基于抽象和搜索空间划分的安全性判定方法

王昌达 华明辉 周从华 宋香梅 鞠时光
(江苏大学计算机科学与通信工程学院 镇江 212013)

摘 要 为满足访问控制策略安全性快速判定的要求,提出一种基于谓词抽象和验证空间划分的访问控制策略状态空间约减方法,将在访问控制策略原始状态机模型上的安全性分析工作转移到包含较少状态的抽象模型上,并进一步划分抽象模型的验证空间,以提高效率。理论分析和实验数据均表明,其安全性分析所需的时间和空间都得到有效约减。与传统方法相比,它具有速度更快、自动化程度更高等优点。

关键词 访问控制,谓词抽象,安全性分析,模型检测,可信评估
中图法分类号 TP309.2 **文献标识码** A

Security Analysis of Access Control Policy Based on Predicate Abstract and Verification Space Division

WANG Chang-da HUA Ming-hui ZHOU Cong-hua SONG Xiang-mei JU Shi-guang
(School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, China)

Abstract In order to implement security analysis of access control policy rapidly, predicate abstract with verification space division was presented, i. e. transfer pristine state machine model analysis to abstract state machine model which contains fewer states. Furthermore, verification space division was introduced to decrease the dimensions of model checking. Endorsed by both theoretic analysis and experiment, time and space requirement are effectively reduced. Compared with the known methods, our methodology is more efficiency and less human interacted.

Keywords Access control, Predicate abstract, Security analysis, Model checking, Trust evaluation

1 引言

可信计算要求系统服务的可信性具有可论证性[1],访问控制策略的安全性分析是系统服务可信性证明的重要组成部分[2]。目前,通过网络和计算机系统向用户提供的服务日趋丰富,尤其是随着电子商务和网络银行业务的延伸,管理不同类型的用户对系统资源的访问控制权限正变得越来越复杂。在这种情况下,功能相对单一的静态访问控制策略很难满足系统的可用性要求。访问控制可定制系统要求提供对访问控制状态配置的可修改能力[3],即要求系统支持对已有访问控制策略的增加和删减操作,这必然导致访问控制策略的动态调整,进而影响其安全性。为满足这种动态调整的需求,对访问控制策略的安全性进行快速判定就变得非常必要。

访问控制策略的安全性分析包含两方面内容:(1)给定主体对给定客体是否一定能获得它应具有的权限;(2)给定主体对给定客体是否一定不能获得它不应具有的权限[4]。目前用于判定访问控制策略安全性的分析手段主要有3种:形式化方法、定理证明器和模型检测[5]。使用形式化方法和定理证明器的主要问题是:在分析的过程中,需要较多的人工干预或诱导来把握证明推理的方向,自动化程度低、对分析者的专业素质要求高、分析周期长。模型检测是一种在相应工具软件的支持下,通过遍历系统所有状态空间对有穷状态系统进行自动验证,并可构造出不满足验证性质反例的方法。使用模型检测方法分析访问控制策略的安全性,具有自动化程度高的优点。这种方法在验证小规模的访问控制策略方面取得了成功,如英国伯明翰大学、美国得克萨斯州立大学和北卡罗莱纳大学等研究机构在此领域的工作[5-8],Ravi S. Sandhu 的TAM模型[6],文献[7,8]中的UCON模型等。这些模型对于规模较大的访问控制策略来说,状态空间的组合爆炸使得遍历系统所有状态空间的愿望在事实上无法实现,并制约了模型检测方法的使用范围。例如,Li 和 Tripunitara 在文献[9]中证明访问控制中的"创建可信用户(AATU)"和"授权及其回收(AAR)"等问题是coNP完全问题,具有极高的计算复杂性。文献[10]运用谓词抽象技术成功地验证了硬件系统,相对增强了其处理大规模设计的能力。文献[11]设计了内部电路节点来构造状态搜索空间,也起到了约减模型状态数的作用,但其效率不高,后期的安全性验证也不方便。

本文提出一种基于谓词抽象技术和验证空间划分的访问控制策略状态空间约减方法,将在访问控制策略原始状态机模型上的安全性分析工作转移到包含较少状态的抽象模型上,进一步划分抽象模型的验证空间,约减验证规模,并借助模型检测工具进行安全性验证,从而实现快速判定访问控制

到稿日期:2010-11-17 返修日期:2011-05-05 本文受国家自然科学基金(60773049),江苏省自然科学基金(BK2007086),江苏省高校自然科学研究计划(07KJB520016),江苏省六大人才高峰项目(1631170006)和江苏大学高级人才项目(07JDG053)资助。
王昌达(1971—),男,博士,副教授,主要研究方向为信息安全技术,E-mail:changda@ujs.edu.cn;**华明辉**(1985—),男,硕士生,主要研究方向为信息安全技术。

策略安全性的目标。

2 抽象解释理论框架

抽象是目前解决或者缓解状态空间爆炸最有效的方法之一。通过删除原始模型中与验证无关的信息，可得到一个简化的抽象模型，属性验证工作在抽象模型中进行，由于状态空间较小，因此验证效率大大提高，为计算机中的不可判定问题和复杂问题的逼近求解提供了系统性的构造方法和有效算法。

访问控制策略通常可以描述为状态迁移系统。在此基础上，将访问控制语义（操作语义或者指称语义）定义为访问控制对象域上的计算过程或结果。对访问控制策略的抽象解释就是指使用另一个抽象对象域上的计算抽象逼近具体对象域（原始访问控制策略指称的对象域）上的计算，使得抽象解释的执行结果能够反映出具体对象域中的部分结果。在抽象对象域中得以保持的具体对象域中的属性，被称为不动点语义。例如，主体 A 对客体 B 在原始访问控制策略状态迁移系统中的访问控制权限在抽象模型中若维持不变，那么 A 对 B 的访问控制权限就是这个抽象解释中的一个不动点语义。

数学上有很多种抽象方法，如数学分析中的符号函数 sign(x)，若 x 是实数，则其取值只有 $-1,0,1$ 3 种状态，即将所有的实数按其符号（负，零，正）进行抽象。sign(x)×sign(y)的计算结果与 $x\times y$ 拥有相同的符号，但其计算量远小于通过计算 $x\times y$ 获得其符号的方式。符号的保有关系是 sign(x)抽象的不动点语义。

对于状态迁移系统而言，抽象可以直观地理解为合并某些局部状态，这样可由一个较大的、迁移关系复杂的系统构造出一个较小的、迁移关系相对简单的系统。事实上，为了将原始状态机模型上的安全性分析工作转移到包含较少状态的抽象模型上进行，抽象模型必须包含待分析属性的不动点语义，否则在抽象模型上的分析不能得到正确结果。即使抽象模型包含待分析属性的不动点语义，也希望抽象模型的规模尽可能小。因此抽象必须遵循一定的原则。

下面给出不动点语义基于 Galois 抽象解释的理论框架[12]，其作用是保证按着某种方法对系统抽象能在抽象系统中保持期望的不动点语义。

定义 1（完备格） 设$\langle L,\subseteq\rangle$是一个偏序，$\langle L,\subseteq\rangle$称为一个完备格，当且仅当 L 中任何一个子集均存在最小上界和最大下界。特别地，用 $\downarrow=\bigcup\phi=\bigcap L$ 表示 L 中最小元素，用 $\uparrow=\bigcup\phi=\bigcap L$ 表示 L 中的最大元素，其中 $\bigcup$ 和 $\bigcap$ 表示最小上界算子和最大下界算子。完备格一般表示为$\langle L,\subseteq,\bigcup,\bigcap,\downarrow,\uparrow\rangle$。

定义 2（Galois 连接） 设$\langle P,\leqslant\rangle$和$\langle Q,\subseteq\rangle$是两个偏序集合，$\alpha:P\to Q$ 和 $\gamma:Q\to P$ 是两个映射，序偶$\langle\alpha,\gamma\rangle$称为一个 Galois 连接，当且仅当 $\forall x\in P,\forall y\in Q,\alpha(x)\subseteq y$ 当且仅当 $x\leqslant\gamma(y)$，其中 α 称为抽象算子，γ 称为具体算子。

可以验证，抽象算子 α 和具体算子 γ 具有如下性质：

(1) $\forall x\in P,x\leqslant\gamma(\alpha(x))$；

(2) $\forall y\in Q,\alpha(\delta(y))\subseteq y$；

(3) α 和 γ 单调递增。

抽象算子 α 和具体算子 γ 具有的上述性质与 Galois 连接的定义等价。抽象算子 α 刻画了具体对象域和抽象对象域之间的最优上界抽象逼近映射关系，具体算子 γ 的作用则正好相反。

Galois 连接的运算具有如下性质：

(1)两个 Galois 连接的合成是 Galois 连接；

(2)Galois 连接的广义笛卡尔乘积是一个 Galois 连接；

(3)设 P 表示一个指称对象域上的计算，用 $S[[P]]$表示 P 的语义，如果 $S[[P]]$可以通过计算语义泛函 $F[[P]]$的最小不动点得到，即 $S[[P]]=lfp\,F[[P]]$，若$\langle\alpha,\gamma\rangle$是一个 Galois 连接，定义语义泛函 $F'[[P]]=\alpha\circ F[[P]]\circ\gamma$，则 $F[[P]]$的抽象 $\alpha(F[[P]])=lfp\,F[[P]]$。

Galois 连接的上述性质表明，对 P 的语义可以连续地进行抽象，直到抽象对象域上的计算满足计算复杂度的要求为止。在抽象解释理论框架中，可使用 Widening 算子给出 $lfp\,F[[P]]$ 的一个上界逼近，Narrowing 算子则给出 $lfp\,F[[P]]$更精细的一个上界逼近。

定义 3（Widening 算子） 设$\langle L,\subseteq,\bigcup,\bigcap,\downarrow,\uparrow\rangle$是一个完备格，算子 $\nabla:L\times L\to L$ 称为 Widening 算子，当且仅当

(1) ∇是一个上界算子，即 $\forall l_1,l_2\in L,l_1,l_2\in l_1\nabla l_2$；

(2)对于任意的递增链$(l_n)_n$，递增链$(l_n^{\nabla})_n$ 都收敛，其中(l_n^{∇})定义为：如果 $n=0$，则$(l_n^{\nabla})=l_n$，否则 $l_n^{\nabla}=(l_{n-1}^{\nabla})\nabla l_n$。

偏序集合可以扩张成为完备格，然后可以为完备格定义 Widening 算子。

定义 4（Narrowing 算子） 设$\langle L,\subseteq,\bigcup,\bigcap,\downarrow,\uparrow\rangle$是一个完备格，算子 $\Delta:L\times L\to L$ 称为 Narrowing 算子，当且仅当

(1) $\forall l_1,l_2\in L$，若 $l_2\subseteq l_1$，则 $l_2\subseteq(l_2\Delta l_1)\subseteq l_1$；

(2) 对于所有的递降链$(l_n)_n$，递降链$(l_n^{\Delta})_n$ 收敛；其中(l_n^{Δ})定义为：如果 $n=0$，则$(l_n^{\Delta})=l_n$；否则 $l_n^{\Delta}=(l_{n-1}^{\Delta})\Delta l_n$。

3 谓词抽象

尽管关于不动点语义的抽象解释可以依靠完备格上的 Widening 算子和 Narrowing 算子为其提供正确性保障，但对于具体问题，基于 Galois 连接的抽象解释理论框架给出的 Widening 算子和 Narrowing 算子的构造方法过于复杂，难以自动化实现，限制了抽象解释的应用与推广。

1997 年，Graf 和 Saidi 提出了谓词抽象方法。它是抽象解释的一种特殊形式，用以解决传统抽象方法难以自动化实现的问题[13]。谓词抽象以给定的一组谓词上的计算抽象程序指称对象域上的计算[14]。

对于一个状态迁移系统而言，给定一组谓词 ϕ，根据 ϕ 的可满足性能够定义一个状态集合 C 上的等价关系，而等价关系则决定了集合 C 的一个划分（等价类），每一个状态属于且仅属于一个等价类，可将一个等价类作为一个抽象状态。

给定一组谓词 $\phi_1,\phi_2,\cdots,\phi_n$，定义一组布尔变量 $B_1,B_2,\cdots,B_n$，B_i 与谓词 ϕ_i 对应，B_i 的两个状态 1 和 0 分别表示谓词 ϕ_i 的可满足性。状态变迁系统的原始状态用该状态满足的谓词公式 ϕ 表示，其中 ϕ 是 $\phi_1,\phi_2,\cdots,\phi_n$ 的合取范式。抽象模型状态集合 A 是变量 $B_1,B_2,\cdots,B_n$ 上的一组正交布尔表达式，抽象状态用公式 $\exp^A(B_1,B_2,\cdots,B_n)$表示，抽象模型的初始状态集合用 I_A 表示。那么，Galois 连接中的抽象算子 α 和具体算子 γ 的定义如下。

定义 5（抽象算子） $\alpha:C\to A$ 把原始状态变迁系统中的状态映射到抽象状态变迁系统中的状态：

$$\alpha(\phi)=\wedge\{\exp^A(B_1,B_2,\cdots,B_n)\mid\phi\Rightarrow\exp^A[\bar{\phi}\mid\bar{B}]\}$$

式中，$\bar{\phi}$ 表示向量$(\phi_1,\phi_2,\cdots,\phi_n)$，$B\bar{B}$ 表示向量$(B_1,B_2,\cdots,$

B_n),$[\overline{\phi}|\overline{B}]$表示一个置换,即表示将 $\overline{\phi}$ 中的每一个分量 ϕ_i 用对应的分量 B_i 替换。

定义 6(具体算子) $\gamma: A \to C$ 把抽象状态变迁系统中的状态映射到原始状态变迁系统中的状态:

$$\gamma(\exp^A(B_1, B_2, \cdots, B_n)) = \exp^A[\overline{\phi}|\overline{B}]$$

对于任意一个抽象状态 $\exp^A(B_1, B_2, \cdots, B_n)$,把其中的布尔变量 B_i 用相应的谓词 ϕ_i 替换,即可得到对应于该抽象状态的所有原始状态。

在实际应用中,由于难以求出所有满足 $\phi \Rightarrow \exp^A[\overline{\phi}|\overline{B}]$的公式,需要化简抽象算子。一般常用 α 的一个上界逼近 $\alpha'(\phi) = \wedge\{B_i | \phi \Rightarrow \phi_i, 1 \leqslant i \leqslant n\}$代替 α。由原始状态求抽象状态的方法是:将每一个抽象状态用原子公式 false 或者合取范式 $c_1 \wedge c_2 \wedge \cdots \wedge c_n$ 表示,其中,$c_i (1 \leqslant i \leqslant n)$的取值为 B_i(当前状态满足谓词 ϕ_i),$\neg B_i$(当前状态满足谓词$\neg B_i$),或 true(当前状态既不满足谓词 ϕ_i,也不满足$\neg\phi_i$)。

抽象状态之间的迁移关系由原始状态之间的迁移关系决定。设原始状态 ϕ 的抽象状态为 $\exp^A(B_1, B_2, \cdots, B_n)$,从 ϕ 出发的原始状态迁移关系为 $\tau_j, 0 \leqslant j \leqslant |R_\phi|$,$R_\phi$ 是所有从 ϕ 出发的原始状态迁移关系集合 R_C 的子集,则从抽象状态 $\exp^A(B_1, B_2, \cdots, B_n)$出发的与 τ_j 对应的抽象状态迁移关系 τ_j^A 可由以下公式计算得出:

$$\tau_j^A(\exp^A(B_1, B_2, \cdots, B_n)) =$$

$$\begin{cases} \text{false}, & \text{if } \exp^A[\overline{\phi}|\overline{B}] \Rightarrow \neg g_j \\ \wedge_{i=1}^{n} \begin{cases} B_i, & \text{if post}[\tau_j](\exp^A[\overline{\phi}|\overline{B}]) \Rightarrow \phi_i \\ \neg B_i, & \text{if post}[\tau_j](\exp^A[\overline{\phi}|\overline{B}]) \Rightarrow \neg\phi_i \\ \text{true}, & \text{otherwise} \end{cases} \end{cases}$$

式中,g_j 是 τ_j 的发生条件,$post[\tau_j](\exp^A[\overline{\phi}|\overline{B}])$表示在迁移关系 τ_j 下当前状态($\exp^A[\overline{\phi}|\overline{B}]$)的后继状态。若 $post[\tau_j](\exp^A[\overline{\phi}|\overline{B}]) \Rightarrow \phi_i$,则 c_i 的取值为 B_i;若 $post[\tau_j](\exp^A[\overline{\phi}|\overline{B}]) \Rightarrow \neg\phi_i$,则 c_i 的取值为$\neg B_i$,否则 c_i 的取值为 true。

事实上,由于难以求出所有满足 $\phi \Rightarrow \exp^A[\overline{\phi}|\overline{B}]$的公式,因此这种算法在本质上是精化算子 α(对应于 widening 算子)的一个上界近似[12],以牺牲精确性为代价换取了易计算性,但需验证的性质仍然得到了保留。例如,一个无穷状态变迁系统[15]:

Initially:

$a=0$

Rules:

$0 \leqslant a \leqslant 10 \to a := a+1$;

true $a := 2a$;

$a \geqslant 2 \to a := a-2$;

若在此系统上验证 $a \neq 151$,因为原始系统是一个无穷系统,所以不可能采用穷举的方式加以证明。我们在此系统中定义谓词:

$$\phi_1 \equiv a(\text{mod } 2) = 0; \phi_2 \equiv 0 \leqslant a \leqslant 10$$

则其抽象系统可以描述为仅包含 3 个状态[0,1],[1,1],[1,0]的系统,见图 1。

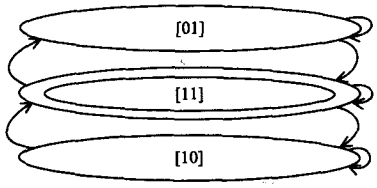

图 1 抽象系统

其中一个给定的状态,如[0,1],其第 1 分量 0 表示当前状态满足谓词$\neg\phi_1$,第 2 分量 1 表示当前状态满足谓词 ϕ_2。可以看出系统中没有[0,0]这个状态,是因为在原始系统的任何一个状态下,ϕ_1 和 ϕ_2 至少满足其一,即 $a(\text{mod } 2)=0$ 和 $0 \leqslant a \leqslant 10$至少满足其一。所以 a 若不是偶数,则必有 $0 \leqslant a \leqslant 10$,$a \neq 151$ 得以验证。

4 验证空间的划分

通过抽象技术得到的抽象模型有时在其验证空间上仍然过于庞大,因此考虑将待验证属性划分为若干个易于处理的子问题,然后通过验证每一个或其中的一些子问题而得到关于整个属性的验证。

假设已得到抽象模型 M,待验证的性质是 ϕ(一般地,ϕ 是一组由模型 M 中的命题所组成的公式),若要验证 ϕ 在 M 中成立,即 $M \models \forall\phi$,可以对 M 的搜索空间进行划分,这样公式 ϕ 就可以在每一个子搜索空间中进行验证。这种方法的特征是在验证 $M \models \forall\phi$ 时,只有与子空间相关的模型 M 中的迹(状态机模型 M 从初始状态开始的一组无穷状态变迁序列)被搜索,而那些与子空间无关的迹则被去掉。所以采用空间划分的方式对某一性质进行验证,最直接的效果是可以让验证以模块化的方式递进地进行,单次验证的空间规模在可接受的范围内。明显地,采用验证空间划分的方法所面临的主要问题是:要能够保证针对子空间的验证覆盖了待检验模型在其搜索空间上的所有可能情景。

因为抽象模型和原始模型都从 Kripke 结构的角度上观察,在形式上并无实际差别,所以可直接引入以下定理。关于这个定理的证明可以参阅文献[16]。

定理 1 设公式 $\phi_1, \phi_2, \cdots, \phi_n$ 满足 $M \models \forall(\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n)$,则 $M \models \forall\phi$,当且仅当 $M \models \forall(\phi_i \to \phi)$对所有的 $i \in \{1, 2, \cdots, n\}$成立。

若 ϕ 是一个公式,记$[[\phi]]$表示 M 中所有满足 ϕ 的迹的集合。如果 $M \models \forall(\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n)$,则称$\{[[\phi_1]], [[\phi_2]], \cdots, [[\phi_n]]\}$是一个 M 划分。

定理 1 提供了一种对验证空间进行划分的可行方法。

5 访问控制策略安全性的模型检测

这里主要讨论如何使用模型检测的方法验证访问控制策略的安全性。其基本思想是先将访问控制策略描述成一个状态变迁系统 TS,然后将待检验的安全性质用 LTL(Linear Temporal Logic)形式化地表示成公式 ϕ,最后通过模型检测工具,如 NuSMV 自动化地验证 TS 是否满足 ϕ。若 TS 不满足 ϕ,则给出一个反例。反例是 TS 的一组状态变迁序列 s_0,$s_1, s_2, \cdots$,可以验证 TS 按这样的序列迁移必然不满足安全性质 ϕ。

定义 7 状态迁移系统 $TS := (S, ACT, \to, I, AP, L)$,其中:

- S 是状态的集合;
- ACT 是操作的集合;
- $\to \subseteq S \times ACT \times S$ 是状态之间的迁移关系;
- $I \subseteq S$ 是初始状态的集合;
- AP 是原子命题的集合(AP 用于形式化地对状态 $\forall s \in S$ 进行描述);

• $L: S \to 2^{AP}$ 是标签函数(建立状态与 AP 之间的关系)。

定义 8 设原子命题的集合为 AP,LTL 公式的语法定义如下:

• 若 $\phi \in AP$,则 ϕ 是 LTL 公式;

• 若 ϕ 和 ϕ 是 LTL 公式,则 $\neg\phi, \phi \wedge \varphi, \phi \vee \varphi, \phi \to \varphi, G\phi, F\phi, X\phi, \phi U\varphi, \phi R\varphi$ 等是 LTL 公式。

X, G, F, U, R 分别表示下一个(neXt)、所有未来状态(Globally)、某未来状态(Future)、直到…全(Until)、直到…有(Release)等模态词;$\neg, \wedge, \vee, \to$ 等符号是连接词。

应指出,不同的 LTL 公式使用的标记符号和二元模态词略有差别,但它们的表达能力是一致的。LTL 公式适用于表达状态变迁系统 TS 上的由状态变迁顺序所定义的性质。

在验证 $TS \vdash \phi$ 时,若发现 $TS \not\models \phi$,则需要给出一个反例,用于分析出现系统不满足性质 ϕ 的原因。所以一般并不直接验证 $TS \models \phi$,而是采用验证 $TS \models \neg\phi$ 的方法。因为一旦找到 TS 中的一条路径 p 满足 $\neg\phi$,那么 p 就是 $TS \models \phi$ 的一个反例。

6 实例

本文以可信系统中的隐通道问题为实例进行分析。隐通道是指可信系统中的高安全级用户通过违反系统安全策略的方式向系统中具有较低或不可比安全级的用户传送信息的一种机制[18]。因为隐通道利用了系统原本不是用于数据传送的资源来传送数据,所以这种通信方式一般不能被系统的固有安全机制所检测和控制。例如:①一个高安全级用户,通过对某个文件的加、解锁来控制低安全级用户对该文件的写入是否成功,并以此向低安全级的用户传递信息;②一个高安全级进程通过选择自己的 CPU 占用状态来影响 CPU 对低安全级进程的响应,并以此向低安全级进程传递信息,这些都是典型的隐通道。隐通道分析是高等级可信评估的重要指标,在 TCSEC,CC 和我国的 GB17859－1999 等标准中均有相关要求[19]。

根据隐通道的工作原理,可知道基于模型检测的方法搜索系统中的隐通道主要是通过检测在每一个共享变量 v_i 上的操作序列$\{op_i\}$是否满足隐通道存在的条件。这个条件可以简单地概括为:一个高安全级的进程使用某种操作更改了共享变量 v 的某一个属性在先,而另一个具有较低安全级的进程可借助某种操作观察到这种属性的改变在后[20]。

```
Process (v0, v1, v2, v3, v4, v5)
{…
do switch
  Label 1: Write_Variable (vi); …
  Label 2: Read_Variable (vi); …
  Label 3: Lock_Variable (vi); …
  Label 4: Unlock_Variable (vi); …
  Label 5: Open_Variable (vi); …
  Label 6: Close_Variable (vi); …
  Label 7: Varible_Locked (vi); …
  Label 8: Variable_Opened (vi); …
  od
…}
```

图 2 待分析程序片断

以一个含有 6 个可被不同安全级操作共享的变量$\{v_0, v_1, \cdots, v_5\}$和 8 种针对共享变量的操作{*Write_Variable*, *Read_Variable*, *Lock_Variable*, *Unlock_Variable*, *Open_Variable*, *Close_Variable*, *Varible_Locked*, *Variable_Opened*}的程序片断作为分析对象,采用模型检测的方法进行隐通道分析。这里的共享变量具有广义性,即可以是普通的内存变量,也可以是文件或 I/O 端口,见图 2。

为了便于分析,在状态迁移系统 $TS: = (S, ACT, \to, I, AP, L)$ 的建模过程中,将状态 S 定义为$\{v_i, op_0, op_1, \cdots, op_7\}$,即共享变量与其上操作的组合。那么原始状态机的状态数为 241,920,而状态之间的变迁数量则为 58,525,044,480 种。根据第 2 节中的讨论,在搜索隐通道的过程中,若一个具有较高安全级的操作作用在变量 v_i 上,那么其后具有较低安全级的操作只有作用在 v_i 上才可能构成隐通道。即在搜索隐通道的过程中,操作序列的第一个操作 op_0 选定,那么其后的操作 $op_i (1 \leqslant i \leqslant \infty)$的选择具有一定的确定性,即 $op_i (1 \leqslant i \leqslant \infty)$必须作用在与 op_0 作用的相同变量上才可能形成隐通道,因此可据此使用定理 1 对检测空间进行划分。即以$\{v_0, v_1, \cdots, v_5\}$为参考点将原始状态机划分成 6 块,尽管 6 块所包含的状态数并未减少,但若针对每一块单独进行模型检测,则状态变迁数却仅为原始状态机上状态变迁数的16.67%。

根据隐通道的工作原理,我们知道高安全级操作 op_i 的发送行为是指 op_i 更改了共享变量 v 的某一个属性值;而具有较低安全级的操作 op_i 的接收行为是指 op_i 作用于共享变量 v 时可以觉察到其某一个属性值已被改变。所以在操作系统中常用的这 8 种操作{*Write_Variable*, *Read_Variable*, *Lock_Variable*, *Unlock_Variable*, *Open_Variable*, *Close_Variable*, *Varible_Locked*, *Variable_Opened*},可以通过选取适当的谓词抽象成两类:一类具有更改变量属性的功能,这里简称为 *Modi*,如 *Write_Variable*, *Lock_Variable*, *Unlock_Variable*, *Open_Variable*, *Close_Variable* 等;另一类具有感知变量属性改变的功能,这里简称为 *Read*,如 *Read_Variable*, *Varible_Locked*, *Variable_Opened* 等。需要指出的是,有些操作兼具两种功能,如 *Write_Variable* 既具有改变变量值的功能,也具有感知功能。若 *Write_Variable* 试图写入某一变量而不成功,则它可推断该变量已被其它操作上锁 *Lock_Variable*。

所以在检测隐通道时,上述 8 种操作可以抽象成 *Modi* 和 *Read* 两类操作。这样根据本文第 2 节中的讨论,状态$\{v_i, op_0, op_1, \cdots, op_7\}$使用谓词"*Modi*"和"*Read*"划分后得到的抽象状态为$\{v_i, B_1, B_2\}$,则原来的全部 241,920 种状态可以划分为 12 个等价类。将每一个等价类作为一个状态进行抽象,在空间划分后的状态机上,状态变迁数仅有 12 种,见表 1。

表 1 状态数、状态变迁数对比

	状态数	状态变迁数
原始状态机	241,920	58,525,044,480
空间划分后的状态机	241,920	9,753,972,480
空间划分且抽象的状态机	12	12

本文采用模型检测工具 NuSMV 在原始状态空间中进行隐通道的搜索工作。整个工作由 4 个模块组成:高安全级操作模块 *High_Op*,低安全级操作模块 *Low_Op*,以及两个进

行信息传递的通道模块 *H2Lchan*,*L2Hchan*。由于隐通道中 *High_Op* 和 *Low_Op* 两模块是通过共享变量 v_i 属性的改变而进行信息传递的,故两个通道模块构建在共享变量 v_i 之上。

High_Op 发送消息,每条消息包含数据 *data* 和控制位 *ctrbit* 两个部分。整型数据 *data* 表示高安全级操作 op_i 对共享变量 v 属性值的更改,一个数据代表一次更改。*Low_Op* 收到了带有控制位的数据包后(即觉察到共享变量某一属性值的更改)会通过通道 *L2Hchan* 返回给 *High_Op* 一个确认消息 *ack*(*ack* 为真)。若没有收到数据包,则返回一个失败消息 *ack*(*ack* 为假)。*High_Op* 根据返回的不同消息而做出相应的动作:(1)若消息表明 *Low_Op* 模块成功收到数据包,则 *High_Op* 模块继续发送下一个数据包,即对共享变量属性再次更改;(2)若 *Low_Op* 模块返回的消息是没收到数据包,无法感知共享变量属性的更改,则 *High_Op* 模块再重复发送上条消息,直到 *Low_Op* 模块收到。程序中变量 *ctrbit* 和 *ack* 都是为建模方便而添加的辅助变量。

待验证的性质用 LTL 公式描述为:

LTLSPEC G F(*High_Op. state* = *sent* −> *F* (*Low_Op. state*=*received*))。即一个高安全级的操作 op_i 对共享变量某一个属性的修改(发送信息),其后总可以被一个具有较低安全级的操作 op_i 所感知(接收信息)。

实验环境:硬件 Intel(R)CPU CORE 2 DUO(TM)E7500 双核,主频为 2.93GHz,内存 1.96G。操作系统为 Microsoft Windows XP professional。程序运行结果见表 2。

表 2 模型检测的实践与内存消耗

	时间消耗	内存消耗
原始状态机	约 4800 小时	66,464k
空间划分后的状态机	约 140 时	55,456k
空间划分且抽象的状态机	0.09 秒	48,876k

由于机器配置的限制,原始状态机模型的状态变迁数超出了计算机的处理能力,我们通过计算每天可以检测的状态变迁数进行推算。在此机器配置下,检测原始状态机模型约需要 4800h。模型经空间划分后,虽然状态数维持不变,但状态变迁数量被有效约减,模型检测的时间消耗缩减为大约 140h。最后再经过抽象,只有 12 个状态,只需 0.09s 即可完成。

从这个实例中可以观察到,尽管原始状态机模型相当复杂,但针对具体的隐通道问题进行验证空间划分和抽象后,得到的待检测模型中只含有与隐通道相关的信息,模型检测的规模得到了极大程度的约减。需要指出的是,搜索空间划分和抽象后的待检测模型的规模与具体需要检验的安全性问题本身有关,即针对不同的安全性问题得到的抽象模型规模也会有所差别。

结束语 针对传统访问控制策略安全性分析方法在速度上和效率上的不足,在模型检测技术的支持下,提出了一种将抽象技术和验证空间划分相结合的访问控制策略安全性分析方法。抽象和验证空间的划分在顺序上没有约定,可以根据具体问题选择是抽象在前,还是验证空间划分在前。隐通道是访问控制策略安全性分析中的一个重要问题,以此为例,检验了将抽象技术和验证空间划分相结合进行访问控制策略安全性分析的方法的有效性和可行性。

参 考 文 献

[1] 曾红卫,缪淮扣. 构件组合的抽象精化检验[J]. 软件学报,2008,19(5):1171-1181

[2] 沈昌祥,张焕国,冯登国,等. 信息安全综述[J]. 中国科学 E 辑,2007,37(2):129-150

[3] 蔡嘉勇,卿斯汉,刘伟,等. 基于规则推导的特权隐式授权分析[J]. 软件学报,2008,19(8):2102-2113

[4] Reith M, Niu Jian-wei, Winborough W. Apply Model Checking to Security Analysis in Trust Management[C]// IEEE 23rd International Conference on Data Engineering. 2007:734-743

[5] 屈婉霞,李暾,郭阳,等. 谓词抽象技术研究[J]. 软件学报,2008,19(1):11-22

[6] Sandhu R S. The Typed Access Matrix Model[C]// IEEE Symposium on Research in Security and Privacy. 1992:122-136

[7] Bertino E, Kan L R, Sandhu R, et al. Secure Knowledge Management: Confidentiality, Trust, and Privacy[J]. IEEE Transaction on Systems, Man, and Cybernetics, Part A, 2006, 36(3):429-438

[8] Park J, Sandhu R. The $UCON_{ABC}$ Usage Control Model [J]. ACM Transactions on Information and System Security, 2004, 7(1):128-174

[9] Zhang Xin-wen, Parisi-Presicce F, Sandhu R, et al. Formal Model and Policy Specification of Usage Control[J]. ACM Transactions on Information and System Security, 2005, 8(4):351-387

[10] Wang D. SAT-based abstraction refinement for hardware verification[D]. Pittsburgh: Carnegie Mellon University, 2003

[11] Wu Q, Hsiao M S. A new simulation-based property checking algorithm based on partitioned alternative Search Space Traversal[J]. IEEE Transactions on Computers, 2006, 55(11): 1325-1334

[12] 李梦君,李舟军,陈火旺. 基于抽象解释理论的程序验证技术[J]. 软件学报,2008,19(1):17-26

[13] Graf S, Saidi H. Construction of abstract state graphs with PVS [C]// LNCS 1524. Haifa, 1997:72-83

[14] 李暾,屈婉霞,郭阳,等. 基于符号模拟和约束逻辑编程的 RTL 级 Verilog 谓词抽象方法[J]. 计算机学报,2007,30(7):1138-1144

[15] Das S. Predicate Abstraction [D]. Stanford: Stanford University, 2003

[16] 普飞,张文辉. 结合搜索空间划分和抽象进行 LTL 模型检测[J]. 中国科学 E 辑,2007,37(12):1504-1520

[17] Baier C, Katoen J-P. Principles of Model Checking[M]. Massachusetts: The MIT Press, 2008

[18] McHugh J. Covert channel analysis: a chapter of the handbook for the computer security certification of trusted systems[R]. Portland: Portland State University, 1995

[19] 王昌达,鞠时光,周从华,等. 一种隐通道威胁审计的度量方法[J]. 计算机学报,2009,32(4):751-762

[20] 王昌达,鞠时光,宋香梅. 一种动态的隐通道消除算法[J]. 小型微型计算机系统,2009,30(2):236-241