多线程环境中的二维降密策略

金 丽[1] 朱 浩[2,3]
(南通大学江苏省专用集成电路设计重点实验室 南通 226019)[1]
(南京航空航天大学计算机科学与技术学院 南京 210016)[2]
(南通大学计算机科学与技术学院 南通 226019)[3]

摘 要 降密策略的主要目的在于确保程序中敏感信息的安全释放。目前,降密策略的安全条件和实施机制的研究主要集中在顺序式程序设计语言,它们不能直接移植到多线程并发环境,原因在于攻击者能利用线程调度的某些性质推导出敏感信息。为此,基于多线程程序设计语言模型和线程调度模型,建立了支持多线程并发环境的二维降密策略,有效确保了在合适的程序点降密合适的信息;建立了多线程并发环境下该降密策略的动态监控机制,并证明了该实施机制的可靠性。

关键词 信息流,多线程环境,机密性,无干扰
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.052

Two-dimension Declassification Policy in Multithreaded Environments

JIN Li[1] ZHU Hao[2,3]
(Jiangsu Key Laboratory of Asic Design, Nantong University, Nantong 226019, China)[1]
(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(School of Computer Science and Technology, Nantong University, Nantong 226019, China)[3]

Abstract Information declassification aims at secure release of sensitive information. Existing security specifications and enforcement mechanisms of declassification policies focus on sequential programs, and they can not be directly transplanted to multithreaded environments for that attackers can take advantage of some properties of thread scheduling to derive sensitive information. To this end, a two-dimension declassification policy in multithreaded environments was proposed, based on the multi-threaded programming language model and thread scheduling model, effectively ensuring that appropriate information is released at the appropriate point of programs. Moreover, dynamic monitoring mechanisms of the policy in multithreaded environments were presented, and the soundness of enforcements was proved.

Keywords Information flow, Multithreaded environments, Confidentiality, Non-interference

目前,信息降密策略的研究集中在顺序式程序语言环境中,不能直接应用到并发式多线程环境[1]。多线程是现代程序语言的重要特征,它能提供一个遵循"职责分离"(separation-of-duties)的实施机制,对安全性要求高的系统非常重要[2]。因此,研究并发多线程环境中的降密策略及其实施机制是非常有必要的。

并发多线程引发了一些新的攻击行为,比如内部时间隐通道[3,12]。在多线程环境中,概率无干扰判定在程序输出中,低安全级别变量的可能取值不依赖于高安全级别的输入变量[4]。但是,概率无干扰依赖纯粹的不确定性的(purely nondeterministic)线程调度器,当线程的调度采用服从一定概率分布的调度器时,将会引入时间隐通道,概率无干扰策略不能检测出这种隐蔽通道,不能提供相应的安全保证。比如,考虑包含以下两个线程 P_1 和 P_2 的程序:

P_1:if $x=1$ then (C;C;)
  $y:=1$;
P_2:C;$y:=0$;

其中变量 x 是高机密性变量,它的取值只有 0 和 1 两种,变量 y 是低机密性变量,命令 C 中不包含给 y 赋值的语句,且 C 需要很多转移步才能完成。假设线程调度是纯粹不确定的,那么该程序满足概率无干扰,原因是变量 y 的最终值不依赖于变量 x 的初始值。但是,假设线程 P_1 和线程 P_2 的调度服从均匀分布,那么变量 x 的值拷贝给变量 y 的概率非常大,即变量 x 的初值的变化将影响变量 y 的最终值的概率分布。根据

到稿日期:2014-12-10 返修日期:2015-02-26 本文受江苏省博士后科研资助计划(1401022C),江苏省高校自然科学研究面上项目(14KJB510028),南通市科技计划项目(BK2014061)资助。

金 丽(1978—),女,讲师,主要研究方向为软件安全,E-mail:jin.li@ntu.edu.cn;朱 浩(1977—),男,博士,副教授,CCF 会员,主要研究方向为软件安全,E-mail:searain@nuaa.edu.cn(通信作者)。

降密策略的稳健性原则，不包含降密语句的程序应该遵循无干扰策略[1]。上述例子显然不满足无干扰策略，因此多线程环境中的降密策略需要拒绝这些攻击，但又不能破坏安全策略的宽容性。

目前，并发多线程环境中的降密策略成果较少，理论基础较为薄弱。国外 Heiko Mantel 等人[5,6]基于强互模拟关系研究了多线程语言下的降密策略，并给出了一个可靠的类型系统，但是它们的策略在地点维度的限制性过强。国内姜励等人[7]基于强互模拟等价关系定义了多线程环境下降密的属性，但仅考虑了降密的内容维度。虽然并发多线程环境下的释放策略成果较少，但是并发多线程环境下的无干扰策略却有相对较多的成果。Focardi 等人[8]提出 NDC(Non Deduci-bility on Composition)性质：高安全级别的进程不能改变低安全级别的行为，并基于该性质利用相关工具对一系列加密协议进行了安全分析，发现了若干安全漏洞。Russo 和 Sabelfeld[9]提出了多线程环境下的无干扰模型。本文在前人工作的基础上，研究多线程环境下的降密。在我们前期的工作中，已经提出集成内容和地点维度的降密策略[10]，策略的语义条件规定了在程序中合适的地点释放合适内容的敏感信息。在本文中，将该二维降密策略扩展到多线程环境中，并提出了多线程环境下该策略的动态监控机制，同时证明了该机制的可靠性。本文第 1 节描述了多线程的程序语言 MWhile；第 2 节给出了处理器的调度模型；第 3 节描述了多线程环境中二维降密策略的语义条件；第 4 节给出了多线程环境下策略的监控机制，并进行了可靠性证明；最后总结全文。

1 多线程的语言模型

本节描述了一种支持动态创建多线程的程序语言 MWhile，它是在我们前期工作所采用的 While 语言的基础上[10]，增加了动态创建多线程、提升和降低线程安全级别的原语[9]。假设在多线程环境中，只有一个 CPU，线程之间以共享内存的方式通信。MWhile 语言的语法如图 1 所示。

e::= n | x | e⊕e

A::=skip|x:=e|x:=declassify(e) | hide | unhide

fork(c,$\vec{d}$)|hfork(c,$\vec{d}$)

B::=if e then C else C | while e do C

C::=A | B | C;C

图 1 MWhile 语言的语法

其中每个常量和变量都具有一个机密性级别，分为低机密性 L 和高机密性 H 两级。对外部观察者而言，低机密性 L 的数据是完全公开的，而高机密性 H 的数据是不可见的。令常量的机密性级别为 L，变量的机密性级别可通过映射 Γ 获取，Γ 是一个从变量到机密性级别的映射(也称为机密性环境)。

MWhile 支持低机密性线程和高机密性线程，线程池分为低机密性线程池和高机密性线程池。调度器对不同级别的线程作不同处理，使得并发线程中出现的可观察事件不依赖于敏感信息。命令的操作语义除了增加的原语外，其余的都与 While 语言[10]相同。命令 $x:=\text{declassify}(e)$ 是高机密信息的降密语句，将表达式 e 的机密性级别从 H 降低到 L，并赋值给变量 x，其中 e 称作降密表达式。在命令的操作语义中，命令配置之间的转移关系可描述为$\langle m,c\rangle \xrightarrow{\alpha}_{\ell} \langle m',c'\rangle$，表示命令 c 在存储状态 m 下单步转移后到达$\langle m',c'\rangle$，其中 α 是 c 触发的转移事件，ℓ 是可观察事件，它由对低机密性变量的赋值和降密语句触发。$\xrightarrow{\vec{\alpha}}_{\vec{\ell}}$ 或 $\xrightarrow{*}_{\vec{\ell}}$ 表示多步转移，其中 $\vec{\alpha}$ 和 $\vec{\ell}$ 分别表示转移事件序列和可观察事件序列；当不关心转移后的情况时，略写为$\langle m,c\rangle \xrightarrow{*}_{\vec{\ell}}$。增加的原语的操作语义如图 2 所示。转移事件 α 扩充了 4 个事件$\{\rightsquigarrow L, \rightsquigarrow H, L_{\vec{d}}, H_{\vec{d}}\}$，其中 $\vec{d}$ 表示线程的集合。事件 $\rightsquigarrow H$ 由 hide 命令触发，被线程调度器捕获，然后调度器将该线程从低机密性线程池转移到高机密性线程池；与此相反，$\rightsquigarrow L$ 由 unhide 命令触发，调度器捕获该事件，将当前线程从高机密性池转移到低机密性池。命令 hide 与 unhide 都不能嵌套使用。fork(c,$\vec{d}$)和 hfork(c,$\vec{d}$)表示当前线程 c 动态地创建了线程集合 $\vec{d}$，这两个命令的区别在于 fork(c,$\vec{d}$)表示创建的线程都是低机密性的，对应的事件为 $L_{\vec{d}}$，而 hfork(c,$\vec{d}$)表示创建的线程都是高机密性的，对应的事件为 $H_{\vec{d}}$。

(S-HIDE): $\langle m,\text{hid}\rangle \xrightarrow{\rightsquigarrow H} \langle m,\text{stop}\rangle$

(S-UNHIDE): $\langle m,\text{unhide}\rangle \xrightarrow{\rightsquigarrow L} \langle m,\text{stop}\rangle$

(S-FORK): $\langle m,\text{fork}(c,\vec{d})\rangle \xrightarrow{L_{\vec{d}}} \langle m,c\rangle$

(S-HFORK): $\langle m,\text{hfork}(c,\vec{d})\rangle \xrightarrow{H_{\vec{d}}} \langle m,c\rangle$

图 2 MWhile 语言的操作语义

2 调度模型

假定调度器是安全的，不存在任何机密信息的泄漏，即满足无干扰策略[1]。调度器的配置表示为$\langle \nu,\sigma\rangle$，其中 ν 为调度器的存储，与各个线程的存储不相交，σ 为调度器程序。在调度器的存储中，变量 q 表示一个线程在一次调度中允许发生转移的次数。待调度的线程集合用变量 t 表示，其中低机密性线程集合和高机密性线程集合分别用 t_L 和 t_H 表示；通过命令 hide 和 unhide，线程可在 t_H 和 t_L 之间来回切换。变量 r 表示当前运行的线程。变量 s 用来限制低机密性线程的调度，防止低机密性线程观察到高机密性线程的内部时间信息。当变量 s 的值为 L 时，低机密性和高机密性线程都能被调度；当 s 的值为 H 时，只有高机密性级别的线程能被调度。调度器配置之间的转移关系可表示为：$\langle \nu,\sigma\rangle \xrightarrow{\alpha} \langle \nu',\sigma'\rangle$，表示调度程序 σ 在存储状态 ν 下单步执行后到达配置$\langle \nu',\sigma'\rangle$，其中 α 为调度事件，表示如下：α::=$H^r_{\vec{d}}$ | $L^r_{\vec{d}}$ | $r \rightsquigarrow \times$ | $\uparrow r$ | $r \rightsquigarrow H$ | $r \rightsquigarrow L$ | $r \rightsquigarrow$。$H^r_{\vec{d}}$ 表示线程 r 要求动态创建高机密性线程集合 $\vec{d}$，$L^r_{\vec{d}}$ 表示线程 r 要求动态创建低机密性线程集合 $\vec{d}$，$r \rightsquigarrow \times$ 表示线程 r 执行终止，$\uparrow r$ 表示调度器将选择高机密性的线程 r 运行，$r \rightsquigarrow H$ 表示线程 r 的机密性级别提升为高机密性，$r \rightsquigarrow L$ 表示线程 r 的机密性级别降为低机密性，$r \rightsquigarrow$ 表示线程 r 的除上述事件以外的一次非终止转移。此外，存储状态 ν 下的

相关变量用 q,t,r,s 表示，存储状态 ν' 下的相关变量用 q',t',r',s' 表示。

调度器的操作语义如图 3 所示。规则(S-CREA)表示调度器将根据新创建的线程的机密性级别的高低更新线程池的相应部分，调度器配置转移后变量 q 的值减 1。规则(S-TERM)表示当线程 r 触发事件 $r \rightsquigarrow \times$ 时，调度器将根据机密性级别高低从相应的线程池中删除线程 r，并在转移后将变量 q 的值置为 0。规则(S-SEL-L)表示当变量 s 的值是 L 时，调度时可以选择高机密性线程，也可选择低机密性线程。规则(S-SEL-H)表示当变量 s 的值是 H 时，调度时只能选择高机密性线程。规则(S-UP)表示调度器根据事件 $r \rightsquigarrow H$ 将当前线程 r 从低机密性线程池转移到高机密性线程池，并在转移后将变量 s 的值设置为 H，变量 q 的值减 1。规则(S-DOWN)表示调度器根据命令 unhide 触发的事件 $r \rightsquigarrow L$ 将当前线程 r 从高机密性线程池转移到低机密性线程池，并在转移后将变量 s 的值置为 L，变量 q 的值置 0。规则(S-TRAN)表示调度器将根据事件 $r \rightsquigarrow$ 同步地发生一次调度器状态转移，调度器配置转移后变量 q 的值减 1。

$$\text{(S-CREA)}:\frac{q'=q-1 \quad t_x'=t_x\cup\vec{d} \quad x\in\{H,L\}}{\langle\nu,\sigma\rangle\xrightarrow{x\$}\langle\nu',\sigma'\rangle}$$

$$\text{(S-TERM)}:\frac{q'=0 \quad t_x'=t_x\backslash\{r\} \quad x\in\{H,L\}}{\langle\nu,\sigma\rangle\xrightarrow{r\rightsquigarrow\times}\langle\nu',\sigma'\rangle}$$

$$\text{(S-SEL-L)}:\frac{q=0 \quad s=L \quad q'>0 \quad r'\in t_L\cup t_H}{\langle\nu,\sigma\rangle\xrightarrow{\uparrow r'}\langle\nu',\sigma'\rangle}$$

$$\text{(S-SEL-H)}:\frac{q=0 \quad s=H \quad q'>0 \quad r'\in t_H}{\langle\nu,\sigma\rangle\xrightarrow{\uparrow r'}\langle\nu',\sigma'\rangle}$$

$$\text{(S-UP)}:\frac{q'=q-1 \quad s'=H \quad t_L'=t_L\backslash\{r\} \quad t_H'=t_H\backslash\{r\}}{\langle\nu,\sigma\rangle\xrightarrow{r\rightsquigarrow H}\langle\nu',\sigma'\rangle}$$

$$\text{(S-DOWN)}:\frac{q'=0 \quad s'=L \quad t_L'=t_L\cup\{r\} \quad t_H'=t_H\backslash\{r\}}{\langle\nu,\sigma\rangle\xrightarrow{r\rightsquigarrow L}\langle\nu',\sigma'\rangle}$$

$$\text{(S-TRAN)}:\frac{q'=q-1}{\langle\nu,\sigma\rangle\xrightarrow{r\rightsquigarrow}\langle\nu',\sigma'\rangle}$$

图 3　调度器的操作语义

3　多线程环境下的降密策略

在前期的工作中[10]，基于攻击者知识[11]提出了结合内容和地点维度的降密策略，本节简要回顾该策略的语义条件，并将其扩展到多线程并发环境。

令 m_L 表示存储状态 m 中低机密性变量的映射，关系 $m_{1L}=m_{2L}\Leftrightarrow\forall x\cdot\Gamma(x)=L\Rightarrow m_1(x)=m_2(x)$。

定义 1(多线程环境中的攻击者知识)　假设调度器 σ 满足无干扰策略，ν 为调度器存储。线程池 $\vec{c}$ 从初始存储 m 开始运行，产生可观察事件序列 $\vec{\ell}$，则攻击者知识 $k(m_L,\vec{c},\nu,\sigma,\vec{\ell})$ 为：

$$k(m_L,\vec{c},\nu,\sigma,\vec{\ell})=\{m'\mid m_L'=m_L\wedge(\langle m',\vec{c},\nu,\sigma\rangle\xrightarrow{*}_{\vec{\ell}}\langle m'',stop,\nu',\sigma'\rangle\vee\langle m',\vec{c},\nu,\sigma\rangle\xrightarrow{*}_{\vec{\ell}.\ell})\}$$

其中，$\langle m,\vec{c},\nu,\sigma\rangle$ 是线程池的配置，$\xrightarrow{*}_{\vec{\ell}}$ 是配置之间的多步转移，攻击者初始知识 $k(m_L,\vec{c},\nu,\sigma,\phi)$ 为：

$$k(m_L,\vec{c},\nu,\sigma,\phi)=\{m'\mid m_L'=m_L\wedge\exists m'',\vec{\ell}\cdot\langle m',\vec{c},\nu,\sigma\rangle\xrightarrow{*}_{\vec{\ell}}\langle m'',stop,\nu',\sigma'\rangle\}$$

定义 2(多线程环境下“WHAT&WHERE”策略)　假设调度器 σ 满足无干扰策略。线程池 $\vec{c}$ 从初始存储状态 m_0 开始运行，产生长度为 n 的可观察事件序列 $\vec{\ell}_n=\ell_1\cdots\ell_n$，$n\geqslant 1$，其中 $\ell_{r_1}\cdots\ell_{r_k}$ 是 $\vec{\ell}_n$ 中由降密语句触发的可观察事件序列，$1\leqslant r_i\leqslant r_{i+1}\leqslant n$ 且 $1\leqslant i\leqslant k$，该序列中每个 ℓ_{r_i} 对应的降密表达式为 e_{r_i}，其中 $1\leqslant i\leqslant k$，令 m^{r_i} 表示 ℓ_{r_i} 发生时的当前存储状态，程序 c 的运行满足“WHAT&WHERE”策略当且仅当同时满足下列两个安全条件：

I. $\forall i(1\leqslant i\leqslant n),(\forall j(1\leqslant j\leqslant k),i\neq r_j)\Rightarrow k(m_{0L},\vec{c},\nu,\sigma,\vec{\ell}_{i-1})=k(m_{0L},\vec{c},\nu,\sigma,\vec{\ell}_i)$，其中，$k(m_{0L},\vec{c},\nu,\sigma,\vec{\ell}_0)=k(m_{0L},\vec{c},\nu,\sigma,\phi)$；

II. $\forall j(1\leqslant j\leqslant k),m_0(e_{r_j})=m^{r_j}(e_{r_j})$

4　实施机制

为了动态实施多线程环境下的“WHAT&WHERE”策略，本节给出了该策略的动态监控机制，并进行了实例分析和可靠性分析。

4.1　监控机制

监控配置用四元组 $\langle m_0,\Gamma,st,hc\rangle$ 表示，其中 m_0 表示程序运行的初始存储状态(不包括调度器的存储)，st 是一个栈，Γ 是机密性环境，变量 hc 用来跟踪当前线程的机密性级别，取值为 H 和 L，分别表示高机密性和低机密性级别。监控配置之间的转移关系为 $\langle m_0,\Gamma,st,hc\rangle\xrightarrow{\alpha}\langle m_0,\Gamma',st',hc'\rangle$，其中 α 为转移事件。图 4 描述了实施多线程环境下“WHAT & WHERE”策略的监控语义规则。

$$\text{(T-SKIP)}:\langle m_0,\Gamma,st,hc\rangle\xrightarrow{nop}\langle m_0,\Gamma,st,hc\rangle$$

$$\text{(T-DECLASSIFY)}:\frac{m_0=m(e) \quad lev(st)=L \quad hc=L}{\langle m_0,\Gamma,st,hc\rangle\xrightarrow{d(x,e)}\langle m_0,\Gamma,st,hc\rangle}$$

$$\text{(T-ASSIGN)}:\frac{\Gamma(e)\sqsubseteq\Gamma(x) \quad lev(st)\sqsubseteq\Gamma(x) \quad hc\sqsubseteq\Gamma(x) \quad lev(st)\sqsubseteq hc}{\langle m_0,\Gamma,st,hc\rangle\xrightarrow{a(x,e)}\langle m_0,\Gamma,st,hc\rangle}$$

$$\text{(T-PUSH)}:\langle m_0,\Gamma,st,hc\rangle\xrightarrow{b(e)}\langle m_0,\Gamma,\Gamma(e)\sqcup lev(st)::st,hc\rangle$$

$$\text{(T-POP)}:\langle m_0,\Gamma,l::st,hc\rangle\xrightarrow{f}\langle m_0,\Gamma,st,hc\rangle$$

$$\text{(T-HIDE)}:\langle m_0,\Gamma,st,hc\rangle\xrightarrow{\rightsquigarrow H}\langle m_0,\Gamma,st,hc:=H\rangle$$

$$\text{(T-UNHIDE)}:\langle m_0,\Gamma,st,hc\rangle\xrightarrow{\rightsquigarrow L}\langle m_0,\Gamma,st,hc:=L\rangle$$

$$\text{(T-FORK)}:\frac{hc=L}{\langle m_0,\Gamma,st,hc\rangle\xrightarrow{L_{\vec{q}}}\langle m_0,\Gamma,st,hc\rangle}$$

$$\text{(T-HFORK)}:\frac{hc=H}{\langle m_0,\Gamma,st,hc\rangle\xrightarrow{H_{\vec{q}}}\langle m_0,\Gamma,st,hc\rangle}$$

图 4　多线程环境下“WHAT&WHERE”策略的监控语义规则

规则(T-SKIP)表示命令 skip 的运行始终是安全的，它的执行不会引起监控配置的任何改变，其中 nop 代表 skip 命令触发的转移事件。规则(T-DECLASSIFY)表示 $x:=$declassify(e)的安全执行须满足 3 个条件：1)降密表达式的值在初始存储和当前存储下必须相等；2)st 栈顶元素的机密性级别为 L，其中 $lev(st)$ 表示获取 st 栈顶元素；3)变量 hc 的值为 L，

即当前线程为低机密性线程。$d(x,e)$是由降密语句 $x:=$declassify(e)触发的转移事件。规则(T-ASSIGN)表示赋值语句$x:=e$的安全执行必须满足以下条件:表达式 e 的机密性级别、st 的栈顶元素以及变量 hc 的机密性级别必须分别小于等于变量 x 的机密性级别,而且 st 的栈顶元素的机密性级别必须小于等于变量 hc 的机密性级别。赋值语句的执行不会引起监控配置的任何改变。$a(x,e)$是由赋值语句 $x:=e$ 触发的转移事件。规则(T-PUSH)表示分支命令的执行将引起栈 st 中元素的变化,需要将当前分支表达式 e 的机密性级别与栈顶元素的最小上界压入栈 st 中,其中符号"::"表示入栈操作。$b(e)$是由分支命令触发的转移事件,e 是分支表达式。循环语句可以看成分支语句的若干次执行,是一种特殊的分支语句。规则(T-POP)表示当退出当前层次分支命令的作用范围时,st 的栈顶元素需要出栈,f 是由退出当前层的分支语句触发的转移事件。规则(T-HIDE)表示 hide 命令的执行将引起变量 hc 的值提升为 H;相反,规则(T-UNHIDE)表示 unhide 命令的执行将引起变量 hc 的值降低为 L。规则(T-FORK)表示 fork 命令的安全执行必须满足 $hc=L$,即只有低机密性级别的线程才能执行 fork 命令。规则(T-HFORK)表示 hfork 命令的安全执行必须满足 $hc=H$,即只有高机密性级别的线程才能执行 hfork 命令。

4.2 实例分析

考虑如下的多线程程序:

P_3:if $h>0$ then sleep(100) else skip;
  $l:=1$;

P_4:sleep(50);$l:=0$;

其中线程 P_3 和线程 P_4 都是低机密性线程,h 是高机密性变量,l 是低机密性变量。图 4 的监控语义能检测出该多线程的执行是不安全的,原因在于当线程 P_3 的 if 语句未执行完而让出 CPU 资源给线程 P_4 执行时,规则(T-ASSIGN)的前提条件 $lev(st)\sqsubseteq hc$ 不满足,所以该程序的本次执行是不符合多线程环境下的"WHAT&WHERE"降密策略的,但是如果做如下修改:

P_3':hide;if $h>0$ then sleep(100) else skip;unhide;
  $l:=1$;

P_4':sleep(50);$l:=0$;

由于在线程 P_3'起始部分增加了 hide 命令,这将线程 P_3'的当前机密性级别提升为 H;根据图 3 的调度器语义规则(S-UP),变量 s 的值被置为 H;再根据图 3 调度器语义规则(S-SEL-H)、(S-SEL-L)和变量 s 的值,调度器在后续的调度中不会调度线程 P_4',只会调度线程 P_3'中的 if 语句,直到 if 语句执行完毕后再执行 unhide 语句;根据调度器语义规则(S-UP),变量 s 的值被置为 L;根据调度器语义规则(S-SEL-L),此时线程 P_4'才会被调度;而当线程 P_4'被调度执行时,线程 P_3'中的 if 语句已经执行完毕,根据监控语义规则(T-POP),此时 st 中的栈顶元素已经出栈,$lev(st)$的值已经恢复为 L,hc 的值也为 L,此时规则(T-ASSIN)中的前提条件 $lev(st)\sqsubseteq hc$ 是满足的。综上,修改后多线程程序 P_3'和 P_4'执行满足图 4 的监控语义规则,符合多线程环境下的"WHAT&WHERE"释放策略。

考虑另外一个多线程程序:

P_5:$x_1:=x_2$;$x_3:=x_2$;

P_6:$l:=6$;skip;$t:=$declassify$(x_1+x_2+x_3)$;$l:=8$;

假设变量 x_1-x_3 都是高机密性变量,t 和 l 是低机密性变量,线程 P_5 和 P_6 是低机密性线程。假设调度器的调度后线程中的命令执行序列为:

$l:=6$;skip;$x_1:=x_2$;$x_3:=x_2$;$t:=$declassify$(x_1+x_2+x_3)$;$l:=8$;

设初始存储 m_0 下变量 x_1-x_3 的值分别是整数 4,5,8,降密表达式在初始存储 m_0 的值:$m_0(x_1+x_2+x_3)=17$,而当降密语句执行时,降密表达式的当前值是 $m(x_1+x_2+x_3)=15$,这显然不满足监控语义规则(T-DECLASSIFY)的前提条件 $m_0(e)=m(e)$,由此可判定该多线程程序的本次执行不符合多线程环境下的"WHAT&WHERE"降密策略。

4.3 可靠性分析

定理 1(可靠性定理) 在图 4 的监控语义规则下,线程池$\vec{c}$监控执行满足定义 2 给出的多线程环境下的"WHAT&WHERE"降密策略。

证明:对可观察事件序列$\vec{\ell}_n$进行结构化归纳证明。

1)基础:当 $n=0$ 时,定义 2 的安全条件 I 和 II 恒成立。

2)归纳:假设结论对序列长度为 $n-1$ 的可观察事件序列$\vec{\ell}_{n-1}$成立,这里 $n\geqslant 1$,需要证明结论对序列长度为 n 的可观察事件序列$\vec{\ell}_n$成立。由于可观察事件ℓ_n包括由降密语句触发的可观察事件和普通赋值语句触发的可观察事件两种情况,因此可对ℓ_n分两种情况进行讨论:

①当ℓ_n是由普通赋值语句触发的可观察事件时,设对应的转移事件是 $a(x,e)$,根据监控规则(T-ASSIGN)中的条件 $\Gamma(e)\sqsubseteq\Gamma(x)$和 $lev(st)\sqsubseteq\Gamma(x)$可知,赋值语句的执行不会引起非法的直接流和非法的间接流;又根据规则(T-PUSH)和(T-POP),排除分支语句嵌套环境下的非法的间接流;另外,规则(T-ASSIGN)中的前提条件 $hc\sqsubseteq\Gamma(x)$和 $lev(st)\sqsubseteq hc$ 防止了在高机密性线程中对低机密性变量进行赋值以及高机密性的隐式流流入低机密性线程中,从而避免了时间隐通道导致的高机密性信息的泄漏。因此,攻击者没有获得任何高机密性信息,攻击者知识没有发生改变,所以安全条件 I 成立,又因为ℓ_n不是降密语句触发的可观察事件,所以安全条件 II 成立。

②当ℓ_n是降密语句触发的可观察事件时,安全条件 I 蕴涵式的左边的条件不成立,从而不要求攻击者知识不发生变化,即此时安全条件 I 恒成立。根据监控规则(T-DECLASSIFY),可直接得到安全条件 II 成立。

综上,定理 1 的结论成立。证毕。

结束语 降密策略是放松的无干扰策略,用来保证敏感信息的安全释放。本文将前期工作中提出的基于内容和地点维度的降密策略拓展到多线程环境,有效防止了在线程调度中由于时间隐通道造成的信息泄露,有效确保了在合适的程序点降密合适的信息。在后续的工作中,将在多线程环境中研究降密策略的更多维度,进一步提高程序的安全性。

(下转第 282 页)

总体上，对于给定的时间序列，DPSSD 算法可以获得理论上的全局最优解，而 DBA 算法则常常落入局部最优解。

结束语 本文提出了基于最小化中心序列到样本序列的 DTW 距离平方和来求解两条序列的中心序列的动态规划算法 DPSSD。本文通过引入匹配度理论，在理论上证明了该算法能够得到两条时间序列的 DTW 中心的最优解，同时还基于该理论，提出了相应的剪枝策略，从而降低了算法的时间复杂度。实验结果显示，应用该算法求得的中心序列与两条样本序列的 DTW 距离和小于 DBA 的解，序列长度也小于 DBA 的解，同时保留了时间序列的形状特征，算法在鲁棒性上也优于 DBA。

在下一步的研究中，我们将进一步讨论如何求解多条时间序列的 DTW 中心，并将我们的研究用于生物、金融和图像分析等领域。提出的算法也可以被用来对大脑认知时间序列进行动态时变分析。

参 考 文 献

[1] Anami B S, Pagi V B. Acoustic signal-based approach for fault detection in motorcycles using chaincode of the pseudospectrum and dynamic time warping classifier[J]. IET Intelligent Transport Systems, 2014, 8(1): 21-27

[2] Trajcevski G, Gunopulos D, Aggarwal C C. Time-series data clustering[M]. Aggarwal C C, Reddy C K, eds. Data Clustering: Algorithms and Applications, CRC Press, 2013: 357-375

[3] Hu B, Rakthanmanon T, Hao Y, et al. Using the minimum description length to discover the intrinsic cardinality and dimensionality of time series[J]. Data Mining and Knowledge Discovery, 2015, 29(2): 358-399

[4] Hautamaki V, Nykanen P, Franti P. Time-series clustering by approximate prototypes[C]// Proceedings of International Conference on Pattern Recognition. IEEE, 2008: 1-4

[5] Berndt D J, Clifford J. Using dynamic time warping to find patterns in time series[J]. KDD Workshop, ser. Seattle, WA, 1994, 10(16): 359-370

[6] 谢福鼎，李迎，孙岩，等. 一种基于关键点的时间序列聚类算法[J]. 计算机科学，2012，39(3)：157-159
Xie F D, Li Y, Sun Y, et al. Cluster Algorithm for Time Series Based on Key Points[J]. Computer Science, 2012, 39(3): 157-159

[7] 郭崇慧，苏木亚. 基于独立成分分析的时间序列谱聚类方法[J]. 系统工程理论与实践，2011，31(10)：1921-1931
Guo Chong-hui, Su Mu-ya. Spectral clustering method based on independent component analysis for time series[J]. Systems Engineering Theory & Practice, 2011, 31(10): 1921-1931

[8] Gupta L, Molfese D, Tammana R, et al. Nonlinear alignment and averaging for estimating the evoked potential[J]. IEEE Transactions on Biomedical Engineering, 1996, 43(4): 348-356

[9] Salvador S, Chan P. Toward accurate dynamic time warping in linear time and space[J]. Intelligent Data Analysis, 2007, 11(5): 561-580

[10] Niennattrakul V, Ratanamahatana C. Shape averaging under time warping[C]// Proceedings of International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. IEEE, 2009: 626-629

[11] Petitjean F, Ketterlin A, Gançarski P. A global averaging method for dynamic time warping, with applications to clustering [J]. Pattern Recognition, 2011, 44(3): 678-693

[12] Keogh E, Folias T. The UCR time series data mining archive [OL]. http://www.cs.ucr.edu/ eamonn/time series data

(上接第 246 页)

参 考 文 献

[1] Sabelfeld A, Sands D. Declassification: dimensions and principles [J]. Journal of Computer Security, 2009, 17(5): 517-548

[2] Sabelfeld A, Russo A. Securing interaction between threads and the scheduler [C]// 19th IEEE Computer Security Foundations Workshop. 2006: 177-189

[3] Sabelfeld A. The impact of synchronisation on secure information flow in concurrent programs [M]// Perspectives of System Informatics, LNCS 2244. 2001: 225-239

[4] Smith G, Volpano D. Secure information flow in a multi-threaded imperative language [C]// 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. 1998: 355-364

[5] Mantel H, Reinhard A. Controlling the what and where of declassification in language-based security [M]// Programming Languages and Systems, LNCS 4421. 2007: 141-156

[6] Lux A, Mantel H, Perner M. Scheduler-Independent Declassification [M]// Mathematics of Program Construction, LNCS 7342. 2012: 25-47

[7] 姜励，陈健，平玲娣，等. 多线程程序的信息抹除和降密安全策略[J]. 浙江大学学报(工学版)，2010 (5)：854-862
Jiang L, Chen J, Ping L D, et al, Security policy for information erasing and leaking in multithreaded codes[J]. Journal of Zhejiang University(Engineering Science), 2010(5): 854-862

[8] Focardi R, Gorrieri R, Martinelli F. Non Interference for the Analysis of Cryptographic Protocols[M]// Automata Languages and Programming, LNCS 1853. 2000: 354-372

[9] Russo A, Sabelfeld A. Securing interaction between threads and the scheduler [C]// 19th IEEE Computer Security Foundations Workshop. 2006: 177-189

[10] 朱浩，庄毅，薛羽，等. 基于内容和地点维度的机密信息降级策略[J]. 计算机科学，2012，39(8)：153-157
Zhu H, Zhuang Y, Xue Y, et al, Declassification Policy Based on Content and Location Dimensions[J]. Computer Science, 2012, 39(8): 153-157

[11] Askarov A, Myers A. A semantic framework for declassification and endorsement[M]// Programming Languages and Systems, LNCS 6012. 2010: 64-84

[12] 李沁，袁志祥. 一种宽容的多线程程序内部时间信息流类型系统[J]. 计算机科学，2014，41(3)：163-168
Li Qin, Yuan Zhi-xiang. Permissive Type System for Internal Timing Information Flow in Multi-thread Programs[J]. Computer Science, 2014, 41(3): 163-168