

基于 UML 和模型检测的安全模型验证方法

程 亮^{1),2)} 张 阳²⁾

¹⁾(中国科学技术大学电子工程与信息科学系 合肥 230027)

²⁾(中国科学院软件研究所信息安全国家重点实验室 北京 100190)

摘 要 安全策略的形式化分析与验证随着安全操作系统研究的不断深入已成为当前的研究热点之一. 文中在总结前人工作的基础上,首次提出一种基于 UML 和模型检测器的安全模型验证方法. 该方法采用 UML 将安全策略模型描述为状态机图和类图,然后利用转换工具将 UML 图转化为模型检测器的输入语言,最后由模型检测器来验证安全模型对于安全需求的满足性. 作者使用该方法验证了 DBLP 和 SLCF 模型对机密性原则的违反.

关键词 安全操作系统;系统操作安全;安全策略模型;形式化验证;模型检测;UML

中图法分类号 TP309 **DOI 号:** 10.3724/SP.J.1016.2009.00699

A Verification Method of Security Model Based on UML and Model Checking

CHENG Liang^{1),2)} ZHANG Yang²⁾

¹⁾(Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027)

²⁾(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190)

Abstract As the development of security operating system, the formal analysis and verification of the security models has been one of the hottest topics now. A new method to verify the correctness of security models is proposed in this paper based on the study of predecessors' work, which made use of the Unified Modeling Language (UML) and model checking. This approach first used the UML to specify the security model in the form of finite state machine diagrams and the class diagrams, and then translated these UML diagrams to the input language of model checkers. And it used the model checker to verify the model's correctness or the violation of security properties for the last step. The authors demonstrate the violation of confidentiality of the DBLP and SLCF model by the new approach.

Keywords security operating system; security model; formal verification; model checking; UML

1 引 言

安全策略模型的形式化分析与验证一直是安全操作系统研发中的重点问题^[1-3]. 国标 GB 17859-1999 在第五级中明确提出了需要对安全模型和顶

层规范进行一致性的证明^[4]. 然而,由于安全模型都是采用形式化方法进行描述的,对于实际的软件开发来说,直接采用形式化安全模型作为他们的开发任务是件很困难的事情. 因此,需要一种机制帮助软件开发或者系统管理员在软件设计阶段理解、验证和实施安全模型以及相关策略. 事实上,在软件

设计开发领域中,UML 语言已成为实际上的设计与分析面向对象软件系统的规范语言^①.它对系统不同层次的动态、静态以及结构属性提供了丰富的描述方法.因而,利用 UML 对安全策略模型进行描述以辅助安全系统的设计人员对系统安全属性进行验证的方法获得了国内外广泛的关注.

Koch 和 Parisi-Presicce^[2]最早证明了使用 UML 图形形式化描述安全模型的可能性,并利用图形变换来对访问控制策略进行验证. Park 等人则在文献[5]中使用 Alloy^[6]和 USE^[7]工具验证了一个用 UML 的类图、序列图和对象图描述的 RBAC 策略的实例.之后, Ahn^[8]在其提出的 AMF 框架^[9]中给出了使用 UML 的类图和对象图描述 RBAC 模型、利用 OCL 描述模型策略、利用 Alloy 分析器对安全模型的功能属性和授权属性的验证方法,由于他们只利用了静态 UML 图来描述安全策略,经过转换算法得到的模型也只能表示策略的静态属性,对于检查动态的行为序列中发生的属性违反,该方法并没有很好的办法.与此同时, Ray 在文献[10]中提出了一种可以对特定行为序列中的安全属性的满足性进行验证的方法,该方法采用对象图模拟对象行为的发生顺序,可以部分地检测对象的动态行为,并采用了 OCLE 和 USE 等静态分析工具对 RBAC 模型进行了实验.可惜的是,正如其文中所说,由于这种方法需要手工构建行为序列的场景,如果系统状态太多,该方法的实际操作性就值得商榷了.

目前流行的对 UML 化的安全模型进行验证的 Alloy 等工具,是基于—阶逻辑的轻量级形式化工具,功能简单,无法描述和历史相关的时序属性以及状态机的工作流程,应付操作系统纷繁复杂的安全策略则显得捉襟见肘.相较而言,模型检测器作为一种自动化的验证工具,具有验证速度快、效率高,并且可以得到违反属性的执行路径的优点.这种验证方法通常以安全模型作为输入,用时态逻辑来描述待验证的安全属性,通过遍历模型所有状态的方式检测模型与属性之间的一致性.

Hansen 等人^[11]首先使用模型检测器 SPIN^[12]对 RBAC^[13]的策略与实现之间的一致性进行了验证.之后, Arnaud 等^[3]又采用扩展有限状态机(Extended Finite State Machine, EFSM)对 SAP 业务流程中的 RBAC 策略系统的安全属性进行了验证,并重点讨论了模型检测过程中可能出现的状态爆炸问题.然而,上述方法的建模过程以及安全策略的形式化描述都由手工完成,而且牵涉到大量逻辑

公式,工作效率比较低.

另一方面,文献[14]最早给出了 UML 状态机图的形式化语义,并给出算法将 UML 的状态图子集转化为 PROMELA 语言,而且证明了该算法的正确性. Knapp 等^[15]则给出了可以将 UML 图转换为多种模型检测器输入语言的工具 Hugo/RT. 国内的董威^[16]、张频^[17]也独立地对 UML 模型到模型检测器输入语言的转换算法进行了研究,从理论上证明了将 UML 模型转换为模型检测器输入语言的可能性.但是,他们并没有将这种方法引入到安全策略模型的属性验证中,因此,我们在本文中尝试结合 UML 语言描述和模型检测器对 UML 化的安全策略做形式化验证的方法.

本文第 2 节介绍验证方法的框架;第 3 节是方法各个步骤的详细描述;第 4 节介绍对两个 BLP 模型变种的验证实例;第 5 节是实验结果分析;最后是结论以及下一步的工作计划.

2 模型验证框架

由于目前对于安全策略模型的验证工作或者要求较高的数学理论基础或者验证强度不能保证,因此我们提出一种利用 UML 语言和模型检测来验证安全策略模型与安全需求一致性的方法,这种方法既有一定的易用性也能够保证得到的验证结果理论依据充分.

基于 UML 和模型检测的安全模型验证法的框架图如图 1 所示.

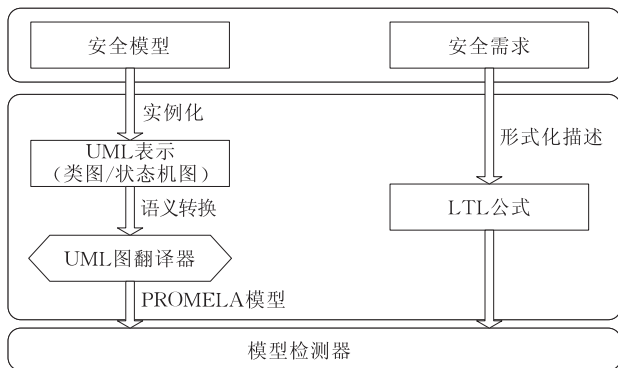


图 1 基于 UML 和模型检测的安全模型验证框架图

一般来说,安全策略模型可以用状态机(Finite State Automata, FSA)模型来描述.在状态机模型中,系统状态集由一个个安全状态组成,安全策略被

① Object Management Group. Unified modeling language specification, Version 1.4. Draft, OMG, 2001. [http://cgi.omg.org/cgi-bin/doc?ad/01-02\(-\)14](http://cgi.omg.org/cgi-bin/doc?ad/01-02(-)14)

定义成约束和限制安全状态迁移的迁移规则。

定义 1. 安全策略模型 M 是一个有限状态自动机(FSA),表示为 $M=(V, V_0, L, T, F, C)$,其中,

- (1) V 是有限状态集合;
- (2) V_0 是初始状态集合, $V_0 \subseteq V$;
- (3) L 是一个有限标签集合;
- (4) T 是状态迁移集合, $T \subseteq (V \times L \times V)$;
- (5) F 是最终状态集合, $F \subseteq V$;
- (6) C 是系统状态 V 和状态迁移 T 的约束,即安全策略。

设策略模型的安全需求为 P_r ,那么,如果有

$$(V, C) \models P_r \text{ 且 } (V_0, C) \models P_r,$$

则称安全策略模型 M 满足安全需求 P_r 。

我们的框架分为 3 个部分:

(1) 将安全策略模型表示转化为 UML 语言的状态机图和类图。在 UML 语言中,类图用于表示类定义了一组有着状态和行为的对象,类图中的属性和关联用来描述状态。而状态机图则是展示状态与状态转换的图。通常一个状态机依附于一个类,并且描述一个类的实例对接受到的事件所发生的反应。当对象探测到一个事件后,它依照当前的状态做出反应,反应包括执行一个动作和转换到新状态^[6]。因此,我们可以用 UML 语言中的状态机图来表示安全策略模型中的状态迁移 T ,策略模型中的安全策略 C 由 UML 状态机图中的守卫条件表示,状态机图中的各个状态则对应了安全策略模型中的各安全状态。安全策略模型中关于安全状态的描述 V 和 V_0 。则可以使用 UML 类图表示。

(2) 我们需要给出 UML 图的形式化语法,以便将安全策略模型的 UML 描述转化为模型检测器的输入语言。

(3) 用模型检测器来验证安全策略模型是否满足待验证的安全属性。这一步需要将待验证的安全属性(安全需求)用模型检测器可以接受的形式表示,如果模型检测器得到该属性的反例,则说明安全策略模型和安全需求是不一致的,根据反例的路径我们就可以找出策略模型中对应的安全状态迁移路径;如果没有,则说明模型可以满足安全需求。

下面,我们将在文章的第 3 节详细介绍该框架中各步骤所使用的方法。

3 基于 UML 和模型检测的验证方法

3.1 安全策略模型的 UML 表示

在安全策略模型中,系统的安全状态通常由主

体、客体和访问方式来表述。典型地,BLP 模型将系统状态描述为 $v \in V$, $V = (b, \mathbf{K}, f)$, 其中 $b \subseteq (S \times O \times A)$ 表示在某个特定的状态下,哪些主体以何种访问方式访问哪些客体, S 是主体集, O 为客体集, $A = \{\text{read}, \text{write}, \text{append}, \text{execute}\}$ 是访问操作集合; \mathbf{K} 表示访问控制矩阵; $f = (\text{level}(s), \text{level}_c(s), \text{level}(o))$ 表示主客体敏感标记函数。

与以往模型检测所验证的系统行为模型不同的是,安全策略模型是高度抽象的,不可能也不应该对系统状态、访问控制矩阵这些抽象数据结构的基数给出具体定义或做出限定。而就模型检测器而言,虽同样要求其验证对象为状态机,但对状态的数量有严格的要求,否则会导致最让人头疼的状态爆炸问题^[5]。另一方面,UML 语言的状态机图也是基于类的实例的。因此,考虑到便于模型检测器的使用,我们在用 UML 描述安全模型时就必须确定系统状态数量、访问控制矩阵等一系列的抽象数据结构。

由于模型检测器一般以有限状态机为输入对象,如果以如 BLP 模型中定义的系统状态作为模型检测器的输入状态机的话,首先要面临的问题就是,每一个 $\beta \in b$, $b \subseteq (S \times O \times A)$, 都对应一个不同的系统状态,这样,系统状态的数量直接取决于模型实例中 (s, o, a) 三元组的数量,要在 UML 图中画出所有的这些状态,工作量太大,甚至不可操作。因此,我们采用一个 UML 状态机图来表示模型中主体对客体提出访问要求(客户端),而用另一个 UML 状态机依据安全模型的安全策略来对主体提出的访问要求进行仲裁(服务器端)。这种描述方法的好处在于,避免了在 UML 状态机图中穷举所有的 (s, o, a) 三元组,减少了 UML 描述的工作量。

客户端模拟主体 $s \in S$ 对访问权限的申请过程:先经初始化以获得自身的安全级别,然后随即获取要访问的客体 $o \in O$ 以及访问操作 $a \in A$,接着将访问申请 (s, o, a) 交由访问仲裁机构仲裁,若批准,则实施状态迁移,进入下一个系统状态,而无论申请成功与否,都会进入第二次申请,两次申请的客体及申请权限都可能不一样,如此循环。服务器端则依据模型的策略描述以及访问控制矩阵对客户端提出的申请进行仲裁。如图 2 所示。

3.2 UML 模型到模型检测输入的转化

UML 到模型检测器输入的转化的关键在于状态机图的转化。安全策略模型的动态行为都在 UML 状态机图中描述。由于当前的模型检测器大多使用符号执行抽象地遍历系统的状态空间,因此,

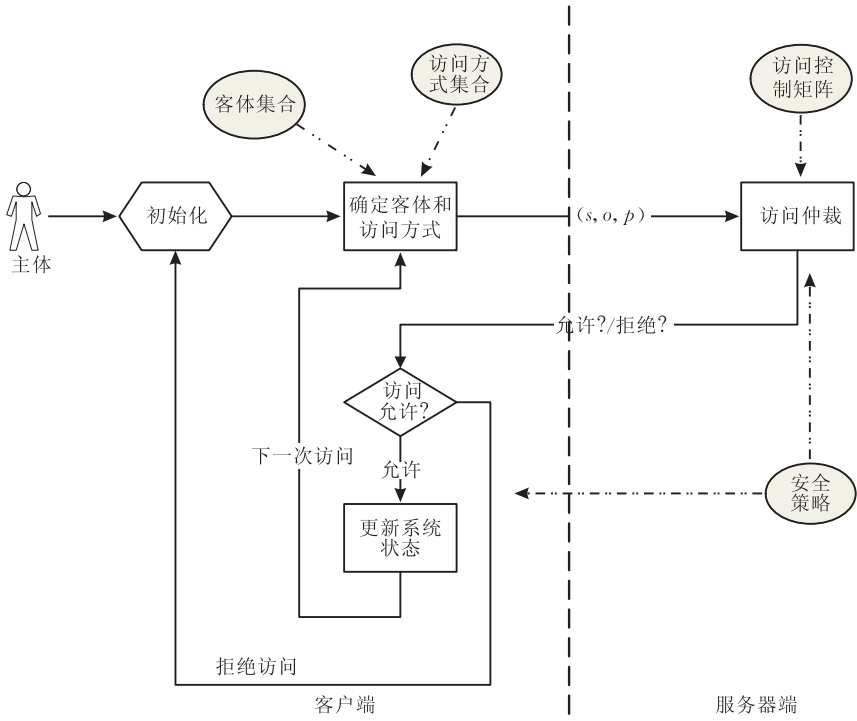


图 2 安全状态迁移模式图

为了验证策略模型的正确性,我们需要将 UML 状态机图转化为模型检测器的输入语言. 我们采用了文献[18]的执行算法思想来描述 UML 状态机图的语义. 这种语义描述基于 UML v1.5 版本的规范, 具体描述如下.

定义 2. 状态 s 包括:

- 种类 $kind$, $kind(s) \in \{init, final, simple, composite, concurrent, junction, join, fork\}$;
- 一个进入动作 $entry(s) \in Act$;
- 一个退出动作 $exit(s) \in Act$.

其中, Act 为动作集合. 对于伪状态, 有 $kind(s) \in \{init, junction, join, fork\}$, $entry(s) = skip$, $exit(s) = skip$; 复合状态则有 $kind(s) \in \{composite, concurrent\}$.

定义 3. 状态层次由树 (S, E) 给出, S 为状态的有限集合, $E \subseteq S \times S$, 我们记 $substates(s) = \{s' \in S \mid (s, s') \in E\}$ 表示状态 s 的子状态, 那么 E 为非空子状态关系, 且其满足下列约束:

- (1) 如果 $substates(s) \neq \emptyset$, 那么 $kind(s) \in \{composite, concurrent\}$;
- (2) 如果 $kind(s) = concurrent$, 那么对于 $\forall s' \in substates(s)$, 有 $\# substates(s) \geq 2$, 并且 $kind(s') = composite$;
- (3) 如果 $kind(s) = composite$, 那么 $\#\{s \in substates(s) \mid kind(s') = initial\} \leq 1$.

定义 4. 给定状态层次 $H = (S, E)$, 其中, H 的迁移 t 包含: 源状态 $source(t) \in S$; 目标状态 $target(s) \in S$; 触发事件 $trigger(t) \in Event$ ($Event$ 为事件集合, 其中完成事件记为 $*$); 守卫表达式 $guard(t) \in Exp$ (Exp 为表达式集合) 以及效果动作 $effect(t) \in Act$, 那么 H 满足下列约束:

- (1) $kind(source(t)) \neq final$ 并且 $kind(target(t)) \neq initial$;
- (2) 如果 $kind(source(t)) \in \{initial, fork\}$, 那么 $target(t)$ 不是伪状态;
- (3) 如果 $kind(source(t)) = initial$, 那么 $container(target(t)) = container(source(t))$;
- (4) 如果 $kind(target(t)) = join$, 那么 $source(t)$ 不是伪状态;
- (5) 如果 $kind(source(t)) = composite$, 那么 $kind(container(source(t))) \neq concurrent$;
- (6) 如果 $kind(target(t)) = composite$, 那么 $kind(container(source(t))) \neq concurrent$;
- (7) 如果 $kind(source(t)) \in \{initial, fork, join\}$, 那么 $guard(t) = true$;
- (8) 如果 $kind(target(t)) = join$, 那么 $effect(t) = true$;
- (9) 如果 $kind(source(t)) = initial$, 那么 $effect(t) = skip$;
- (10) 如果 $source(t)$ 是伪状态, 那么 $trigger(t) = *$.

定义 5. 状态机由二元组 (H, T) 给定, 其中, $H = (S, E)$ 为状态层次, T 为 H 迁移的有限集合, 记 $outgoings(s) = \{t \in T \mid source(t) = s\}$, $incomings(s) = \{t \in T \mid target(t) = s\}$, $source(M) = \{source(t) \mid t \in M\}$, $target(M) = \{target(t) \mid t \in M\}$, 那么, 对于 $\forall t \in T$, 它满足下列约束:

- (1) 如果 $kind(t) = initial$, 那么 $\#outgoings(s) = 1$;
- (2) 如果 $kind(t) = junction$, 那么 $\#incomings = 1$ 且 $\#outgoings(s) \geq 1$;
- (3) 如果 $kind(t) = fork$, 那么 $\#incomings = 1$ 且 $\#outgoings(s) \geq 2$;
- (4) 如果 $kind(t) = fork$, 那么存在 $s' \in S$, $kind(s') = concurrent$, 使得 $targets(outgoings(s)) \subseteq substates^+(s') \setminus substates(s')$, 并有, 如果 $t, t' \in outgoings(s)$, 使得某些 $s'' \in substates^+(s')$ 满足 $\{target(t), target(t')\} \subseteq substates^+(s'')$, 那么 $t = t'$.
- (5) 如果 $kind(t) = join$, 那么将情况(3)、(4)中的 $target$ 替换成 $source$ 、 $outgoing$ 替换成 $incoming$ 时也成立.

情况(4)、(5)需要从同一个并发复合状态的不同正交区域中 $fork$ 出来或 $join$ 到同一个并发复合状态的不同正交区域中.

3.3 安全策略模型的模型检测

模型检测是基于时态逻辑的, 所以为了检测定义 1 中公式的满足性, 模型检测器还需要将待验证的安全需求描述成时序逻辑公式的形式. 应用于模型检测的时态逻辑分为两种: 线性时态逻辑 (LTL) 和计算树时态逻辑, 由于我们下面将使用 SPIN 来验证安全模型, 所以我们采用了 LTL 来描述安全需求.

我们以机密性为例. 机密性的目标是为了防止非可信主体将高机密级别的信息泄露给低机密级别的主体. 虽然机密性要求任何由高等级向低等级的信息流都不能出现, 这其中也包括了隐通道在内, 但是, 一般来说, 安全策略模型旨在控制各类显式非法信息在非可信主体与客体间传递, 隐通道并不在其考虑范围内^[19]. 因此, 机密性要求的 LTL 描述如下.

定义 6. 定义

$readlike: O \times S \rightarrow \{\text{True}, \text{False}\}$, 判断主体 s 是否以只读操作访问客体 o , 如果是, 则返回 True; 否则为 False;

$writelike: O \times S \rightarrow \{\text{True}, \text{False}\}$, 判断主体 s 是否以只写操作访问客体 o , 如果是, 则返回 True; 否

则为 False;

记 s 为非可信主体, $O(s)$ 表示主体 s 所访问过的客体集合, $level_c(o)$ 表示客体 o 当前安全敏感标记. 那么, 机密性要求表示如下:

$$\rightarrow \exists o_1, o_2 \in O(s) \cdot (readlike(o_1) \rightarrow$$

$$\Diamond writelike(o_2)) \wedge level_c(o_2) \leq level_c(o_1),$$

即不允许出现主体 s 先读高等级的客体再向低等级客体写入数据的情况.

下面, 我们以两个安全策略模型 DBLP^[20] 和 SLCF^[21] 为例, 来说明如何利用 UML 和模型检测器来验证安全策略模型的正确性.

4 两个改进型 BLP 模型的机密性验证

BLP 模型作为安全策略模型的鼻祖, 在最初提出时, 人们希望该模型规则应用于系统中的所有主体^[22], 然而, 很快人们就在实践中发现由于 $*$ -属性的限制过严, BLP 模型无法构造实际中的安全系统: 在实际的计算机系统中, 双向信息流动是不可避免的, 一个系统如果只有单向信息流, 将导致系统的不可用^[23]. 为了保证模型的实际可用性, 人们在 BLP 中加入了可信主体的概念, 并做了多方改进以提高其实用性, DBLP 和 SLCF 则是其中的两种.

4.1 DBLP 的验证

DBLP 模型引入了动态调整敏感标签范围的规则; 引入单级客体、多级客体以及专用于 IPC 的访问模式, 并给出它们应满足的不变量及限制性条件; 从而使动态特征非常明显的 IPC 对象得到合理、有效管理的机制^[20].

4.1.1 DBLP 简介

提出 DBLP 的目的在于通过增加动态调整非可信主体敏感级的规则来提高原始 BLP 模型的实用性. DBLP 模型采用了 BLP 模型的系统描述方法, 但它改进了主客体敏感级标记. 在 DBLP 中, 主体有 3 个敏感级标签: f_s (主体最大敏感级标签)、最小可写敏感级 a_min_s 和最大可读敏感级 v_max_s , 而客体具有最大敏感级标签 L_min_o 和最小敏感级标签 L_max_o .

DBLP 模型定义了单级客体和多级客体, 分别记为 $single(o)$ 和 $multi(o)$. 单级客体是可信主体和非可信主体共同访问的对象, 而多级客体仅是可信主体访问的对象. 非可信主体可以访问 $L_min_o(o) \neq L_max_o(o)$ 的客体, 只要非可信主体对该客体的访问仅由该客体安全级范围的一个安全级确定即可.

DBLP 模型的描述以及其系统状态迁移规则详见文献[20], 本文不再复述.

4.1.2 DBLP 的 UML 描述

首先是模型的实例化. 我们构造了典型的 4 级军事机密安全模型的 DBLP 实例. 为了便于描述, 我们将敏感级集合取为 $\{1, 2, 3, 4\}$, 敏感程度按偏序“ \leq ”依次上升, 敏感级 4 对应最高机密信息, 而敏感级 1 则对应非机密信息. 非可信主体 s 初始时拥有全部敏感级的全部访问权限.

根据 3.1 节的安全策略模型的 UML 表示法, 系统状态需要保存的状态变量有

- (1) 主体最小可写敏感级 a_{min_s} 和最大可读敏感级 v_{max_s} ;
- (2) 客体最大最小敏感级标签 L_{min_o} 和 L_{max_o} ;
- (3) 客体当前敏感级标签 $level_c$.

由于要比较 s 读/写过的客体的当前敏感级, 还需要引入变量 $read_level$ 、 $write_level$ 记录之. 完整的系统状态与仲裁机构的 UML 类图描述如图 3

所示.

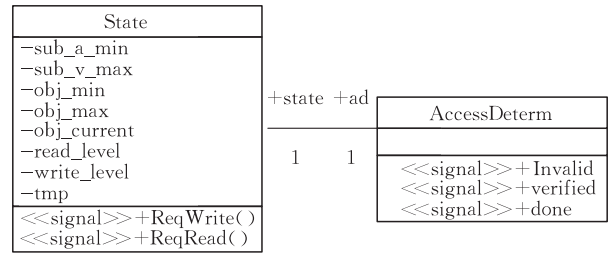


图 3 DBLP 模型 UML 描述类图

其中, 系统状态中的 tmp 为临时变量, 用于交换敏感级之用, 信号 $ReqWrite$ 、 $ReqRead$ 为主体向仲裁机构发送的读写请求; 仲裁机构中的信号 $Invalid/verified$ 表示主体的访问申请被否定或批准.

仲裁机构的状态机图如图 4 所示. 下半部分是对只读申请的判定; 上半部分则是对只写申请的判定. 判定规则由 DBLP 的 3 条迁移规则决定, 表示为状态转换的守卫条件.

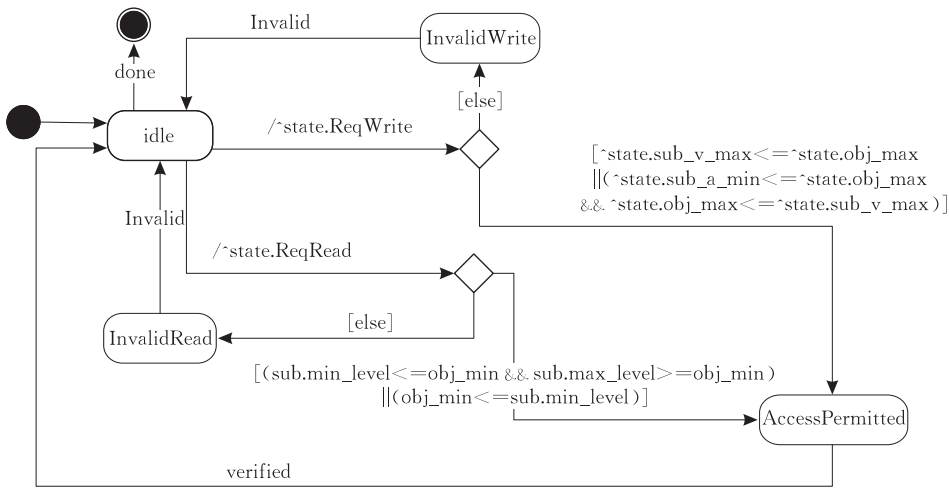


图 4 访问控制仲裁机构的状态机图

系统状态迁移的状态机图如图 5 所示. 图 5 为图 2 系统状态迁移模式图的 UML 具体实现, 需要注意的是其中的 $SelectObj$ 子状态机图. 其目的是为主体随机挑选出一个客体进行访问——随机生成客体的 L_{min_o} 、 L_{max_o} 以及 $level_c$.

4.2 SLCF 的验证

SLCF 在 BLP 模型的基础上, 做了如下改进:

- (1) 除主体的两个标记函数 f_c 和 f_s 以外, 还为主体增加了 4 个标记函数 $(f_{il}, f_{ih}, f_{ol}, f_{oh})$, 它们分别表示在一个进程的生命周期内流入信息的最低标记、流入信息的最高标记、流出信息的最低标记、流出信息的最高标记, 用于记录主体的访问历史.

- (2) 为客体增加了一个标记函数 f_d , f_d 把客体标记 f_o 映射为易于理解的信息(一般是一组字符串), 用于客体信息的输出(如显示或发布信息).

SLCF 在实施时首先将主体标记初始值置为 $f_{il} = f_{ih} = LOW$ (系统的最小标记值), $f_{ol} = f_{oh} = HIGH$ (系统的最大标记值). 然后根据自己的访问控制规则来实现多级安全.

由于和 DBLP 模型的主要差别在于主客体的标记设置和访问控制规则, 因此建模方法可以沿用上述验证 DBLP 的方法, 只需将其 UML 描述类图中的系统状态变量以及两个状态机图中迁移规则的守卫条件根据 SLCF 做相应替换即可. SLCF 的

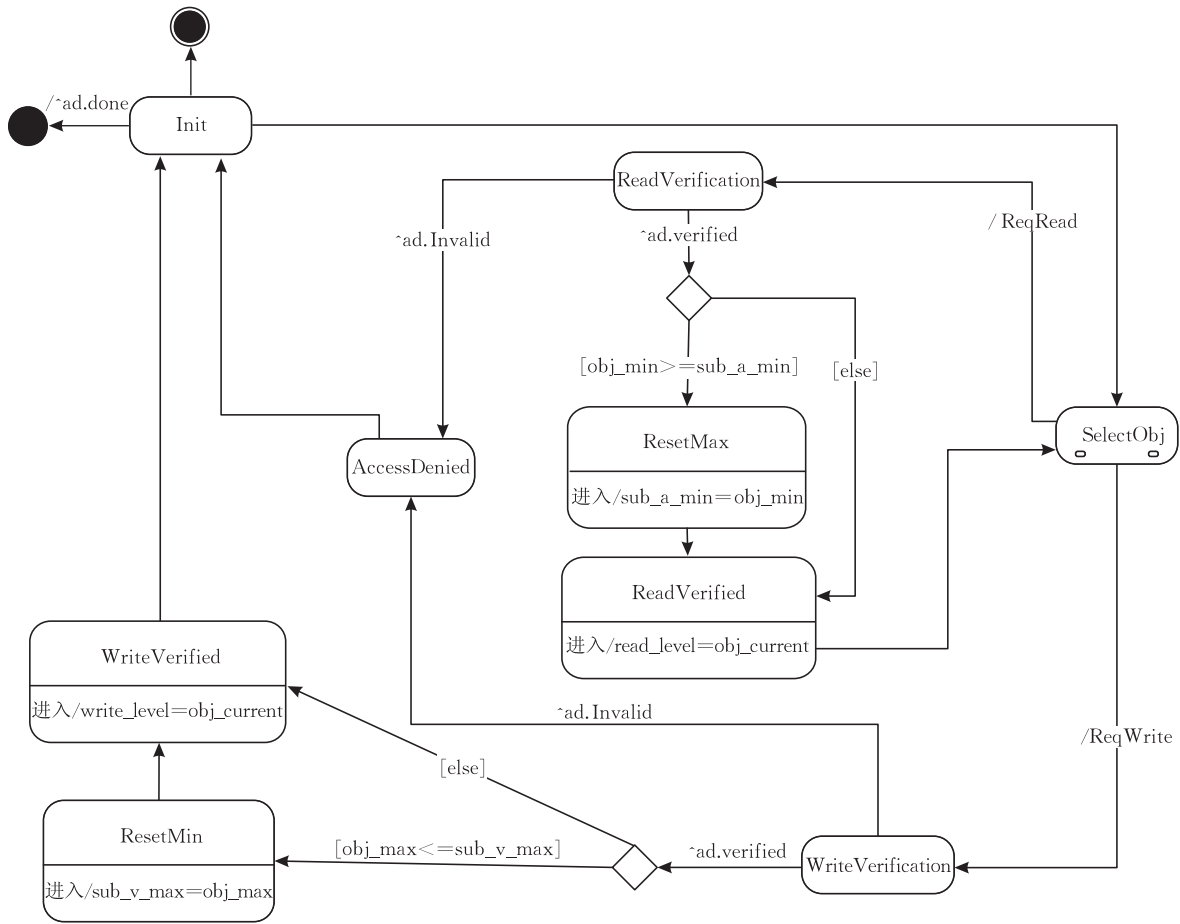


图 5 主体申请对客体访问的状态机图

访问控制规则请参考文献[21],在此不再赘述.

4.3 安全需求的描述

这个两个 BLP 的改进模型意在提高 BLP 的实用性,这一点应当基于不损害其基本机密性原则的基础上,也即,DBLP 应当满足定义 6 描述的机密性属性.我们在验证中使用的模型检测器 SPIN 提供了 assertion 断言来描述 LTL 公式.定义 6 用 assertion 断言描述如下:

```
assertion {
    G (state. read_level>state. write_level);
}
```

其中,G 为 LTL 算子,表示“所有未来状态”.

5 实验结果分析

我们对本文描述的方法进行了实验.考虑到 UML 描述的一致性,UML 语言版本采用 1.5 版,UML 建模模型翻译工具采用 Hugo/RT^[15],版本 0.42.实验结果分述如下.

5.1 DBLP

我们首先将 4.2 节所示的 DBLP 模型的 UML 描述用 argoUML 绘出.考虑到 Hugo/RT 对部分 UML 功能支持得不是很好,需要手工输入待验证安全属性,所以第 2 步用 Hugo/RT 将 UML 模型转换为 UTE 格式的中间代码.接着,在 UTE 文件中加入 4.3 节中的 assertion 断言.第 4 步,再利用 Hugo/RT 将加入了断言的 UTE 文件转换为 PROMELA 语言文件.最后,利用 SPIN 模型检测器对模型进行验证.

实验结果显示:模型并不满足加入的 assertion 断言,也就是说 DBLP 确实存在违反定义 6 的路径.

```
warning: never claim + accept labels requires -a flag
to fully verify
hint: this search is more efficient if pan.c is compiled
-DSAFETY
pan: claim violated! (at depth 492)
pan: wrote dblp.spin.trail
(Spin Version 5.1.3 -- 11 December 2007)
```

Warning: Search not completed

Full statespace search for:

never claim +

assertion violations+(if within scope of claim)

acceptance cycles-(not selected)

invalid end states-(disabled by never claim)

State-vector 128 byte, depth reached 551, errors: 1

390 states, stored

211 states, matched

601 transitions (=stored+matched)

590 atomic steps

hash conflicts: 0 (resolved)

2.501 memory usage (Mbyte)

pan: elapsed time 0.01 seconds

根据 SPIN 得到的 trail 文件,我们描述属性违反路径如下:

对于上节描述的 4 级军事机密系统以及 $a_{min_s}=v_{max_s}=4$ 的非可信主体 s ,违反定义 6 的属性的行为路径为

(1) s 先只读访问客体 $\{L_{min_o}=2, L_{max_o}=4, level_c(o)=3\}$, 则其敏感级变为 $\{a_{min_s}=2, v_{max_s}=4\}$;

(2) s 只写访问客体 $\{L_{min_o}=2, L_{max_o}=2, level_c(o)=2\}$, 由于此时 $a_{min_s}(s) \leq L_{max_o}(o) \leq v_{max_s}(s)$, 允许写, 而主体敏感级变为 $\{a_{min_s}=2, v_{max_s}=4\}$.

然而,整个过程中非可信主体先读取了敏感级为 3 的客体的信息,接着又向敏感级为 2 的客体写入内容,造成了信息从高敏感级向低敏感级的流动,显然违背了定义 6 中的机密性原则.也就是说, DBLP 并不满足机密性原则,从而实验证明了我们提出方法的有效性,事实上,这与文献[22]的结果是一致的.

5.2 SLCF

同样的,SLCF 也被发现不满足 4.3 节的安全需求.根据 SPIN 的 trail 文件我们可以构造出如下的违反路径:

(1) 主体 s 拥有权限 $f_c(s)=4, f_{ih}(s)=1, f_{ol}(s)=4$, 并只读访问客体 $\{f_o(1)=4\}$, 根据 SLCF 规则, s 敏感级不变;

(2) 接着, s 只写访问客体 $\{f_o(2)=2\}$, 由于此时 $f_{ih}(s) \leq f_o \leq f_c(s)$, 写操作允许, 从而主体敏感级变为 $\{f_c(s)=2, f_{ol}(s)=2\}$.

显然,这样的操作造成了高敏感级客体的信息

($f_o(1)=4$)向低敏感级客体($f_o(2)=2$)的流动,是不满足机密性要求的.事实上,SLCF 的访问规则对写操作的判定是根据 $f_{ih}(s)$ 进行的,而由于 f_{ih} 初始化为系统最低敏感级 LOW,无论 s 之前对任何等级的客体进行过读操作,之后的写访问请求总是被许可的.这种设定显然是危险的.

6 小 结

本文在前人工作的基础之上提出了一种基于 UML 和模型检测的安全策略模型验证方法.本文的主要贡献在于:首次将安全策略模型用 UML 类图和状态机图描述成有限状态自动机模型,并利用工具将 UML 描述转化为模型检测器的输入语言,由模型检测器验证安全策略模型的正确性.本文旨在利用自动化工具辅助安全模型的正确性验证,减少传统安全模型验证方法中对验证人员形式化理论的过高要求,以及繁重的手工推导过程.

文中给出的安全模型的 UML 建模方式依赖于实际待验证的安全属性,只能验证单一的安全属性.对于 DBLP,该建模方法得到的系统安全状态迁移模型仅能对安全模型的机密性进行检验.在实际应用中,可以考虑将多个待验证属性的系统状态迁移图视为同一正交复合状态中的不同的子状态机并发地进行验证.今后,将进一步研究系统安全状态迁移的建模方式以及化简安全状态迁移模型的方法,并在此基础上扩展该方法对不同安全策略模型以及不同待验证安全属性的有效性与适用性.

参 考 文 献

- [1] Tidswell J E, Jaeger T. An access control model for simplifying constraint expression//Proceedings of the 7th ACM Conference on Computer and Communications Security. Athens, Greece, 2000: 154-163
- [2] Koch M, Parisi-Presicce F. Visual specifications of policies and their verification//Proceedings of the Workshop on Fundamental Approaches to Software Engineering. Barcelona, Spain, LNCS 2621. 2003: 278-293
- [3] Arnaud Dury, Boroday Sergiy, Petrenko Alexandre, Lotz Volkmar. Formal verification of business workflows and role based access control systems//Proceedings of the International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007). Valencia, Spain, 2007: 201-210
- [4] General Administration of Quality Supervision, Inspection and

- Quarantine of PRC, National Standards of the People's Republic of China, Classified Criteria for Security GB 17859 — 1999, September, 1999(in Chinese)
(中华人民共和国国家标准. 计算机信息系统安全保护等级划分准则. 中国国家质量技术监督局, 1999 年 9 月 13 日发布, 2001 年 1 月 1 日实施)
- [5] Park Sachoun, Kwon Gihwon. Verification of UML-based security policy model//Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2005). Singapore, 2005; 973-982
- [6] Jackson D. Alloy: A lightweight object modeling notation. *ACM Transactions on Software Engineering and Methodology*, 2002, 11(2): 256-290
- [7] Richters M, Gogolla M. Validating UML models and OCL constraints//Proceedings of the 3rd International Conference on the Unified Modeling Language (UML 2000). York, UK, 2000; 265-277
- [8] Hu Hongxin, Ahn Gail-Joon. Enabling verification and conformance testing for access control model//Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (SACMAT). Estes Park, Colorado, USA, 2008; 195-204
- [9] Ahn G J, Hu H. Towards realizing a formal RBAC model in real systems//Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT '07). New York, USA, 2007; 215-224
- [10] Yu Lijun, France Robert, Ray Indrakshi. Scenario-based static analysis of UML behavioral properties//Proceedings of the ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems. Toulouse, France, 2008; 234-248
- [11] Hansen F, Oleshchuk V A. Conformance checking of RBAC policy and its implementation//Proceedings of the 1st Information Security Practice and Experience Conference (ISPEC 2005). Singapore, 2005; 144-155
- [12] Holzmann G J. The SPIN model checker. *IEEE Transactions on Software Engineering*, 1997, 23: 279-295
- [13] Ferraiolo D F, Sandhu R S, Gavrila S I, Kuhn D R, Chandramouli R. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 2001, 4(3): 224-274
- [14] Latella D, Majzik I, Massink M. Automatic verification of a behavioral subset of UML statechart diagram using SPIN model-checker. *Formal Aspects of Computing*, 1999, 11(6): 637-664
- [15] Knapp Alexander, Merz Stephan. Model checking and code generation for UML state machines and collaborations//Dominik Haneberg, Gerhard Schellhorn, Wolfgang Reif eds. *Proceedings of the 5th Workshop on Tools for System Design and Verification*. Technical Report 2002-11, Institut für Informatik, Universität Augsburg, 2002; 59-64
- [16] Dong Wei, Wang Ji, Qi Zhi-Chang. An approach of model checking UML Statecharts. *Journal of Software*, 2003, 14(4): 750-756(in Chinese)
(董威, 王戟, 齐治昌. UML Statecharts 的模型检验方法. 软件学报, 2003, 14(4): 750-756)
- [17] Zhang Pin, Luo Gui-Ming. Research of model checking UML. *Computer Applications*, 2007, 27(10): 2493-2497(in Chinese)
(张频, 罗贵明. UML 模型检测方法的研究. 计算机应用, 2007, 27(10): 2493-2497)
- [18] Lilius J, Paltor I P. Formalizing UML state machines for model checking//France R, Rumpe B eds. *Proceedings of the 2nd International Conference on Unified Modeling Language (UML'99)*. Lecture Notes in Computer Science 1723. Berlin: Springer, 1999; 430-445
- [19] Gasser M. *Building a Security Computer System*. New York: Van Nostrand Reinhold Company, 1988
- [20] Ji Qing-Guang, Qing Si-Han, He Ye-Ping. An improved dynamically modified confidentiality policies model. *Journal of Software*, 2004, 15(10): 1547-1557(in Chinese)
(季庆光, 卿斯汉, 贺也平. 一个改进的可动态调节的机密性策略模型. 软件学报, 2004, 15(10): 1547-1557)
- [21] Liang Hong-Liang, Sun Yu-Fang, Zhao Qing-Song, Zhang Xiang-Feng, Sun Bo. Design and implementation of a security label common framework. *Journal of Software*, 2003, 14(3): 547-552(in Chinese with English Abstract)
(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)
- [22] Bell D E, La Padula L J. Secure computer system: A retrospective//Proceedings of the 1983 IEEE Symposium on Security and Privacy. Oakland, California, 1983; 161-162
- [23] He Jian-Bo. Study on the key issues of security policy models in secure operating system[Ph. D. dissertation]. Institute of Software, Chinese Academy of Sciences, Beijing, 2007 (in Chinese)
(何建波. 安全操作系统策略模型的关键问题研究[博士学位论文]. 中国科学院软件研究所, 北京, 2007)



CHENG Liang, born in 1982, Ph. D. candidate. His research interests include secure operating system and its verification.

ZHANG Yang, born in 1971, Ph. D.. Her research interests focus on secure operating system and its verification.

Background

The work attributes to the project “Security Analysis of Windows Vista Kernel and Operating System Controllability Research”, which is supported by the National High Technology Research and Development Program (863 Program) of China under grant Nos. 2007AA01Z465, 2006AA01Z433, 2007AA01Z475, 2007AA01Z414.

Since the UML is nowadays widely used in software development and it does not require complicated foundations of mathematics, the UML representation of security model has been adopted in the design and development of software for years. However, the verification of such “UMLized” security policy model is still employing Alloy or USE, which is a lightweight formal tool with poor function and can not deal

with the temporal properties and workflow of state machines.

In this paper, the authors introduce model checking to the verification of UML representation of security policy model and propose a new method of representing the security model by UML. The approach uses UML class diagrams and state machine diagrams to specify both the static and dynamic properties of security model fully and then translates these diagrams into model checkers’ input language in order to employ model checkers’ power of formal verification. At last, The authors demonstrate our approach’s validity by verifying the confidentiality property of DBLP security model.