=dom(a);$$

- 系统实体标记格$\langle L,\leq\rangle$,$\overline{\wedge}$ 为 L 上的最小上界运算;
- 实体标记函数 $l:S\times(P_s\cup N_o)\rightarrow L$,客体的标记包含在其客体名中,且客体的标记不能动态改变:

$$\forall s,t\in S,\forall o\in N_o:l(s,o)=l(t,o),$$

为简化表达,用符号 l_o 表示 $l(s,o)$.

系统 M'的动作集合 A 包括以下动作(设 $s,s'=step(s,a)$为执行动作前后的状态):

- I/O 输入 $in(x,v),x\in IO,v\in V',dom'(in(x,v))=x,\forall s\in S:enabled(s,in(x,v)):val(s',x)=val(s,x)\oplus v$.

其中,$\oplus$为 V'上的二元运算(对状态变化的简单描述).因此,$val(s',x)$只与 $val(s,x)$和 v 有关.来自外部的输入动作在任何状态下都可能发生.

- 读客体 $read(p,o),p\in P_s,o\in N_o,dom'(read(p,o))=p$:

if $o\in O_s$ then

$(val(s',p)=val(s,p)\oplus val(s,o)\oplus \mathrm{OK})\wedge(l(s',p)=l(s,p)\barwedge l_o)$

else

$(val(s',p)=val(s,p)\oplus \mathrm{ERROR})\wedge(l(s',p)=l(s,p)\barwedge l_o)$

当客体不存在时,系统返回错误信息并影响进程状态.当进程读取了高密级客体后,其密级应相应提升.

- 除输入动作 *in* 以外的所有动作 *a* 都有:$\forall s,t\in S: val(s,dom'(a))=val(t,dom'(a))\rightarrow enabled(s,a)=enabled(t,a)$,即动作 *a* 是否能执行只依赖于执行该动作的进程的运行状态;
- 写客体 $write(p,o),p\in P_s,o\in N_o,dom'(write(p,o))=p$:

if $(o\in IO\wedge l(s,p)\le l_o)\vee(o\in O_s\wedge l(s,p)=l_o)$ then

$(val(s',o)=val(s,o)\oplus val(s,p))\wedge(val(s',p)=val(s,p)\oplus \mathrm{OK})$

else

$val(s',p)=val(s,p)\oplus \mathrm{ERROR}$

为简化表达,写的内容并不明确表示,但客体的新值只与进程和原客体值相关.将 I/O 客体与一般客体区别开来,是因为内部客体集合动态可变,客体的存在状态是一种隐式信息.

- 创建客体 $create(p,o),p\in P_s,o\in N_o,dom'(create(p,o))=p$:

if $o\notin O_s\wedge l(s,p)=l_o$ then

$(O_{s'}=O_s\cup\{o\})\wedge(val(s',p)=val(s,p)\oplus \mathrm{OK})\wedge(val(s',o)=v_0)$

else

$val(s',p)=val(s,p)\oplus \mathrm{ERROR}$

- 删除客体 $delete(p,o),p\in P_s,o\in N_o,dom'(delete(p,o))=p$:

if $o\in O_s-IO\wedge l(s,p)=l_o$ then

$(O_{s'}=O_s-\{o\})\wedge(val(s',p)=val(s,p)\oplus \mathrm{OK})$

else

$val(s',p)=val(s,p)\oplus \mathrm{ERROR}$

- 进程通信 $msg(p,q),p\in P_s,q\in N_p,dom'(msg(p,q))=p$:

if $q\in P_s\wedge l(s,p)=l(s,q)$ then

$(val(s',p)=val(s,p)\oplus \mathrm{OK})\wedge(val(s',q)=val(s,q)\oplus val(s,p))$

else

$val(s',p)=val(s,p)\oplus \mathrm{ERROR}$

- 终止进程 $exit(p),p\in P_s,dom'(exit(p))=p: P_{s'}=P_s-\{p\}$
- 提升进程 $rise(p,l),p\in P_s,l\in L,dom'(rise(p,l))=p: l(s',p)=l(s,p)\barwedge l$
- 创建进程 $fork(p),p\in P_s,q\in N_p,dom'(fork(p))=p$:

$(q=getpid(val(s,p)))\wedge(q\in P_{s'}-P_s)\wedge(val(s',p)=val(s,p)\oplus q\oplus \mathrm{OK})\wedge(val(s',q)=val(s,p)\oplus p\oplus q)\wedge(l(s',q)=l(s,p))$

$getpid(val(s,p))$得到一个不重复的进程号:$q\in P_{s'}-P_s$.在此假设 $val(s,p)$中记录所有已创建子进程号,且不同进程的取值不相同.因此,$getpid(val(s,p))$得到一个不重复进程号是可行的.成功执行后,子进程得到进程号和父进程号,而父进程得到子进程号.对于执行一个可执行客体的操作,可以通过先创建一个进程,然后读取该客体来实现.

定义动作的不干扰关系和系统强制的信息流策略:

- $a\not\succ_s i \underline{\underline{\mathrm{def}}}\ l(s,dom'(a))\not\le l_i$
- $x\rightsquigarrow i \underline{\underline{\mathrm{def}}}\ l_x\le l_i$ □

定理 **2**. 不干扰关系$\not\succ_s$的定义对系统 *M*′是安全的.

证明:现分别证明 *M*′满足定理 1 的 4 个条件.

1. 对$\forall i\in IO$定义状态 *S* 上的等价关系:

$s\sim^i t \underline{\text{def}}\ \forall x:(x\in O_s\leftrightarrow x\in O_t)\wedge(x\in P_s\wedge l(s,x)\le l_i\leftrightarrow x\in P_t\wedge l(t,x)\le l_i)\wedge(x\in E_s\wedge l(s,x)\le l_i\rightarrow l(s,x)=l(t,x)\wedge val(s,x)=val(t,x))$.

显然,M'满足 $s\sim^i t\rightarrow value(s,i)=value(t,i)$.

2. 证明 M'满足 $s\sim^i t\wedge enabled(s,a)\wedge\neg enabled(t,a)\rightarrow a\not\vdash_s i$.

当 a 为输入动作时,$\forall s\in S:enabled(s,a)$,显然命题成立.否则,假设 $s\sim^i t\wedge enabled(s,a)\wedge\neg enabled(t,a)$,则 $val(s,dom'(a))\neq val(t,dom'(a))$,由上面$\sim^i$定义可推出 $l(s,dom'(a))\not\le l_i$,即 $a\not\vdash_s i$.

3. 证明 M'满足 $a\not\vdash_s i\wedge enabled(s,a)\rightarrow s\sim^i step(s,a)$.

假设 $a\not\vdash_s i\wedge enabled(s,a)$.

(1) 若 $a=in(x,v)$,a 只改变 $val(step(s,a),x)$,而 $a\not\vdash_s i$ 即 $l_x\not\le l_i$,因此 $s\sim^i step(s,a)$成立.

(2) 若 $a=write(p,o)$,则 $l(s,p)\not\le l_i$.如果$(o\in IO\wedge l(s,p)\le l_o)\vee(o\in O_s\wedge l(s,p)=l_o)$,则 $l_o\not\le l_i$,且动作只改变 $val(step(s,a),p)$和 $val(step(s,a),o)$,否则只改变 $val(step(s,a),p)$,因此,$s\sim^i step(s,a)$成立.

类似可证明对其他动作命题也成立,限于篇幅在此省略.

4. 证明 M'满足 $s\sim^i t\wedge enabled(s,a)\wedge enabled(t,a)\rightarrow step(s,a)\sim^i step(t,a)$.

现证明 $a=write(p,o)$时命题成立,其他动作可类似证明,在此省略.

假设 $s\sim^i t\wedge enabled(s,a)\wedge enabled(t,a)$,分情况讨论:

(1) 若 $l(s,p)\le l_i\wedge l_o\le l_i$,则 $l(t,p)\le l_i\wedge l_o\le l_i$,且 $l(s,p)=l(t,p)\wedge val(s,p)=val(t,p)\wedge val(s,o)=val(t,o)$,因此,$(o\in IO\wedge l(s,p)\le l_o)\vee(o\in O_s\wedge l(s,p)=l_o)\leftrightarrow(o\in IO\wedge l(t,p)\le l_o)\vee(o\in O_t\wedge l(t,p)=l_o)$,即在状态 s 和 t 下,要么都成功执行,要么都失败:$val(s',p)=val(t',p)\wedge val(s',o)=val(t',o)$,而其他状态不变.因此,$step(s,a)\sim^i step(t,a)$;

(2) 若 $l(s,p)\le l_i\wedge l_o\not\le l_i$,则 $l(t,p)\le l_i\wedge l_o\not\le l_i$,且 $l(s,p)\neq l_o\wedge l(s,p)=l(t,p)\wedge val(s,p)=val(t,p)$,因此,$(o\in IO\wedge l(s,p)\le l_o)\vee(o\in O_s\wedge l(s,p)=l_o)\leftrightarrow(o\in IO\wedge l(t,p)\le l_o)\vee(o\in O_t\wedge l(t,p)=l_o)$,即在状态 s 和 t 下,要么都成功执行,要么都失败:$val(s',p)=val(t',p)$.而 $l_o\not\le l_i$,因此,$step(s,a)\sim^i step(t,a)$;

(3) 若 $l(s,p)\not\le l_i\wedge l_o\le l_i$,则 $l(t,p)\not\le l_i\wedge l_o\le l_i$,且 $l(s,p)\not\le l_o\wedge l(t,p)\not\le l_o$,因此,$\neg((o\in IO\wedge l(s,p)\le l_o)\vee(o\in O_s\wedge l(s,p)=l_o))$,且$\neg((o\in IO\wedge l(t,p)\le l_o)\vee(o\in O_t\wedge l(t,p)=l_o))$,即在状态 s 和 t 下都失败:除 $val(s',p)$和 $val(t',p)$外,其他状态不变.而 $l(s,p)\not\le l_i\wedge l(t,p)\not\le l_i$,因此,$step(s,a)\sim^i step(t,a)$;

(4) 若 $l(s,p)\not\le l_i\wedge l_o\not\le l_i$,则 $l(t,p)\not\le l_i\wedge l_o\not\le l_i$,由于动作不修改实体标记,显然,$step(s,a)\sim^i step(t,a)$. □

定理 **3**. 信息流策略$\sim$对系统 M'是安全的.

证明:即证明$\forall x\in IO,i\in IO,a\in A,s\in S:\neg(x\sim i)\wedge dom(a)=x\rightarrow a\not\vdash_s i$.

假设$\neg(x\sim i)\wedge dom(a)=x$,则 a 只可能是输入动作 $in(x,v)$,则 $l_x\not\le l_i$,即 $l(s,dom'(a))\not\le l_i$.根据定义,$a\not\vdash_s i$,因此定理成立. □

4 结束语

系统动作不干扰概念的引入,不仅简化了动作序列的提取操作,而且从概念上指出了动作序列提取的实质.系统动作的不干扰概念直观、易懂,为系统验证提供了方便,并且可以用来验证动态信息流策略,而这是其他类似模型[2,3,8,10]所不能的.文中给出的单步状态展开条件与文献[3]一样基于状态的等价关系,具有简洁的形式,但我们将其推广到了非确定系统,而文献[8]单步状态展开条件基于状态的序关系.本文最后设计了一个基于动态标记的访问控制模型,并用不干扰模型对其安全性进行了验证.该模型虽然简单,但包含了操作系统的很多关键操作,并定义了这些操作的具体语义,有效地排除了存储隐蔽信道.基于主体动态标记的访问控制模型还有文献[11],但该模型并未讨论存储隐蔽信道问题.文献[4]提出了一个基于语言的机制来检查动态安全标记.我们的访问控制模型表明:读、写等操作背后隐藏的信息流并不像字面上那样明确,为排除隐蔽信道需要增加很多限制.不干扰模型不仅能对其安全性进行验证,还可以指导设计者对信息流的判断.即使因功能需要不能关闭所有隐蔽信道,但至少我们可以知道它们的存在,并尽量减小它们的危害.

References:

[1] Peri RV. Specification and verification of security policies [Ph.D. Thesis]. Virginia: University of Virginia, 1996.

[2] Goguen J, Meseguer J. Security policies and security models. In: Proc. of the 1982 IEEE Symp. on Research in Security and Privacy. Los Alamitos: IEEE Computer Society Press, 1982. 11−20. http://doi.ieeecomputersociety.org/10.1109/SP.1982.10014

[3] Rushby J. Noninterference, transitivity, and channel-control security policies. Technical Report, CSL-92-02, Menlo Park: Stanford Research Institute, 1992.

[4] Zheng L, Myers AC. Dynamic security labels and noninterference. In: Dimitrakos T, Martinelli F, eds. Proc. of the 2nd Int'l Workshop on Formal Aspects in Security and Trust (FAST). Toulouse: Springer-Verlag, 2004. 27−40.

[5] Sabelfeld A, Myers AC. Language-Based information-flow security. IEEE Journal on Selected Areas in Communications, 2003, 21(1):1−15.

[6] Hicks M, Tse S, Hicks B, Zdancewic S. Dynamic updating of information-flow policies. In: Sabelfeld A, ed. Proc. of the Int'l Workshop on Foundations of Computer Security (FCS). Chicago, 2005. 7−18. http://www.cs.chalmers.se/~andrei/FCS05/fcs05.pdf

[7] Ryan P, McLean J, Millen J, Gligor V. Non-Interference, who needs it? In: Proc. of the 14th IEEE Computer Security Foundations Workshop. Los Alamitos: IEEE Computer Society Press, 2001. 237−241. http://csdl.computer.org/comp/proceedings/csfw/2001/1146/00/11460237.pdf

[8] Mantel H. Unwinding possibilistic security properties. In: Cuppens F, Deswarte Y, Gollmann D, Waidner M, eds. Proc. of the 6th European Symp. on Research in Computer Security (ESORICS 2000). Toulouse: Springer-Verlag, 2000. 238−254.

[9] McLean J. Security models. In: Marciniak JJ, ed. Encyclopedia of Software Engineering. New York: John Wiley & Sons, 1994. http://www.cs.pomona.edu/classes/cs190/Spapers/4-mclean94security.pdf

[10] Zakinthinos A, Lee ES. A general theory of security properties. In: Proc. of the 1997 IEEE Symp. on Security and Privacy. Los Alamitos: IEEE Computer Society Press, 1997. 94−102. http://doi.ieeecomputersociety.org/10.1109/SECPRI.1997.601322

[11] Liang HL, Sun YF, Zhao QS, Zhang XF, Sun B. Design and implementation of a security label common framework. Journal of Software, 2003,14(3):547−552 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/547.htm

附中文参考文献:

[11] 梁洪亮,孙玉芳,赵庆松,张相锋,孙波.一个安全标记公共框架的设计与实现.软件学报,2003,14(3):547−552. http://www.jos.org.cn/1000-9825/14/547.htm

谢钧(1973－),男,四川成都人,博士,讲师,主要研究领域为计算机网络,信息安全.

黄皓(1957－),男,博士,教授,博士生导师,主要研究领域为计算机网络,信息安全.