软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(6):1534−1556 [doi: 10.13328/j.cnki.jos.004626]

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

基于开放逻辑 R 反驳计算的访问控制策略精化*

吴迎红, 黄 皓, 吕庆伟, 曾庆凯, 张迪明

(计算机软件新技术国家重点实验室(南京大学),江苏 南京 210046)

通讯作者: 黄皓, E-mail: hhuang@nju.edu.cn

摘 要: 策略精化是解决分布式应用访问控制策略配置复杂性的重要方法,现有精化技术给出了策略分层描述和逐层精化的方法,但处理策略之间关联问题的能力不足.基于精化树描述策略和策略关联,基于叶结点策略冲突判断,采用开放逻辑 R 反驳计算分析精化树策略关联属性,能够消解策略冲突同时保证策略互斥、组合、访问路径协同、精化映射等关联正确,并能够按序消解不同类型策略冲突、自由取舍相冲突的策略.实验与分析计算性能表明,该方法符合 SaaS 平台客户应用系统策略精化需求.

关键词: 访问控制;SaaS;策略精化;策略冲突分析;开放逻辑;R 反驳计算

中图法分类号: TP309

中文引用格式: 吴迎红,黄皓,吕庆伟,曾庆凯,张迪明.基于开放逻辑 R 反驳计算的访问控制策略精化.软件学报,2015,26(6):1534−1556. http://www.jos.org.cn/1000-9825/4626.htm

英文引用格式: Wu YH, Huang H, Lü QW, Zeng QK, Zhang DM. Access control policy refinement technology based on open logic R-refutation calculus. Ruan Jian Xue Bao/Journal of Software, 2015,26(6):1534−1556 (in Chinese). http://www.jos.org.cn/1000-9825/4626.htm

Access Control Policy Refinement Technology Based on Open Logic R-Refutation Calculus

WU Ying-Hong, HUANG Hao, LÜ Qing-Wei, ZENG Qing-Kai, ZHANG Di-Ming

(National Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210046, China)

Abstract: Policy refinement is an important technology to resolve the configuration complexity of access control policies in distributed applications. Existing methods for policy refinement describe and refine policies layer by layer. However, they are weak in dealing with the relationship between policies. In this study, policies and the relationship between them are described based on the policy refinement tree where policies conflict analysis is performed on the leaf nodes to allow using R-refutation calculus of open logic to analyze refinement policy correlation properties. This method can resolve conflicting policies while correctly maintaining mutual exclusion, combination, access path coordination, and refinement mapping of policies. It can also resolve conflicting policies of different types in order, and freely make a choice among conflicting policies. Experiments and performance analysis demonstrate that the presented method meets the need of dynamic adaption of policy refinement for service-oriented application systems on SaaS platform.

Key words: access control; SaaS; policy refinement; policy conflict analysis; open logic; R-refutation calculus

访问控制是实现保密性、完整性和可用性的主要手段.SaaS 通过互联网提供完整的、可以立即使用的应用.弹性计算和动态自适应能力是它的重要特征.通过逐层细化软件系统抽象模型构建实际应用系统的模型驱动架构(model driven architecture,简称 MDA)方法[1]是 SaaS 平台应用系统构建的重要方法.策略是面向服务应用安全管理的关键[2].以系统分层模型对象为策略对象,通过策略映射规则,将上层抽象的访问控制策略转化为下层具体的访问控制策略的过程称为策略精化[3−5].策略精化与模型驱动构建应用、减少系统开发复杂性的思想一致,是解决分布式网络应用系统访问控制策略配置复杂性的重要技术.SaaS 需要服务品质协议

* 基金项目: 国家高技术研究发展计划(863)(2011AA01A202)
收稿时间: 2013-03-21; 修改时间: 2014-01-10; 定稿时间: 2014-05-09

(service-level agreement,简称 SLA)的呈现[6],访问控制抽象策略到配置策略之间的精化关系也是访问控制 SLA 的呈现.

我们用谓词 $p(s,o,a)$表示一个访问控制策略,表示授予主体 s 对客体 o 访问时必须具有的权限或满足条件 a.用 $p(s,o,-a)$表示禁止主体 s 对客体 o 拥有权限 a,假设两个策略 $p_1=p(s_1,o_1,\alpha)$或 $p_2=p(s_2,o_2,\beta)$,$s_1=s_2$,$o_1=o_2$,出现 $\alpha=a$,$\beta=-a$,或者 p_1,p_2 不能同时成立的情况,则 p_1,p_2 冲突.策略精化通过精化推导将抽象策略逐层转化为实际系统配置策略,形成符合完备性的支撑策略.通过策略冲突分析保证精化一致性.

策略精化需要满足的策略关联属性有:

- 完备性,即,所有上层策略在下层得到支撑;
- 一致性,即,同层策略之间、下层策略与上层策略没有冲突;
- 分布式应用一个主体请求访问一个客体往往需要经过访问路径一个或多个控制单元进行权限检查,控制访问过程中的不同侧面,是否能够推导并协同下层系统访问路径一系列控制单元的访问控制策略,是衡量精化能力的重要方面;
- 分布式应用系统有时还根据应用需要规定一组组合在一起不可分割的策略或互斥的策略.

策略冲突分析不仅需要分析策略冲突,还需要分析因冲突引起的策略之间关联属性是否正确并加以修正.策略冲突分析的有序分析不同类型策略冲突能力、消解规则可选范围以及修正策略关联属性能力,决定有效精化策略范围.

基于模板[7]与基于 RBAC 对象结构[8]的精化技术不具备描述和推导下层系统访问路径一系列相关策略的能力.由于网络及其安全监控技术的多样性、变化性、复杂性,难以静态模式化描述,因此也难以发展这两种方法实现分布式访问控制纵深防御.Abadi 等人[9,10]研究由抽象访问控制策略精化推导实际系统访问路径一系列相关策略.Davy 等人[11]、Lück 等人[12,13]在文献[9,10]研究基础上完善系统和访问控制策略分层描述,设计基于系统和策略的分层模型及层间映射规则精化策略,在技术实现层,推导路径协同策略,但是存在以下不足:

(1) 不能描述和处理策略之间组合和互斥关系;
(2) Lück 等人设计方法[12,13]的冲突消解为保证精化完备性和一致性,只能由上层策略取代下层策略,这在 IDS 等根据具体事件和状态实时决定应用策略的情况下是不适用的;
(3) 计算属性描述层次经常存在多种访问路径(可以有 FTP 或 Web 等不同访问路径),现有技术不能计算多条访问路径的精化策略;
(4) 在技术实现层也经常存在多条访问路径,现有技术不能处理各层次多条访问路径策略的协同一致;
(5) 不能处理像 IPSec 策略这样需要有序分析不同类型冲突[14]策略;
(6) 不能提供访问控制 SLA 的呈现.

这些不足的根本原因在于这些技术缺少策略之间的关联描述以及基于策略关联描述、由冲突策略消解、反向推导修正同层或上层策略关联属性的能力.本文设计基于开放逻辑 R 反驳计算的策略精化方法,解决现有技术上述不足指出,提高策略精化能力.

本文第 1 节介绍策略与策略关联属性的精化树描述.第 2 节介绍基于精化计算基于开放逻辑 R 反驳规则消解策略冲突的方法.第 3 节模拟分析并测试该方法的计算性能.第 4 节是本文结论.

1 策略与策略精化形式描述及精化树构建

本文策略描述基于 OMG 组织公布的 MDA 三层系统模型:计算无关模型(computational independent model,简称 CIM)、平台无关模型(platform independent model,简称 PIM)、平台相关模型(platform specific model,简称 PSM).如图 1 所示一个常见的企业信息服务系统,信息服务系统包括内部信息、邮件服务、销售服务、售后服务和财务服务.带箭头虚线表示系统对象在上下层之间的映射关系,如 CIM 层内部信息.服务在 PIM 层对应系统 Internet 认证和具有 Web 和 FTP 接口的技术资料,PIM 层系统 Internet 认证在 PSM 层对应系统 Web 认证和 WLAN 认证,PIM 层技术资料对应 PSM 层 Windows 技术服务器 1 和 Windows 技术服务器 2 上存放的技术文

件.内部员工通过交换机 1 或无线 WLAN 基站访问信息服务系统.本文策略精化计算原理基于文献[9−13].

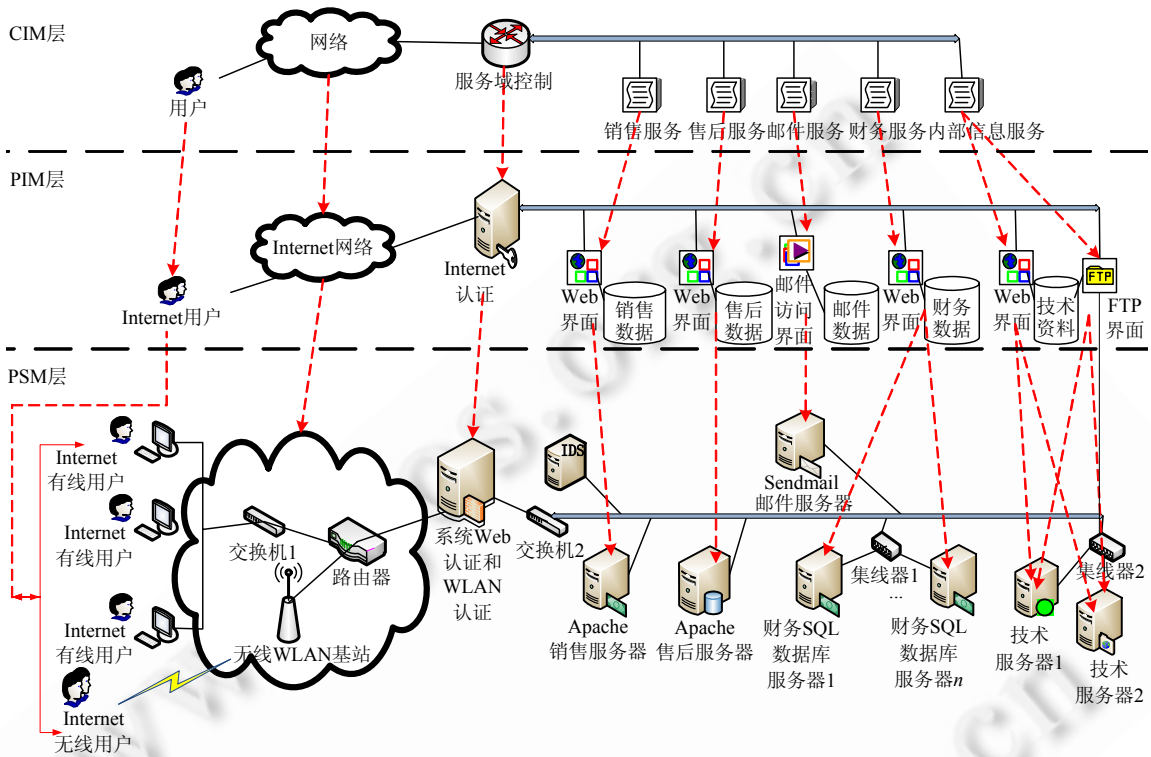

Fig.1 Three layers MDA structure sample of distributed application system

图 1 分布式应用系统 MDA 三层结构示例

1.1 原子策略与原子策略集合

根据 MDA 分层和策略精化研究,CIM,PIM,PSM 层策略主体、客体和权限与约束定义如下:

(1) CIM 层

- $S=\{s_1,...,s_i,...\}$,其中,s_i 为个体用户或任务;
- $O=\{o_1,...,o_i,...\}$,其中,o_i 为资源个体实体(如文件、程序等);
- $A1=\{$读,写,添加,执行,创建,通过认证,...$\}$.

(2) PIM 层

- $S=\{s_1,...,s_i,...\}$,s_i=(个体用户或功能模块,访问类型等属性),s_i 是上层个体用户或服务具体化为特定访问方式的用户或功能模块,比如(用户 A,Web),(用户 B,FTP);
- $O=\{o_1,...,o_i,...\}$,其中,o_i 为资源个体实体(如文件、程序等);
- $A2=A1\cup\{$记录,备份,告警,...$\}$,$A2$ 增加实现目标访问控制计算相关的主体到客体访问方式的认证、审计权限.

(3) PSM 层

- $S=\{s_1,...,s_i,...\}$,s_i=(个体用户或功能模块,访问类型等属性,IP 地址,端口),描述具体的行为主体及其平台相关的属性;
- $O=\{o_1,...,o_i,...\}$,o_i=(文件或程序,IP 地址(有些程序还需要端口号,协议号)),描述了程序、文件等客体及其平台相关的属性;

- $A3=A2\cup\{$入,出,IPSec AH,IPSec ESP,...$\}$,$A3$ 增加实现目标访问控制的主体到客体访问路径的通信以及通信机密性、完整性权限.

谓词 $p(s,o,a)$表示授予主体 s 对客体 o 访问时必须具有的被授予的权限 a 或者 s 对客体 o 访问必须满足的条件 a.$S=\{s_1,\ldots,s_i,\ldots\}$为主体集合,$O=\{o_1,\ldots,o_i,\ldots\}$为客体集合,$A=\{a_1,\ldots,a_i,\ldots\}$为分别权限集合.我们把 a 称为权限,但是这里的 a 比通常意义的权限意义更广,它表示 s 访问 o 必须被授予的权限 a 或必须满足的条件 a.用 $p(s,o,-a)$表示禁止主体 s 对客体 o 拥有权限 a,$p(s,o,a)$和 $p(s,o,-a)$都称为原子策略.在实际应用中,策略往往不是以一个主体对一个客体权限形式描述,而是描述一组主体对一组客体权限,我们用 $p(S,O,a)$来表示,称为原子策略集合:

$$p(S,O,a)=\forall(s_i,o_i)\in SO\ p(s_i,o_i,a) \tag{1}$$

其中,$S=\bigcup_{i=1}^{n}\{s_i\}$,$O=\bigcup_{i=1}^{n}\{o_i\}$,$SO=\{(s_i,o_i)|i\in N\}$,$SO$ 是 $S\times O$ 的子集,以精化树的一个结点记录策略 $p(S,O,a)$.

1.2 组合策略

有两种类型的组合策略,我们分别阐述:

1) 假设有 m 个权限(o_i,a_i),$i=1,2,\ldots,m$,有 n 个主体 s_j,$j=1,2,\ldots,n$.我们有一个 $n\times m$ 策略矩阵:

$$\begin{vmatrix} p(s_1,o_1,a_1) & \ldots & p(s_1,o_m,a_m) \\ \vdots & & \vdots \\ p(s_n,o_1,a_1) & \ldots & p(s_n,o_m,a_m) \end{vmatrix}.$$

对 n 个主体 s_j,$j=1,2,\ldots,n$ 而言,每个主体 s_j 在完成一个任务的时候,m 个策略 $p(s_j,o_i,a_i)$,$i=1,2,\ldots,m$ 都是必须的,要么全部赋予,要么全部撤销,所以,策略表示为 $N=\{1,2,\ldots,n\}$,$\forall j\in N\wedge_{i=1}^{m}p_{ij}$.

另一方面,从进一步精化计算的相关性出发,我们将关于对象 o_i 的策略 $p_i=\{p(s_j,o_i,a_i)|j=1,2,\ldots,n\}$作为一个策略集合处理,$i=1,2,\ldots,m$.策略 p_i 的原子策略全体构成策略矩阵的第 i 列 n 个策略,因此,我们将该组合策略表示成$\wedge\{p_1,\ldots,p_m\}$:

$$\wedge\{p_1,\ldots,p_m\}=\forall j\in N\wedge_{i=1}^{m}p_{ij} \tag{2}$$

在精化树中,我们将组合策略$\wedge\{p_1,\ldots,p_m\}=\forall j\in N\wedge_{i=1}^{m}p_{ij}$ 表示为父结点,而 $p_i=\forall j\in Np(s_j,o_i,a_i)$,$i=1,2,\ldots,m$ 分别作为子结点.父结点类型($Node\_type$)为“and”,原因是各个子结点中的主体同为 s_j 的原子策略必须同时授权或都不授权,即,原子策略关系为 $p(s_j,o_1,a_1)\wedge\ldots\wedge p(s_j,o_m,a_m)$;如果子结点 p_i 中某个原子策略 $p(s_j,o_i,a_i)$不能授权,父结点策略$\forall j\in N\wedge_{i=1}^{m}p_{ij}$ 中相应的策略 $\wedge_{i=1}^{m}p_{ij}$ 就要撤销;父子策略之间关系为

$$\forall j\in N\wedge_{i=1}^{m}p_{ij}\rightarrow\wedge_{i=1}^{m}p_{ij} \tag{3}$$

组合策略精化树表示如图 2 所示.

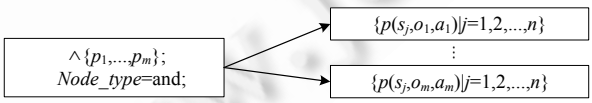

Fig.2 Refinement tree representation of the policy $\wedge\{p_1,\ldots,p_m\}$

图 2 策略$\wedge\{p_1,\ldots,p_m\}$的精化树表示

2) 有 m 个策略 $p_1=p(s_1,o_1,a_1),\ldots,p_m=p(s_m,o_m,a_m)$,它们必须同时授权或同时都不授权,我们同样用$\wedge\{p_1,\ldots,p_m\}$表示这样的策略,称为组合策略,也就是$\wedge\{p_1,\ldots,p_m\}=p(s_1,o_1,a_1)\wedge\ldots\wedge p(s_m,o_m,a_m)$.在精化树中,$\wedge\{p_1,\ldots,p_m\}$表示为父结点,父结点类型($Node\_type$)为“and”,$p(s_j,o_i,a_i)$为子结点,$i=1,2,\ldots,m$.

1.3 策略精化

1.3.1 CIM 层策略精化为 PIM 层策略

CIM 层策略描述主体对客体的访问权限,并没有确定主体用什么方式来访问客体,例如用 Web 的方式或

FTP 的方式等等.然而访问计算方式不同,涉及到的控制流程和数据对象就不同,策略也会不同.

CIM 层原子策略 $p(s_i,o_i,a)$向 PIM 层精化,PIM 层假设有 c_j 种访问方式,$N=\{1,2,...,n\}$,$j\in N$.s_i 对 o_i 的访问细化成了 PIM 层的 n 个的主体$(s_i,c_1),...,(s_i,c_n)$对 o_i 访问的集合,只要一种授予其访问的权限,s_i 就可以访问 o_i,所以 CIM 层策略对应 PIM 层 n 种访问方式策略的"或".CIM 层原子策略 $p(s_i,o_i,a)$权限 a,在 PIM 层每种访问方式 c_j 下可能涉及到多个(n_j)权限控制点,这些控制点上都授权才能实现该种方式的访问,即,CIM 层原子策略 $p(s_i,o_i,a)$在计算方式 c_j 下等价于 PIM 层的 $p((s_i,c_j),o_i,b_{j,1})\wedge...\wedge p((s_i,c_j),o_i,b_{j,n_j})$,表现为"与"关系,所以,CIM 层策略 $p(s_i,o_i,a)$精化表示成

$$p(s_i,o_i,a)\rightarrow \vee_{j=1}^{n} \wedge_{k=1}^{n_j} p((s_i,c_j),o_i,b_{j,k}).$$

对于 CIM 层的组合策略$\forall j\in N \wedge_{i=1}^{m} p_{ij}$,精化只对它的子结点进行,子结点策略形式只是 $p(S,O,a)=\forall(s_i,o_i)\in SOp(s_i,o_i,a)$的一种特例.我们构造 $SC_jO=\{(s_i,c_j),o_i\}|(s_i,o_i)\in SO\}$,$j=1,2,...,n$.策略 $p(S,O,a)$的精化表示成

$$p(S,O,a)\rightarrow \forall(s_i,o_i)\in SO \vee_{j=1}^{n} \wedge_{k=1}^{n_j} p((s_i,c_j),o_i,b_{j,k}) \tag{4}$$

以 n 个结点分别表示:$\forall(s_i,o_i)\in SO \wedge_{k=1}^{n_j} p((s_i,c_j),o_i,b_{j,k})$,$j=1,2,...,n$ 作为 $p(S,O,a)$策略结点的儿子,父结点 $p(S,O,a)$类型为"or",表示父结点策略 $p(s_i,o_i,a)$等价于儿子结点策略 $\wedge_{k=1}^{n_j} p((s_i,c_j),o_i,b_{j,k})$,$j=1,2,...,n$ 的"或".精化计算的精化树表示如图 3 所示.

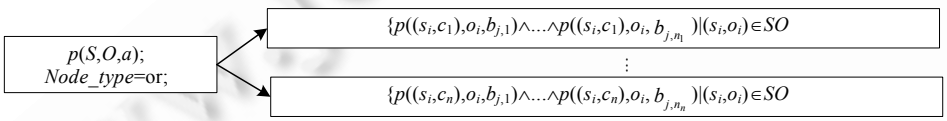

Fig.3 Refinement tree representation of CIM layer to PIM layer policy refinement

图 3 CIM 层到 PIM 层策略精化的精化树表示

1.3.2 PIM 层策略精化为 PSM 层策略

PIM 层策略描述主体以特定的计算方式访问客体的权限,但并没有确定主体或客体的物理位置.然而,主体或客体的物理位置决定主体访问客体不同的网络路径,不同的路径上会有不同的访问控制单元,需要不同的访问控制策略.

PIM 层的结点策略有两种情况:

- 一种是从 CIM 层的策略精化来的,可以表示成

$$\forall(s_i,o_i)\in SO \wedge_{k=1}^{n_j} p((s_i,c_j),o_i,a_{j,k}),j\in\{1,2,...,n\}.$$

- 另一种是 PIM 层首次定义的策略,如果是组合策略$\wedge\{p_1,...,p_m\}$,如前所述,它转化成了若干子结点策略等待进一步精化,策略形式可以统一表示为

$$p(S,O,a)=\forall(s_i,o_i)\in SOp((s_i,c_j),o_i,a),j\in\{1,2,...,n\}.$$

所以,PIM 层待精化策略可以表示成统一形式:

$$\forall(s_i,o_i)\in SO \wedge_{k=1}^{n_j} p((s_i,c_j),o_i,a_{j,k}),j\in\{1,2,...,n\}.$$

PIM 层策略主体(s_i,c_j)对客体 o_i 访问,在 PSM 与位置相关的因素主要包括 IP 地址和端口,我们用 ds=(IP 地址,端口),do=(IP 地址,端口)分别表示主体(s_i,c_j)与客体(o_i)位置相关的 IP 地址和端口,所以在 PSM 层分别用(s_i,c_j,ds_i)和(o_i,do_i)来表示主体和客体.PIM 层的集合 $SC_jO=\{((s_i,c_j),o_i)|(s_i,o_i)\in SO,j\in\{1,2,...,n\}\}$在 PSM 层转化成

$$SC_jPO=\{((s_i,c_j,ds_i),(o_i,do_i))|(s_i,o_i)\in SO,j\in\{1,2,...,n\}\}.$$

如果 PIM 层的一组策略$\forall(s_i,o_i)\in SO \wedge_{k=1}^{n_j} p((s_i,c_j),o_i,a_{j,k}),j\in\{1,2,...,n\}$在 PSM 层主体具有相同访问控制起始物理位置,并且客体也具有相同访问控制终止物理位置,那么对于给定的计算方式 c_j,这组主体和客体之间具有相同访问网络路径,并受到路径上一组相同控制点的控制.我们根据是否具有相同访问控制起点 ds_i 和终点 do_i,

将 SC_jPO 划分成 u 个子集: $SC_jPO=\bigcup_{t=1}^{u}SC_jPO_t$ .PIM 层的策略相应地划分成 PSM 层策略子集:

$$\forall((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k}),t=1,2,...,u.$$

- 以$\forall(s_i,o_i)\in SO\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k}),j\in\{1,2,...,n\}$为父结点;
- 以 u 个结点分别表示$\forall((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k}),t=1,2,...,u$ 作为子结点.

父结点类型为“set”,表示父结点策略集合等于子结点策略集合的并集.即:

$$j\in\{1,2,...,n\},\forall(s_i,o_i)\in SO\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k})\rightarrow\forall t\in\{1,...,u\}\forall((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k}) \quad (5)$$

策略基于物理位置划分(精化)的精化树表示如图 4 中长线虚线框中所示.

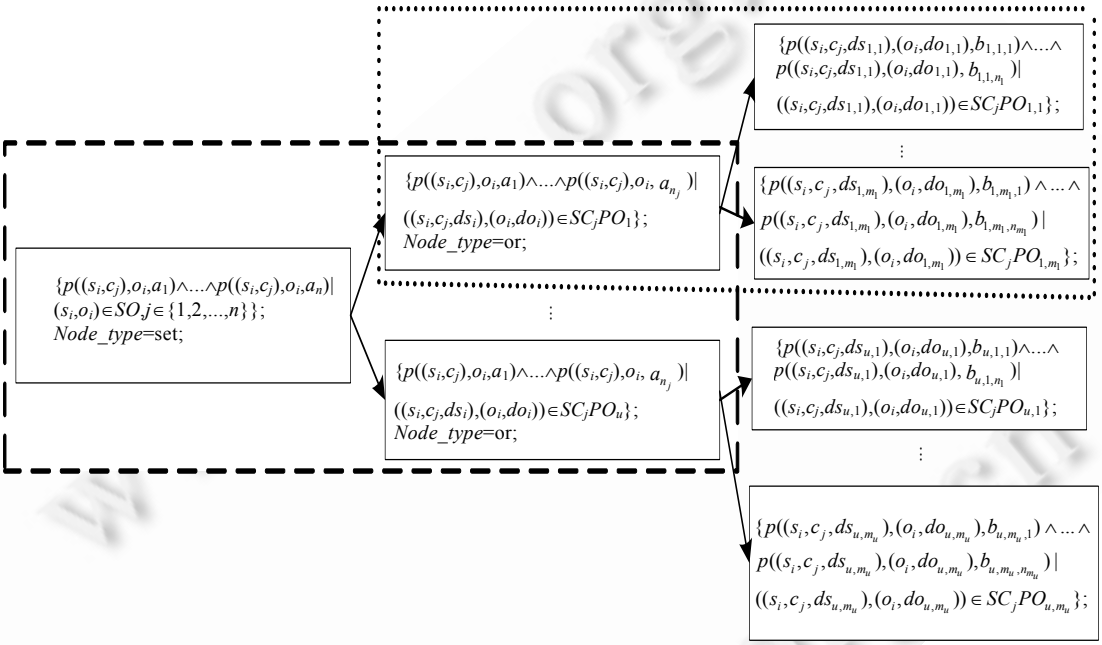

Fig.4 Refinement tree representation of PIM layer to PSM layer policy refinement

图 4 PIM 层到 PSM 层策略精化的精化树表示

每个 SC_jPO_t 中主客体具有相同网络访问路径,假设主客体之间存在的 m_t 条网络访问路径,主体(s_i,c_j,ds_i)对(o_i,do_i)的访问根据细化成了 m_t 个网络路径上主客体$((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}))$之间的访问,$z=1,2,...,m_t$,只要其中的一条路径允许主体$(s_i,c_j,ds_{t,z})$访问客体$(o_i,do_{t,z})$,它就可以实现主体$(s_i,c_j,ds_i)$对客体$(o_i,do_i)$的访问,所以,$m_t$ 条网络访问路径策略之间表示成“或”关系.

由 SC_jPO_t 构造 $SC_jPO_{t,z}$:对于 $z=1,...,m_t$,如果$((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t$,那么:

$$ds_{t,z}=ds_i,do_{t,z}=do_i,((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}))\in SC_jPO_{t,z}.$$

每个 $SC_jPO_{t,z}$ 中,主体访问客体 m_t 条网络访问路径中任意一条网络路径 z 上,PIM 层策略 $\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k})$中,$a_{j,1},...,a_{j,n_j}$ 对应 n_z 个访问监控以及相应 n_z 个权限或约束:$b_{t,z,1},...,b_{t,z,n_z}$.这 n_z 个权限或约束要么同时授予、要么同时撤销,所以第 z 条路径 n_z 个策略表示为“与”关系.即:

$$\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k})\rightarrow\vee_{z=1}^{m_t}\wedge_{k=1}^{n_z}p((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}),b_{t,z,k}).$$

$$\begin{aligned}&j\in\{1,2,...,n\},t\in\{1,2,...,u\},\forall((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k})\rightarrow\\&\vee_{z=1}^{m_t}\forall((s_i,F,ds_{t,z}),(o_i,do_{t,z}))\in SC_1PO_{t,z}\wedge_{k=1}^{n_z}p((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}),b_{t,z,k})\end{aligned} \quad (6)$$

- $\forall((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k})$为父结点;

- 以 m_t 个结点分别表示 $j\in\{1,2,...,n\}$,$\forall((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}))\in SC_jPO_{t,z}\wedge_{k=1}^{n_z}p((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}),b_{t,z,k})$, z=1,2,...,m_t 作为子结点.

父结点类型为“or”,表示父结点策略 $\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k})$为子结点策略 $\wedge_{k=1}^{n_z}p((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}),b_{t,z,k})$,$z$=1,2,..., m_t 的“或”.精化的策略精化树表述如图 4 点线框所示.

1.4 系统策略精化树的类型

基于前文策略形式描述和精化树构建方法,CIM,PIM 和 PSM 层策略精化可能形成的策略精化树形式如图 5~图 9 所示,其中的 I,II,III 代表结点策略的形式,I 代表$\wedge\{p_1,...,p_m\}$形式策略,II 代表 $p(S,O,a)$形式策略,III 代表 $p(s,o,a_1)\wedge...\wedge p(s,o,a_n)$形式策略.“and”,“set”和“or”为 *Node_type* 的值.

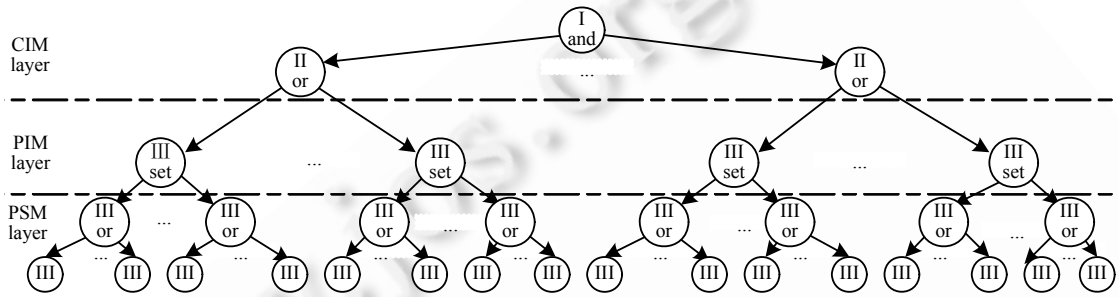

Fig.5 Refinement tree representation of the policy $\wedge\{p_1,...,p_m\}$ in CIM layer

图 5 CIM 层$\wedge\{p_1,...,p_m\}$策略精化树形式

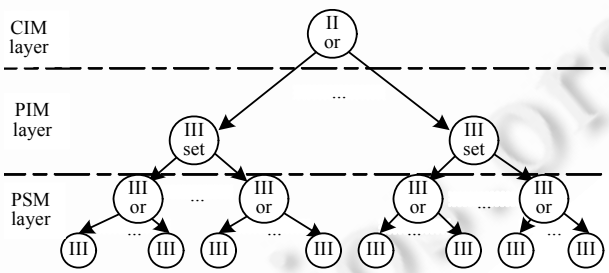

Fig.6 Refinement tree representation of the policy $p(S,O,a)$ in CIM layer

图 6 CIM 层 $p(S,O,a)$策略精化树形式

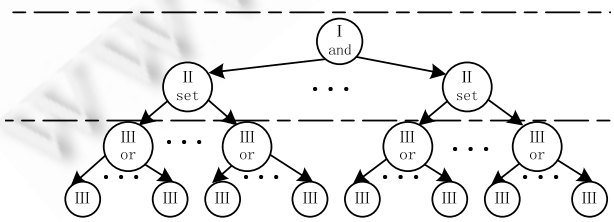

Fig.7 Refinement tree representation of the policy $\wedge\{p_1,...,p_m\}$ in PIM layer

图 7 PIM 层$\wedge\{p_1,...,p_m\}$策略精化树形式

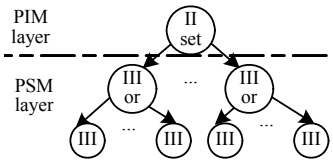

Fig.8 Refinement tree representation of the policy $p(S,O,a)$ in PIM layer

图 8 PIM 层 $p(S,O,a)$策略精化树形式

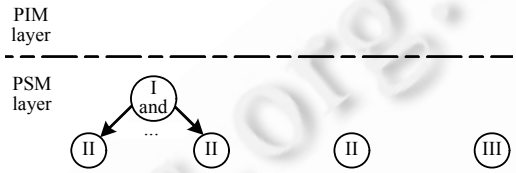

Fig.9 Refinement tree representationS of the policies that belongs to I, II, III in PSM layer

图 9 PSM 层 I 型、II 型、III 型策略精化树形式

2 基于 R 反驳计算消解策略冲突精化方法

如果逻辑系统推导的结论与新增加事实、结论、规则形成反驳时,通过修改逻辑系统的规则,使逻辑系统与新增加事实、结论、规则协调,这样的逻辑系统称为开放逻辑系统.R 反驳计算是一种基于一阶逻辑推理系统的开放逻辑.R 反驳计算原理见附录 A.

精化树的叶结点策略由根结点策略精化推导而来,精化原理[9−13]保证同一棵精化树结点上原子策略之间不存在冲突.但是系统范围不同精化树结点原子策略之间可能会存在冲突,需要进行冲突分析与消解.系统范围同一层次所有叶子结点原子策略冲突分析可以采用现有的访问控制权限空间分析方法或者我们设计的更为高效的基于集合求交递推分析方法[14],由于篇幅限制,此处略去介绍.

精化树是图形方式描述策略精化一阶逻辑系统,精化树叶子结点策略是根结点策略的逻辑结论.精化树上待消解叶子结点冲突策略,是精化树一阶逻辑推导的反驳.R 反驳计算消解精化树叶子结点冲突策略,修正因消解叶子结点冲突策略引起的精化树上逻辑关系失效,保持精化树策略之间关联属性协调.

策略精化计算自上而下分别对 CIM,PIM 和 PSM 三层策略进行形式转换、冲突判定、冲突消解与策略关联属性修正、精化等计算步骤,其中,冲突判定和消解还可能有不同类型策略冲突之分(比如 IPSec 策略),有序完成各层精化相关计算步骤,才能保证系统范围内三层策略精化具有完备性和一致性,策略组合、互斥、路径协同等关联属性协调.

第 2.1 节介绍 R 反驳计算消解精化树冲突策略、分析修正策略关联属性.第 2.2 节介绍精化相关计算步骤的计算顺序.第 2.3 节证明该精化方法保证系统范围内策略精化具有完备性和一致性,策略组合、互斥、路径协同等关联属性协调.

2.1 精化树策略冲突消解以及关联属性修正

假设经过叶子结点原子策略冲突分析,已知一棵精化树叶子结点需要消解的冲突原子策略.首先,调用 R 反驳计算规则消解叶结点冲突原子策略.根据前文介绍精化树构建和形式,叶子结点的兄弟类型有“and”,“or”和“set”这 3 种,根据形式结论必要前提定义,“and”兄弟和“or”兄弟与叶结点冲突原子策略具有“∧”,“∨”关系,是叶结点冲突原子策略的必要前提,需要调用 R 反驳计算规则消解;“set”关系兄弟结点策略不属于形式结论必要前提范围,不做处理.叶结点如果非根结点,冲突原子策略相应父结点策略,也属于其形式结论必要前提,也需要调用反驳规则消解.消解沿精化树反向递归计算,R 反驳计算可达性、可靠性、完备性保证递归调用 R 反驳计算规

则,消解精化树上的叶结点冲突原子策略的所有必要前提,最终精化树策略保持逻辑关系协调.

基于精化树的 R 反驳计算包括 5 个子算法,在描述 R 反驳计算算法之前,先介绍其中涉及的一些符号与函数的意义:

- $p_{L,k}(S,O,a)$常简单表示为 $p_{L,k}$,其中,L 表示策略层号,L=1 表示 CIM 层,L=2 表示 PIM 层,L=3 表示 PSM 层,k=1,2,3,...n 表示该层策略序号;
- $Root(p_{L,k})$=1 表示 $p_{L,k}$ 为精化树根结点策略,$Root(p_{L,k})\neq 1$ 表示 $p_{L,k}$ 非精化树根结点策略;
- $father(p_{L,k})$表示 $p_{L,k}$ 的父结点策略,$father(p_{L,k}.s)$,$father(p_{L,k}.o)$,$father(p_{L,k}.SO)$,$father(p_{L,k}.\alpha)$分别表示 $p_{L,k}$ 的 s,o,SO,α在$father(p_{L,k})$上对应的 s,o,SO,α值.比如:$p_{1,i}(s_i,o_i,a_i)$,$p_{2,k}((s_i,c_j),o_i,b_j)$为一棵精化树上的父子结点策略,那么 $father(p_{2,k}.(s_i,c_j))=p_{1,i}.s_i$;
- $brother_and(p_{L,k}(s_i,o_i,a))$为兄弟结点上与 $p_{L,k}(s_i,o_i,a)$具有“∧”关系的策略;
- $brother_or(p_{L,k}(s_i,o_i,a))$为兄弟结点上与 $p_{L,k}(s_i,o_i,a)$具有“∨”关系的策略;
- $son(p_{L,k}(s_i,o_i,a))$为 $p_{L,k}(s_i,o_i,a)$在精化树上的子树结点策略;
- $p_{or}(p_{L,k}(s_i,o_i,a))$为 0 表示 $p_{L,k}(s_i,o_i,a)$与兄弟之间形式为$(p_{L,k}\wedge\ldots\wedge p_{L,k+i})\vee\ldots\vee(p_{L,t}\wedge\ldots\wedge p_{L,t+j})$的“或”计算为 0;$p_{del\_or}$ 记录精化树上 $p_{or}(p_{L,k}(s_i,o_i,a)))=0$ 的策略(s_i,o_i)集合;
- $Q_{L,q,i}=\{Nodep:\{p_{L,k},\ldots\};S;O;a;SO;\}$是一个描述冲突策略的结构体,此处 S 表示冲突策略主体集合,O 表示冲突策略客体集合,α表示冲突需要消解的权限,SO 表示冲突需要消解的原子策略主客体序对集合,$Q_{L,q,i}$ 表示 L 层的、第 q 类型冲突的、第 i 个冲突策略集合描述结构体,$Nodep:\{p_{L,k},\ldots\}$表示一组包含该冲突策略集合的策略编号;
- $Q_{L,q,i}.p_{L,k}.SO=\{(s_i,o_i)|(s_i,o_i)\in p_{L,k}.SO\wedge(s_i,o_i)\in Q_{L,q,i}.SO\}$,用于策略 $p_{L,k}$ 所在精化树冲突消解.

约定:不影响计算情况下,为简化说明,算法中以 s_i 表示 s_i,(s_i,c_j),(s_i,c_j,ds_i),$(s_i,c_j,ds_{t,z})$等各层主体,以 o_i 表示 o_i,(o_i,do_i),$(o_i,do_{t,z})$等各层客体.

算法 1. 精化树叶结点冲突策略消解.

- 待消解叶子结点策略形式:$\forall(s_i,o_i)\in p_{L,k}.SOp_{L,k}(s_i,o_i,a)$;
- 叶子结点冲突策略为:$\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO\ p_{L,k}(s_t,o_t,a)$;
- 根据附录 A 中的 R 公理和定义 21,$\forall(s_i,o_i)\in p_{L,k}.SOp_{L,k}(s_i,o_i,a)$关于$\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a)$的理想事实反驳为 $p_{L,k}.SO=p_{L,k}.SO-Q_{L,q,i}.p_{L,k}.SO$,$\forall(s_i,o_i)\in p_{L,k}.SOp_{L,k}(s_i,o_i,a)$,即:

$$\text{delete } \forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a).$$

叶结点包括 II 型、III 型结点,II 型为 III 型结点特例,结点策略形式可以统一表示为

$$\forall(s_i,o_i)\in p_{L,k}.SOp_{L,k}(s_i,o_i,a_1)\wedge\ldots\wedge p_{L,k+m-1}(s_i,o_i,a_m).$$

根据附录 A 中定义 9、定义 10,叶结点上与冲突策略具有“∧”关系的策略属于冲突策略的必要前提,需要调用相应的 R-∧规则消解.

根据附录 A 中 R-∧规则和定义 21,$\forall(s_i,o_i)\in p_{L,k}.SOp_{L,k}(s_i,o_i,a_1)\wedge\ldots\wedge p_{L,k+m-1}(s_i,o_i,a_m)$关于$\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a)$的理想事实反驳为 $p_{L,k}.SO=p_{L,k}.SO-Q_{L,q,i}.p_{L,k}.SO$,$\forall(s_i,o_i)\in p_{L,k}.SOp_{L,k}(s_i,o_i,a_1)\wedge\ldots\wedge p_{L,k+m-1}(s_i,o_i,a_m)$,即:

$$\text{delete } \forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}p_{L,k}(s_t,o_t,a_1)\wedge\ldots\wedge p_{L,k+m-1}(s_t,o_t,a_m).$$

根据上述推理,叶结点冲突策略消解算法的伪代码为

Process $R\_node(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$

{For all $(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO$

delete $p_{L,k}(s_t,o_t,a_1)\wedge\ldots\wedge p_{L,k+m-1}(s_t,o_t,a_m)$;}.

算法 2. 精化树“与”兄弟策略关联属性修正.

被消解冲突策略为:$\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a)$;如果它存在“与”兄弟,根据精化树描述,被消解冲突策略同“与”兄弟结点策略关联形式为:$p_{ij}=p(s_j,o_i,a_i)$,$\forall(j\in N\wedge(s_j,o_i)\in SO)\wedge_{i=1}^{m}p_{ij}$ (式子$\forall(j\in N\wedge(s_j,o_i)\in SO)\wedge_{i=1}^{m}p_{ij}$ 的初始形式为$\forall j\in N\wedge_{i=1}^{m}p_{ij}$,经过 $p(s_j,o_i,a_i)$消解后,策略形式统一表示为$\forall(j\in N\wedge(s_j,o_i)\in SO)\wedge_{i=1}^{m}p_{ij}$),即:

$$brother\_and(\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a))=\forall j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO,\forall i=\{1,...,m\}p_{ij}.$$

根据附录 A 中定义 9、定义 10,“与”兄弟结点这些具有“与”关系的策略属于被消解冲突策略的必要前提,根据附录 A 中 R_∧规则和定义 21,$\forall(j\in N\wedge(s_j,o_i)\in SO)\wedge_{i=1}^{m}p_{ij}$ 关于$\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a)$的理想事实反驳为

$$p_{L,k}.SO=p_{L,k}.SO-Q_{L,q,i}.p_{L,k}.SO;\forall(s_j,o_i)\in p_{L,k}.SO\wedge_{i=1}^{m}p_{ij},$$

即:

$$\text{delete } brother\_and(\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a)).$$

另外,删除冲突策略“与”兄弟结点上具有“与”关系的策略,同时删除“与兄弟”策略精化子树,因此,系统也不会形成多余的下层策略.

根据上述推理,“与”兄弟策略关联属性修正的伪代码为

Process $R\_bro\wedge(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$

{

For all $(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO$

{delete $brother\_and((s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a))$;

delete $son(brother\_and((s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SOp_{L,k}(s_t,o_t,a))$;};}.

算法 3. 精化树“或”兄弟策略关联属性修正.

被消解冲突策略为:$j\in\{1,2,...,m\},\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{u=1}^{n_j}p_{L,x}(s_t,o_t,a_{j,u}))$;如果它存在“或”兄弟,根据精化树描述,被消解冲突策略同“或”兄弟结点策略关联形式为

$$j\in\{1,2,...,m\},\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO(\wedge_{u=1}^{n_j}p_{L,x}(s_i,o_i,a_{1,u}))\vee...\vee(\wedge_{u=1}^{n_j}p_{L,y}(s_i,o_i,a_{m,u})).$$

此时,$\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{u=1}^{n_j}p_{L,x}(s_t,o_t,a_{j,u}))=0$.根据附录 A 中定义 9、定义 10,下层支撑策略唯一对应的必要前提是其上层策略,而下层只要具有“或”关系的兄弟结点策略中任何一个兄弟结点策略成立,上层策略就存在下层支撑策略,只有当所有兄弟结点策略失效时,上层策略失去下层支撑策略.

根据附录 A 中定义 21 和 R_∨规则:

- 如果$\forall j=\{1,...,m\}\wedge_{u=1}^{n_j}p_{L,k}(s_t,o_t,a_{j,u})=0$,则所有结点“或”兄弟策略失效:

$$brother\_or(\forall(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{u=1}^{n_j}p_{L,x}(s_t,o_t,a_{j,u}))=\Phi,p_{or}(\wedge_{u=1}^{n_j}p_{L,x}(s_t,o_t,a_{j,u}))=0,$$

  p_{del_or} 记录用于进一步父结点相关必要前提消解;

- 否则,标记 $p_{or}(\wedge_{u=1}^{n_j}p_{L,x}(s_t,o_t,a_{j,u}))=1$.

根据上述推理,“或”兄弟策略关联属性修正算法的伪代码为

Process $R\_bro\vee(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$

{$p_{del\_or}=\Phi$;

{For all $(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO$

$p_{or}(p_{L,k}(s_t,o_t,\alpha))=1$;

If $brother\_or(p_{L,k}(s_t,o_t,\alpha))=\Phi$;

{$p_{or}(p_{L,k}(s_t,o_t,\alpha))=0$;

$p_{del\_or}=p_{del\_or}\cup p_{L,k}.(s_t,o_t)$;};};

$Q_{L,q,i}.p_{L,k}.SO=p_{del\_or}$;}.

算法 4. 精化树父子结点策略关联属性修正.

由精化树构建可以知道,精化树父子结点策略关系形式有 3 种.

(1) 父结点类型为 and.父子策略关系为$\forall(j\in N\wedge(s_j,o_i)\in SO)\wedge_{i=1}^{m}p_{ij}\rightarrow\wedge_{i=1}^{m}p_{ij}$,被消解子结点策略为$\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO)\wedge_{i=1}^{m}p_{ij}$.根据附录 A 中定义 9、定义 10,策略 $father(\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO)$

$\wedge_{i=1}^{m} p_{ij}$)属于被消解冲突策略的必要前提,根据附录 A 中定义 21 和 R 删除规则-I,$\forall(j\in N\wedge(s_j,o_i)\in SO)$ $\wedge_{i=1}^{m} p_{ij} \rightarrow \wedge_{i=1}^{m} p_{ij}$ 关于$\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO)\wedge_{i=1}^{m} p_{ij}$ 的理想事实反驳为

$$\forall(j\in N\wedge(s_j,o_i)\in\{SO-Q_{L,q,i}.p_{L,k}.SO\})\wedge_{i=1}^{m} p_{ij} \rightarrow \wedge_{i=1}^{m} p_{ij},$$

即:

$$\text{delete } father(\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{i=1}^{m} p_{ij}).$$

根据精化树构建原理:$father(\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{i=1}^{m} p_{ij})= father(\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO\ p_{L,k})$.

(2) 父结点类型为 set.为与 $p_{L,k}$ 区分,此处以 v 代替变量 k,父子策略关系表示为:

$$j\in\{1,2,...,n\},\forall(s_i,o_i)\in SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})\rightarrow \forall t=\{1,...,u\},\forall(s_i,o_i)\in SC_jPO_t\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v}).$$

被消解策略为$\forall(s_i,o_i)\in Q_{L,q,i}.\ p_{L,k}.SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})$.

根据附录 A 中定义 9、定义 10,策略 $father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v}))$属于被消解冲突策略的必要前提,根据附录 A 中定义 21 和 R 删除规则-I,$j\in\{1,2,...,n\},\forall(s_i,o_i)\in SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})$关于$\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})$的理想事实反驳为$\forall(s_i,o_i)\in(SO\text{-}father(Q_{L,q,i}.p_{L,k}.SO))\wedge_{v=1}^{n_j} p((s_i,c_j),o_i,a_{j,v})$,即:

$$\text{delete } father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})).$$

根据精化树原理:$father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v}))= father(\forall(j\in N\wedge(s_j,o_i)\in Q_{L,q,i}.p_{L,k}.SO\ p_{L,k})$.

(3) 父结点类型为 or.为说明简洁,→左右均由 s_i、o_i 表示各层主客体, SO 表示策略主客体序对集合.上层访问路径权限集合:$\{a_{j,1},..., a_{j,n_j}\}$,对应下层每条访问路径权限集合的并集:

$$\{b_{t,1,1},..., b_{t,1,n_1}\}\cup,...,\cup\{b_{t,1,1},..., b_{t,m_t,n_{m_t}}\}.$$

v 替换变量 k.精化推导式子(4)、(6)简化表示为:

$$j\in\{1,2,...,n\},t=\{1,...,u\},\forall(s_i,o_i)\in SO\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})\rightarrow \vee_{z=1}^{m_t}\wedge_{v=1}^{n_z} p(s_i,o_i,b_{t,z,v}).$$

被消解策略为$\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\vee_{z=1}^{m_t}\wedge_{v=1}^{n_z} p(s_i,o_i,b_{t,z,v})$.

根据附录 A 中定义 9、定义 10,策略 $father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\vee_{z=1}^{m_t}\wedge_{v=1}^{n_z} p(s_i,o_i,b_{t,z,v}))$属于被消解冲突策略的必要前提,根据附录 A 中定义 21 和 R 删除规则-I,$j\in\{1,2,...,n\},\forall(s_i,o_i)\in SO\wedge_{k=1}^{n_j} p(s_i,o_i,a_{j,v})$关于$\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\vee_{z=1}^{m_t}\wedge_{v=1}^{n_z} p(s_i,o_i,b_{t,z,v})$的理想事实反驳为 $j\in\{1,2,...,n\},\forall(s_i,o_i)\in\{SO\text{-}father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO)\}$ $\wedge_{v=1}^{n_j} p(s_i,o_i,a_{j,v})$,即:

$$\text{delete } father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\vee_{z=1}^{m_t}\wedge_{v=1}^{n_z} p(s_i,o_i,b_{t,z,v})).$$

根据精化树构建原理:$father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\ p_{L,k})= father(\forall(s_i,o_i)\in Q_{L,q,i}.p_{L,k}.SO\vee_{z=1}^{m_t}\wedge_{v=1}^{n_z} p(s_i,o_i,b_{t,z,v}))$.

父子结点策略关联属性修正的算法伪代码为

Process R_farI($Q_{L,q,i}.p_{L,k}.SO,p_{L,k}$)

{For all $(s_t,o_t)\in Q_{L,q,i}.p_{L,k}.SO$

delete $father(p_{L,k}.(s_t,o_t,\alpha))$;

$Q_{L,q,i}.p_{L,k}.SO=father(Q_{L,q,i}.p_{L,k}.SO)$;

$p_{L,k}=father(p_{L,k})$;}.

算法 5. 精化树基于叶结点冲突策略 R 反驳递归计算算法.

Process $tree(Q_{L,q,i},p_{L,k})$是主函数算法的伪代码,它基于叶结点 $p_{L,k}$ 冲突策略 $Q_{L,q,i}$ 递归调用算法 1~算法 4 消解 $p_{L,k}$ 所在精化树冲突策略并修正精化树关联属性、保持精化树逻辑关系协调.

Process $tree(Q_{L,q,i},p_{L,k})$

{$Q_{L,q,i}.p_{L,k}.SO=Q_{L,q,i}.SO$;

Call process $R\_node(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$;

A: If $(ROOT(p_{L,k})=1)\wedge(p_{L,k}.SO=\Phi)$

Delete $Q_{L,q,i}.Nodep.p_{L,k}$;

else if $(ROOT(p_{L,k})=1)\wedge(p_{L,k}.SO\neq\Phi)$

break;

Else

{Case $father(p_{L,k}).Node\_type$="and"

Call Process $R\_bro\wedge(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$;

Call Process $R\_far\mathrm{I}(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$;

goto (A);

Case $father(p_{L,k}).Node\_type$="or"

Call Process $R\_bro\vee(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$;

if $p_{del\_or}\neq\Phi$

{Call Process $R\_far\mathrm{I}(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$;

goto (A);};

Case $father(p_{L,k}).Node\_type$="set"

Call Process $R\_far\mathrm{I}(Q_{L,q,i}.p_{L,k}.SO,p_{L,k})$;

goto (A);};}.

例 1:PIM 层策略“售后人员通过 Web 写维修日志,必须同时写技术统计表.”为简洁起见,以 W 简化表示 Web. $S=\{(s_1,\mathrm{W}),(s_2,\mathrm{W})\}$,$s_1$=李军,$s_2$=高明;$O=\{z_1,z_2\}$,$z_1$=维修日志,$z_2$=技术统计表.w 表示“写”,$a_1=a_2=\mathrm{w}$.以$\wedge\{p_1p_2\}$表示该策略,$p_1=\forall j\in\{1,2\}p((s_j,\mathrm{W}),z_1,a_1)$,$p_2=\forall j\in\{1,2\}p((s_j,\mathrm{W}),z_2,a_2)$.结点类型为“and”.分别以结点表示策略 p_1 与 p_2,作为$\wedge\{p_1p_2\}$的子结点,子结点类型为“set”.p_1,p_2 的 SC_1OP 分别为 $SC_1OP_{(p_1)}$,$SC_1OP_{(p_2)}$.

假设 PSM 层体系结构参数,主体(s_1,W),(s_2,W)物理位置分别为 ds_1,ds_2,在同一访问控制入口点;客体物理位置分别为 do_1,do_2,是在同一维修服务器上.主客体具有相同的访问控制起点和终点,那么:

$$SC_1OP_{(p_1)}=\{((s_1,\mathrm{W},ds_1),(z_1,do_1)),((s_2,\mathrm{W},ds_2),(z_1,do_1))\},\ SC_1OP_{(p_1)t}=SC_1OP_{(p_1)},t=1.$$

$$p_1\rightarrow\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_1,do_1))\in SC_1OP_{(p_1)1}\,p((s_j,\mathrm{W}),z_1,a_1).$$

$\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_1,do_1))\in SC_1OP_{(p_1)1}\,p((s_j,\mathrm{W}),z_1,a_1)$作为 p_1 子结点.

同理:

$$p_2\rightarrow\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_2,do_2))\in SC_2OP_{(p_2)1}\,p((s_j,\mathrm{W}),z_2,a_2).$$

$\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_2,do_2))\in SC_2OP_{(p_2)1}\,p((s_j,\mathrm{W}),z_2,a_2)$作为 p_2 子结点.

假设 PSM 层策略$\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_1,do_1))\in SC_1OP_{(p_1)1}\,p((s_j,\mathrm{W}),z_1,a_1)$与$\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_2,do_2))\in SC_2OP_{(p_2)1}\,p((s_j,\mathrm{W}),z_2,a_2)$均存在唯一网络路径,路径访问监控依次为:包过滤防火墙、系统 http 端口认证、维修服务器访问控制.PIM 层策略 $p((s_j,\mathrm{W}),z_1,a_1)$访问权限 a_1 对应 PSM 层权限:维修服务器访问控制 w、系统有线 http 端口认证 K1 的权限、防火墙入权限 in1 和防火墙出权限 out1.所以:

$$p((s_j,\mathrm{W}),z_1,a_1)\rightarrow p((s_j,\mathrm{W},ds_{1,1}),(z_1,do_{1,1}),\mathrm{w})\wedge p((s_j,\mathrm{W},ds_{1,1}),(z_1,do_{1,1}),\mathrm{in1})\wedge p((s_j,\mathrm{W},ds_{1,1}),(z_1,do_{1,1}),\mathrm{out1})\wedge p((s_j,\mathrm{W},ds_{1,1}),\mathrm{K1},\mathrm{auth});$$

$$\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_1,do_1))\in SC_1OP_{(p_1)1}\,p((s_j,\mathrm{W}),z_1,a_1)\rightarrow\forall j\in\{1,2\}p((s_j,\mathrm{W},ds_{1,1}),(z_1,do_{1,1}),\mathrm{w})\wedge p((s_j,\mathrm{W},ds_{1,1}),(z_1,do_{1,1}),\mathrm{in1})\wedge p((s_j,\mathrm{W},ds_{1,1}),(z_1,do_{1,1}),\mathrm{out1})\wedge p((s_j,\mathrm{W},ds_{1,1}),\mathrm{K1},\mathrm{auth}).$$

将上面式子符号“$\rightarrow$”右边策略集合结点作为左边策略集合结点的子结点,父结点类型为“or”.

同理:

$$\forall j\in\{1,2\}\forall((s_j,\mathrm{W},ds_j),(z_2,do_2))\in SC_2OP_{(p_2)1}\,p((s_j,\mathrm{W}),z_2,a_2)\rightarrow\forall j\in\{1,2\}p((s_j,\mathrm{W},ds_{2,1}),(z_2,do_{2,1}),\mathrm{w})\wedge p((s_j,\mathrm{W},ds_{2,1}),(z_2,do_{2,1}),\mathrm{in1})\wedge p((s_j,\mathrm{W},ds_{2,1}),(z_2,do_{2,1}),\mathrm{out1})\wedge p((s_j,\mathrm{W},ds_{2,1}),\mathrm{K1},\mathrm{auth}).$$

将上面式子符号“$\rightarrow$”右边策略集合结点作为左边策略集合结点的子结点,父结点类型为“or”.策略精化树如图 10 所示白色方框表示的精化树所示.

假如 IDS 发现 http 协议 IP 地址为 202.119.28.211、端口为 80 的用户行为有恶意,设置策略禁止它的所有访问权限,记(202.119.28.211,80)=ds_1,策略为 $p_{3,t}((*,W,ds_{1,1}),*,\neg*)$,将它加入当前 PSM 层的策略集合中,表示为图 10 中 PSM 层黑粗线框策略结点,其中,权限"¬*"表示禁止任何行为.通过 PSM 层所有叶子结点策略集合的原子策略冲突分析,发现图 10 策略精化树上 PSM 层的虚线框叶子结点上原子策略 $p_{3,k}((s_1,W,ds_{1,1}),(z_1,do_{1,1}),w)$与策略 $p_{3,t}((*,W,ds_{1,1}),*,\neg*)$之间存在冲突:

$$Q_{3,1,i}:\{Nodes:\{p_{3,k},\ldots\};S:\{(s_1,W,ds_{1,1})\};O:\{(z_1,do_{1,1})\};\alpha:w;SO:\{(s_1,W,ds_{1,1}),(z_1,do_{1,1})\};\}.$$

采用 R 反驳计算对精化树该冲突策略进行消解并分析修正其关联属性的过程如下:

(1) 假设策略冲突消解规则为否定优先,Process $tree(Q_{L,q,i},p_{L,k})$首先调用叶子结点反驳计算函数 Process $R_node(Q_{3,1,i}.p_{3,k}.SO,p_{3,k})$运用 R 公理,消解 $p_{3,k}$ 中策略主客体序对属于 $Q_{3,1,i}.p_{3,k}.SO$ 的原子策略 $p((s_1,W,ds_{1,1}),(z_1,do_{1,1}),w)$.策略 $p((s_1,W,ds_{1,1}),(z_1,do_{1,1}),w)$存在访问路径协同策略:

$$p((s_1,W,ds_{1,1}),(z_1,do_{1,1}),w)\wedge p((s_1,W,ds_{1,1}),(z_1,do_{1,1}),in1)\wedge p((s_1,W,ds_{1,1}),(z_1,do_{1,1}),out1)\wedge p((s_1,W,ds_{1,1}),K1,auth)$$

R-∧规则计算保证结点上这些具有"∧"关系的原子策略也同时被取消;

(2) $p_{3,k}$ 非根且父结点 $father(p_{3,k})$的 $Node\_type$="or",调用"或"兄弟反驳计算函数 Process $R\_bro\vee(Q_{3,1,i}.p_{3,k}.SO,p_{3,k}),p_{or}(p_{3,k})=0$;

(3) 因为 $p_{or}(p_{3,k})=0$,调用父子反驳计算函数 Process $R\_farI(Q_{3,1,i}.p_{3,k}.SO,p_{3,k})$,运用 R-删除规则 I 删除 $father(p_{3,k})$,即,父策略 $p((s_1,W),z_1,a_1)$;

(4) $father(p((s_1,W),z_1,a_1))$非根且 $Node\_type$="set",调用函数 Process $R\_farI(Q_{3,1,i}.p_{3,k}.SO,father(p_{3,k}))$,运用 R-删除规则 I 删除 $father(father(p_{3,k}))$,即,$p((s_1,W),z_1,a_1)$;

(5) $father(father(p_{3,k}))$的父结点非根且父结点∧$\{p_1,p_2\}$的 $Node\_type$="and",调用函数 Process $R\_bro\wedge(Q_{3,1,i}.p_{3,k}.SO,father(father(p_{3,k})))$,运用 R-∧规则删除 $father(father(p_{3,k}))$的"与"兄弟策略 $p((s_1,W),z_2,a_2)$,并取消它的后代结点 PSM 层策略:

$$p((s_1,W),z_2,a_2),p((s_1,W,ds_{2,1}),(z_2,do_{2,1}),w)\wedge p((s_1,W,ds_{2,1}),(z_2,do_{2,1}),in1)\wedge$$
$$p((s_1,W,ds_{2,1}),(z_2,do_{2,1}),out1)\wedge p((s_1,W,ds_{2,1}),K1,auth);$$

(6) 调用函数 Process $R\_farI(Q_{3,1,i}.p_{3,k}.SO,father(father(p_{3,k})))$,运用 R-删除规则 I 删除∧$\{p_1,p_2\}$上的 $p(u_1,z_1,a_1)\wedge p(u_1,z_2,a_2)$,∧$\{p_1,p_2\}$为非空根结点,结束该精化树的反驳计算.

被消解策略为图 10 中灰框标识中的策略.

2.2 精化相关计算顺序

应用系统策略包括 CIM,PIM 和 PSM 三层策略,每层计算包括策略形式转换、叶结点策略不同类型冲突判定和基于精化树的冲突消解、精化推导等步骤,有序计算是保证精化完备性和一致性的重要方面.

1) 层间计算顺序

R 反驳计算前提是原先的逻辑推理系统必须协调,上层策略精化形成下层策略,所以上层策略冲突分析、消解、修正与精化推导是下层策略冲突分析与精化的前提;否则,上层策略如果因冲突失效,推导下层策略的逻辑系统协调性不成立,下层策略 R 反驳计算前提不能成立.

2) 同层计算顺序

(1) 策略转换是原子策略冲突判定的前提.

策略描述根据应用需求,可能采用 DAC,RBAC,MAC 模型,这些策略即便在模型内部是一致的,也有与模型外其他策略形成冲突的可能,需要转换为本文策略形式,参与系统整体策略分析.

(2) 同层策略冲突判断、消解及修正由此引起的策略关联属性失效是策略精化的前提.

R 反驳计算前提是原先逻辑系统是协调的,所以同一层策略冲突分析、消解、修正必须在策略精化之前;否则,策略因冲突失效,精化树策略之间逻辑推导关系不协调,下层策略 R 反驳计算前提不能成立.

(3) 叶子结点原子策略冲突判断是策略 R 反驳计算消解冲突的前提.

(4) 认证±auth 冲突等同于访问控制策略权限±a 冲突消解.

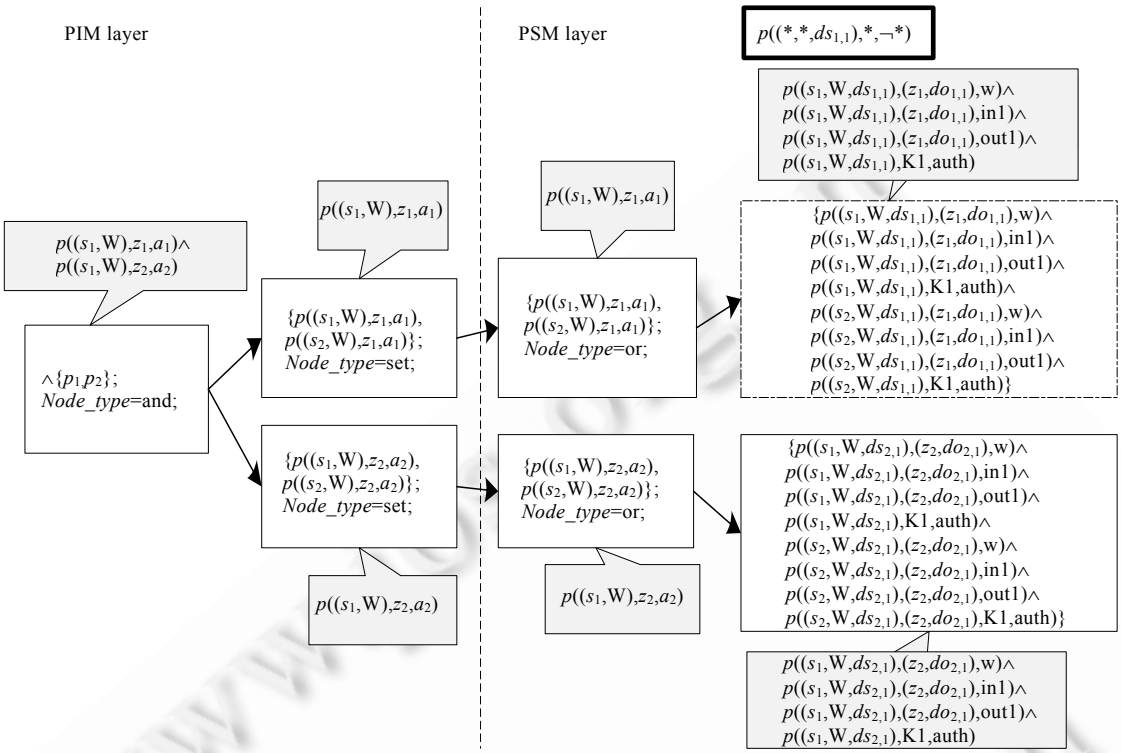

Fig.10 Example of conflict resolution in policy refinement tree

图 10 策略精化树冲突消解示例

(5) 互斥规则分析在权限类型冲突消解最后.

互斥规则是一种重要、常用的权限策略,描述两个不能同时成立的策略.对于原子策略 $p(s_1,o_1,\alpha)$与 $p(s_2,o_2,\beta)$,如果,$s_1=s_2,o_1\neq o_2,p(s_1,o_1,\alpha)$与 $p(s_2,o_2,\beta)$不能同时成立,称 $p(s_1,o_1,\alpha)$与 $p(s_2,o_2,\beta)$互斥,表示为:

$$(p(s_1,o_1,\alpha)\wedge\neg p(s_2,o_2,\beta))\vee(\neg p(s_1,o_1,\alpha)\wedge p(s_2,o_2,\beta)).$$

互斥规则不一定有固定的主体值,它在确定的原子策略基础上判断并加以修正.所以,互斥规则分析在原子策略权限各冲突类型消解之后,采用相同方法消解确定取消的原子策略.

(6) 权限类型冲突消解先于通信安全冲突类型消解,通信安全冲突类型按序分析.

对于策略精化树而言,不同类型冲突谁先消解冲突计算是等价的,但是一方面,访问控制路径相关通信安全约束是由访问控制权限策略精化而来;另一方面,从文献[15]可知,±入、±出类型通信冲突分析结果是其机密性、完整性安全服务冲突分析的前提,隧道冲突在确定的策略之下,总能产生正确的配置,不影响±入、±出和机密性、完整性策略,所以先消解访问控制权限策略冲突,接着按序消解通信约束冲突,再分析互斥规则及冲突,最后生成隧道配置策略,这样有序分析不仅能够按照应用通信安全策略特点生成系统范围一致策略,还可以减少计算量.

(7) 策略精化计算结束.

根据平台相关层安全监控技术,将策略转换为安全监控技术的配置形式,比如 Kerberos,Linux DAC 等[12,13].

表 1 基于上述分析系统策略精化的计算顺序,计算步骤由 L,T,q 参数标示:L 表示层号;T 表示计算类型,T=1 表示策略转换,T=2 表示原子策略冲突分析与消解,T=3 表示互斥规则分析,T=4 表示策略精化,T=5 表示将策略转换为所在安全监控技术策略形式;q 表示原子策略冲突类型.比如第 1 步 CIM 层策略转换,标识计算步骤的参数值为 L=1,T=1,q=0.

SaaS 平台具有动态弹性特征,策略分析参数也随之动态变化.另外,应用策略也会根据应用需求而变化,所以策略精化计算的动态自适应能力是非常需要的.访问控制策略精化作为 SaaS 平台应用服务模块,通过监控机制在线获取相关参数,动态分析并保持策略的一致性.表 1 相关参数栏目参数的变动改变策略的精化结果,需要触发新的精化计算.由于精化计算各步骤顺序进行,当精化计算相关策略或环境参数变化,先序计算步骤的输入未发生改变,因此计算结果也不会改变,所以只需要从出现变化相关的位置开始按顺序向下计算,这样就使得计算简化、效率提高.表 1 中,相关参数所在位置的 L,T,R 是 SaaS 平台各种变化触发的精化计算位置.比如 PSM 层 IDS 生成新策略,那么分析可以从 $L=3,T=1,q=0$ 处开始计算,不需要像文献[12,13]的技术,需要从头开始计算.而且本文原子的策略冲突分析,策略是以原子策略集合形式记录在精化树结点,分析选择基于集合求交的递推算法[14],得出的冲突原子策略也是原子策略集合形式,R 反驳计算以集合形式而非单个形式消解原子策略冲突,显然有利于提高效率.

Table 1 Application system policies refinement calculation sequence

表 1 系统策略精化计算顺序

策略层次	精化与冲突计算		相关参数
CIM (L=1)	RBAC,MAC 策略经模型内部分析后,转换为本文策略直接表示(T=1).		CIM 层对象和结构信息, CIM 层其他描述形式策略
	叶子结点原子策略冲突判断与基于原子策略冲突判断的 R 反驳计算(T=2)	CIM 层叶子结点原子策略冲突判断(q=0)	CIM 层非互斥策略, 策略冲突定义修改
		$Q_{1,1}$ 原子策略反驳计算(q=1)	
		…	
	互斥规则分析(T=3)		CIM 层互斥策略
	策略精化(T=4)		PIM 层对象和结构信息
PIM (L=2)	RBAC,MAC 策略经模型内部分析后,转换为本文策略直接表示(T=1).		PIM 层其他描述形式策略
	叶子结点原子策略冲突判断与基于原子策略冲突判断的 R 反驳计算(T=2)	PIM 层叶子结点原子策略冲突判断(q=0)	PIM 层非互斥策略, 策略冲突定义修改
		$Q_{2,1}$ 原子策略冲突反驳计算(q=1)	
		…	
	互斥规则分析(T=3)		PIM 层互斥策略
	策略精化(T=4)//依次基于客体物理结点精化、基于客体访问拓扑路径的精化		PSM 层对象和结构信息
PSM (L=3)	RBAC,MAC 策略经模型内部分析后,转换为本文策略直接表示(T=1).		PSM 层其他描述形式策略
	叶子结点原子策略冲突判断与基于原子策略冲突判断的 R 反驳计算(T=2)	PSM 层叶子结点原子策略冲突判断(q=0)	PSM 层非互斥策略, 策略冲突定义修改
		$Q_{3,1}$ 原子策略冲突反驳计算(q=1)	
		$Q_{3,2}$ 原子策略冲突反驳计算(q=2)	
		…	
		$Q_{3,*}$:IPSec 通信冲突分析(q=*,...)	
		…	
	互斥规则分析(T=3)		PSM 层互斥策略
	$Q_{3,*}$:IPSec 隧道分析(q=*,...)		
	将策略转换为所在安全监控策略形式(T=5)		

2.3 基于R反驳计算消解策略冲突精化方法正确性证明

定理 4. 基于 R 反驳计算消解原子策略冲突、修正策略之间关联属性的策略精化是可计算的,精化计算结果保持完备性和一致性.

定理证明见附录 B.

3 性能分析与实验

随着云技术的发展,动态自适应成为重要属性,因此,我们模拟实验分析本文精化方法的计算性能,判断其于云平台的适用性.

对于一定安全级别集中管理的信息系统,随着企业规模的增长,主客体数量增长,访问控制精化需要分析计算安全监控的主客体权限呈几何级数增加,而需要分析的冲突类型差别不是很大.一类常见的应用,比如企业规

模不同,但是安全策略总量、各个层次策略数量差别并不是很大.另一类常见的应用,比如不断增加数量的分支机构或营业网点,安全策略总量以及各个层次策略数量增长明显.本文模拟测试这两种类型应用随着企业规模的增长,基于 R 反驳策略精化计算的性能.策略数量级分布为万、十万、百万、千万、亿.

精化、原子策略冲突分析、反驳计算、(原子策略)互斥判断在三层是依次递归执行.由于策略精化自上而下,下层策略数量必然大于上层策略;同时,由于访问路径和访问控制点的精化,所以下层的原子策略数量远远大于上层.本文选择测试 PIM 层精化为 PSM 层策略、PSM 层原子策略冲突分析、PSM 层反驳计算、PSM 层(原子策略)互斥判断的时间,以假设 1 和假设 2 估算基于 R 反驳消解策略冲突、修正策略关联属性的精化计算的性能.

假设 1:CIM 层初始状态开始策略精化所用时间为策略形式转换时间×3+系统(L=2,T=4)处开始精化计算平均时间×2+系统(L=3,T=2,q=0)处开始原子策略冲突计算平均时间×3+系统(L=3,T=2,$q\neq0$)处开始反驳计算平均时间×3+系统(L=3,T=3)处开始互斥规则判断计算平均时间×3.

假设 2:运行过程中 PSM 层策略或环境动态调整情况下,策略精化所用时间为 PSM 层策略形式转换时间+系统(L=3,T=2,q=0)处开始原子策略冲突计算平均时间+系统(L=3,T=2,$q\neq0$)处开始反驳计算平均时间+系统(L=3,T=3)处开始互斥规则判断计算.

实验系统为图 1 所示一个常见企业信息服务系统访问控制策略.表 2 为企业信息服务系统常见的访问控制策略表形式,假设安全等级为 2,读权限的数据在网络通信中采用 IPSec AH 保护,写权限的数据在网络通信采用 IPSec ESP 保护.原子冲突类型:4 个(±权限、±通信、¬通信/IPSec AH、¬通信/IPSec ESP). 实验平台为:intel E5-1650 3.2GHz,32GB memory,3TB storage,windows 8.1 professional.实验结果见表 3.

从表 3 的实验结果可以看出:CIM 层初始状态开始策略精化所用时间估计在不足 1s~60s 之间;PSM 层策略或环境动态调整情况下,策略精化所用时间估计在不足 0.5s~20s 之间.从模拟实验和分析可以看出,该算法性能处于非实时应用的动态自适应需求可以接受的范围.

4 结 论

本文基于策略和策略关联的精化树描述以及原子策略冲突判断,采用开放逻辑 R 反驳计算消解策略冲突、修正策略关联属性失效,分析同层策略转化、叶结点策略冲突计算、层间冲突消解与精化的计算关系,设计策略精化递归计算流程,保证系统范围策略精化完备性与一致性,拓展了访问控制策略精化技术以下几方面应用能力: (1) 提高精化中策略关联属性分析能力,分析并保证策略之间组合、互斥、精化、路径协同关联属性正确性;(2) 提供自由选择冲突解决规则,使冲突解决规则不再局制于上层策略取代下层策略,符合应用需要;(3) 提供 CIM 层策略向 PIM 层策略精化时不同访问方式的精化计算;(4) 能够处理 PIM 层、PSM 层多条路径访问控制策略的协同一致;(5) 基于精化树 R 反驳计算的不同类型冲突消解计算次序先后是等价的,这样,像 IPSec 策略这类需要有序分析多种冲突的技术,就可以按照应用属性有序分析;(6) 精化树记录访问控制目标与实现机制以及实现技术和参数之间的映射关系,是访问控制 SLA 的直观呈现,可以提供安全目标及其实现的审核;(7) 本文设计的精化技术通过 R 反驳计算,直接分析修正精化树上记录的策略和策略关系,而不是像原先这些技术,在技术实现层,基于网络模型及网络模型对应的访问控制策略模型的检索匹配推导路径协同策略,因为网络拓扑及参数非常复杂,难于建模,即使网络模型已经建立,计算性能也依赖网络模型库的好坏和大小.策略精化性能模拟实验和分析显示,该精化技术适应 SaaS 平台应用访问控制动态自适应需求.

Table 2 Enterprise information system access control policies table

表 2 企业信息系统访问控制策略表

策略	安全等级
内部员工允许使用邮件服务器个人帐户: $p_{1,1}$(内部员工,邮件服务器个人帐户,使用).	1
销售和售后人员允许读技术资料: $p_{1,2}$(销售和售后人员,技术资料,读).	1
…	
销售经理允许通过 FTP 或者 WEB 读(销售/销售汇总文件) $p_{2,1}$(销售经理,FTP),(销售/销售汇总文件),读); $p_{2,2}$(销售经理,WEB),(销售/销售汇总文件),读).	2
销售人员允许通过 WEB 写(销售/销售单据 *i*): $p_{2,3}$(销售人员,WEB),(销售/销售单据 *i*),写).	2
售后人员允许通过 WEB 填写(维修/维修日志)同时必须通过 WEB 填写(维修/技术统计表): $p_{2,4}$(维修人员,WEB),(维修/维修日志),写)∧$P_{2,5}$(维修人员,WEB),(维修/技术统计表),写).	1
成本会计允许通过 WEB 写(财务/成本核算): $p_{2,6}$(成本会计,WEB),(财务/成本核算),写).	2
成本会计通过 WEB 写(财务/成本核算/文件 1)同时必须通过 WEB 填写(财务/成本核算/文件 2): $p_{2,7}$(成本会计,WEB),(财务/成本核算/文件 1),写)∧$P_{2,8}$(成本会计,WEB),(财务/成本核算/文件 2),写).	
出纳允许通过 WEB 写(财务/出纳/现金支取登记): $p_{2,9}$(出纳,WEB),(财务/出纳/现金支取登记),写).	2
会计允许通过 WEB 写(财务/会计): $p_{2,10}$(会计,WEB),(财务/会计),写).	2
互斥规则:(任何人通过 WEB 写(财务/出纳))⊕(任何人通过 WEB 写(财务/会计)): $p_{2,11}$((*,WEB),(财务/出纳),写)⊕$P_{2,12}$((*,WEB),(财务/会计),写).	2
…	
禁止 202.119.*.*的 IP 地址入(防火墙规则): $p_{3,1}$((*,*,*,¬202.119.*.*,*),(*,*,*,202.119.16.*,*),¬入).	1
允许 IP 地址为 202.119.18.*的入(防火墙规则): $p_{3,2}$((*,*,*,202.119.18.*,*),(*,*,*,202.119.16.*,*),入).	1
禁止 IP 地址为(202.119.18.116)的机器执行任何操作(IDS 规则): $p_{3,3}$((*,*,*,202.119.18.116,*),*,¬*).	1
…	

Table 3 Experimental data

表 3 实验数据

系统主体个数	系统客体个数	策略主体集合元素平均个数	策略客体集合元素平均个数	策略数量	三层安全监控中原子策略数量级	PSM 层原子策略冲突数量,冲突原子策略涉及精化树数量	系统(L=2, T=4)开始精化计算平均时间(s)	系统(L=3, T=2,q≠0)开始反驳计算平均时间(s)	系统(L=3, T=2,q=0)开始原子策略冲突计算平均时间(s)	系统(L=3, T=3)开始互斥规则判断计算平均时间(s)
50	100	10-20	10~20	10~20	万级	10, 5	0.063	0.183	0.203	0.093
100	500	30-50	50~100	10~20	十万级	200, 5	0.106	0.245	0.294	0.206
100	500	10~20	10~20	100~200	十万级	200, 50	0.153	0.369	0.398	0.355
500	1 000	100~150	50~100	10~20	百万级	1 000, 5	0.237	0.587	0.601	0.596
500	1 000	10~20	10~20	500~1 000	百万级	1 000, 250	0.439	0.785	0.807	0.799
1 000	5 000	200~300	500~1 000	10~20	千万级	2 000, 5	0.506	0.895	0.926	0.937

Table 3 Experimental data (Continued)

表 3 实验数据(续)

系统主体个数	系统客体个数	策略主体集合元素平均个数	策略客体集合元素平均个数	策略数量	三层安全监控中原子策略数量级	PSM 层原子策略冲突数量,冲突原子策略涉及精化树数量	系统(L=2, T=4)开始精化计算平均时间(s)	系统(L=3, T=2,q≠0)开始反驳计算平均时间(s)	系统(L=3, T=2,q=0)开始原子策略冲突计算平均时间(s)	系统(L=3, T=3)开始互斥规则判断计算平均时间(s)
1 000	5 000	10~20	10~20	1 000~2 000	千万级	2 000, 500	1.124	1.262	1.304	1.329
10 000	50 000	3 000~5 000	5 000~10 000	10~20	亿级	10 000, 5	1.525	2.025	2.105	2.200
10 000	50 000	10~20	10~20	1 000~2 000	亿级	10 000, 2 500	3.214	5.214	5.263	5.369

References:

[1] Sloman M. Policy driven management for distributed systems. Journal of Network and Systems Management, 1994,2(4):333−360. [doi: 10.1007/BF02283186]

[2] Jason B. The SOA management landscape. Zapthink. 2006. http://www.zapthink.com/2006/11/30/the-soa-management-landscape/

[3] Maullo MJ, Calo SB. Policy management: An architecture and approach. In: Proc. of the 1st Int'l Workshop on Systems Management. Piscataway: IEEE, 1993. 13−26. [doi: 10.1109/IWSM.1993.315293]

[4] Pieters W, Dimkov T, Pavlovic D. Security policy alignment: A formal approach. IEEE Systems Journal, 2013,7:275−287. [doi: 10.1109/JSYST.2012.2221933]

[5] Mont CM, Baldwin A, Goh C. POWER prototype: Towards integrated policy-based management. In: Proc. of the Network Operations and Management Symp. Piscataway: IEEE/IFIP, 2000. 789−802. [doi: 10.1109/NOMS.2000.830429]

[6] The Open Group. SLAmanagement Handbook. Vol.4, TMF, 2004. http://www.afutt.org/Qostic/qostic1/SLA-DI-USG-TMF-060091-SLA_TMForum.pdf

[7] Kumari P, Pretschner A. Deriving implementation-level policiesfor usage control enforcement. In: Proc. of the CODASPY 2012. New York: ACM Press, 2012. 83−94. [doi: 10.1145/2133601.2133612]

[8] Jayaraman K, Ganesh V, Tripunitara M, Rinard M, Chapin S. Automatic error finding in access-control policies. In: Proc. of the CCS 2011. New York: ACM Press, 2011. 163−174. [doi: 10.1145/2046707.2046727]

[9] Lampson B, Abadi M, Burrows M, Wobber E. Authentication in distributed systems: Theory and practice. ACM Trans. on Computer Systems, 1992,10(4):265−310. [doi: 10.1145/138873.138874]

[10] Abadi M, Burrows M, Lampson B, Plotkin G. A calculus for access control in distributed systems. ACM Trans. on Programming Languages and Systems, 1993,15(3):706−734. [doi: 10.1145/155183.155225]

[11] Davy S, Jennings B, Strassner J. On harnessing information models and ontologies for policy conflict analysis. In: Proc. of the IFIP/IEEE Int'l Symp. on Integrated Network Management 2009. 2009. 821−826. [doi: 10.1109/INM.2009.5188889]

[12] Lück I, Vögel S, Krumm H. Model-Based configuration of VPNs. In: Proc. of the Network Operations and Management Symp. 2002. IEEE/IFIP, 2002. 589−602. [doi: 10.1109/NOMS.2002.1015610]

[13] Albuquerque JP, Krumm H, Geus PL. Formal validation of automated policy refinement in the management of network security systems. Int'l Journal of Information Security, 2010,9(2):99−125. [doi: 10.1007/s10207-010-0101-6]

[14] Wu YH, Huang H, Zhou JK, Zeng QK. Conflict analysis of distributed application access control policies refinement. Journal of Computer Applications, 2014,34(2):421−427 (in Chinese with English abstract).

[15] Fu Z, Wu FS. Automatic generation of IPSec/VPN security policies in an intra-domain environment. In: Proc. of the 12th Int Worshop on Distributed Systems. Nancy: DSOM, 2001.

[16] 李未.一个开放的逻辑系统.中国科学(A 辑),1992,10:1103−1113.

[17] 李未.R−演算:一个修正程序规约的演算系统.中国科学(E 辑),2002,32(5):662−673.

[18] Li W. R-Calculus: An inference system for belief revision. The Computer Journal, 2007,50(4):378−390. [doi: 10.1093/comjnl/bxl069]

[19] Su KL. R-Reconstruction in open logic. Chinese Science Bulletin, 1995,40(5):365−366.

[20] Su KL. Open logic about facts refute and general hypothesis. Chinese Science Bulletin, 1994,39(16):1441−1443 (in Chinese with English abstract).

附中文参考文献:

[14] 吴迎红,黄皓,周靖康,曾庆凯.分布式应用访问控制策略精化冲突分析.计算机应用,2014,34(2):421−427.

[20] 苏开乐.关于事实反驳与一般假说的开放逻辑.科学通报,1994,39(16):1441−1443.

附录 A: R 反驳计算

R 反驳计算基于一阶逻辑 G 系统[16−18],一阶逻辑 G 系统公式和推理规则形式描述如下:

定义 1(公式).

$$A:=P(t_1,\ldots,t_n)|t_1=t_2|\neg A|A\wedge B|A\vee B|A\rightarrow B|\forall x.A|\exists x.A,$$

其中,$t_i,i=1,\ldots,n$ 为项,$P(t_1,\ldots,t_n)$为 n 元谓词.

定义 2(¬规则).

- $\neg$-L: $\dfrac{\Gamma,\Delta\vdash A,\Lambda}{\Gamma,\neg A,\Delta\vdash,\Lambda}$;
- $\neg$-R: $\dfrac{A,\Gamma\vdash\Lambda,\Theta}{\Gamma\vdash\Lambda,\neg A,\Theta}$.

定义 3(∨规则).

- $\vee$-L: $\dfrac{\Gamma,A,\Delta\vdash\Lambda\ \Gamma,B,\Delta\vdash\Lambda}{\Gamma,A\vee B,\vdash\Lambda}$;
- $\vee$-R: $\dfrac{\Gamma\vdash\Lambda,A,B,\Theta}{\Gamma\vdash\Lambda,A\vee B,\Theta}$.

定义 4(∧规则).

- $\wedge$-L: $\dfrac{\Gamma,A,B,\Delta\vdash\Lambda}{\Gamma,A\wedge B,\Delta\vdash\Lambda}$;
- $\wedge$-R: $\dfrac{\Gamma\vdash\Lambda,A,\Theta\ \Gamma\vdash\Lambda,B,\Theta}{\Gamma\vdash\Lambda,A\wedge B,\Theta}$.

定义 5(→规则).

- $\rightarrow$-L: $\dfrac{\Gamma,\Delta\vdash A,\Lambda\ B,\Gamma,\Delta\vdash\Lambda}{\Gamma,A\rightarrow B,\Delta\vdash\Lambda}$;
- $\rightarrow$-R: $\dfrac{A,\Gamma\vdash B,\Lambda,\Theta}{\Gamma\vdash\Lambda,A\rightarrow B,\Theta}$.

定义 6(∀规则).

- $\forall$-L: $\dfrac{\Gamma,A\left[\dfrac{t}{x}\right],\forall xA(x),\Delta\vdash\Lambda}{\Gamma,\forall xA(x),\Delta\vdash\Lambda}$;
- $\forall$-R: $\dfrac{\Gamma\vdash\Lambda,A\left[\dfrac{t}{x}\right],\Theta}{\Gamma\vdash\Lambda,\forall xA(x),\Theta}$.

定义 7(∃规则).

- $\exists$-L: $\dfrac{\Gamma,A\left[\dfrac{y}{x}\right],\Delta\vdash\Lambda}{\Gamma,\exists xA(x),\Delta\vdash\Lambda}$;
- $\exists$-R: $\dfrac{\Gamma\vdash\Lambda,A\left[\dfrac{t}{x}\right],\exists xA(x),\Theta}{\Gamma\vdash\Lambda,\exists xA(x),\Theta}$.

定义 8(**R 反驳**). 设Γ为一个形式理论,又设Δ是由有穷个原子语句或原子语句的否定组成的形式理论.如果

Δ与Γ不协调,那么称Δ是Γ的一个 R 反驳.

例如一阶逻辑系统Γ,Γ存在逻辑结论 A,如果存在事实$\neg A$,则事实$\neg A$ 与Γ不协调,是Γ的一个 R 反驳.

R 反驳计算消解Γ中与Δ不协调的规则集合,也称为Γ中与Δ不协调的、需要消解的形式结论的必要前提,并保证经过消解的Γ与Δ协调.

定义 9(形式结论的必要前提). 设Γ为公式集合,A 为公式并且$\Gamma\vdash A$ 可证.称公式集合Δ是公式 A 关于Γ的必要前提集合,如果$\Delta\subseteq\Gamma$是使$\Delta\vdash A$ 成立的极小公式集合,即:如果$\Delta'\subset\Delta$,那么$\Delta'\vdash A$ 不可证.称 B 为形式结论 A 的一个必要前提,如果公式 $B\in\Delta$成立,并记为 $B\mapsto_A A$.

定义 10(关于证明树的前提集合). 设Γ为公式集合,A 为公式并且$\Gamma\vdash A$ 可证.又设 Y 是其证明树,而 P,Q 和 R 是在证明树 Y 中出现的公式.

(1) 如果Γ是一个公式集合,Γ,$P\vdash P$ 是证明树 Y 的叶,那么$\vdash$左边的 P 是右边的 P 关于证明树 Y 的前提;

(2) 如果证明树 Y 的一个结点是 G 系统的某一个右规则的实例,而 Q 是此规则的分母中出现的 $B\wedge C$,$B\vee C$,$B\to C$,$\neg B$,$\forall xB(x)$,$\exists xB(x)$,即规则的主公式,而 P 是此规则的分子中出现的 B,C,$B[t/x]$,$B[y/x]$,即规则的辅公式,那么 P 是 Q 关于证明树 Y 的前提;

(3) 如果证明树 Y 的一个结点是某一个左规则的实例,而 P 是此规则的分母中出现的 $B\wedge C$,$B\vee C$,$B\to C$,$\neg B$,$\forall xB(x)$,$\exists xB(x)$,即规则的主公式,而 Q 是此规则的分子中出现的 B,C,$B[t/x]$,$B[y/x]$,即规则的辅公式,那么 P 是 Q 关于证明树 Y 的前提,而辅公式 B,C,$B[t/x]$及 $B[y/x]$又是规则分母中$\vdash$右边的辅公式关于证明树 Y 的前提;

(4) 如果 P 是 Q 关于证明树 Y 的前提,而 Q 是 R 关于证明树 Y 的前提,那么 P 是 R 关于证明树 Y 的前提.

令$\rho(\Gamma,A,Y)$为 A 的关于证明树 Y 的所有前提组成的集合.

一个可证序贯可能对应不同的证明树,必要前提集合可能不唯一,但是每一个证明树对应唯一必要前提集.

R 反驳包括以下计算规则:

定义 11(R 公理).

$$A,\Delta|\neg A,\Gamma\Rightarrow A,\Delta|\Gamma.$$

它表示如果形式理论Γ包含$\neg A$ 与(事实反驳)A 不协调,那么形式理论Γ必须删除 A.

定义 12(R-$\wedge$规则).

- $\dfrac{\Delta|A,\Gamma\Rightarrow\Delta|\Gamma}{\Delta|A\wedge B,\Gamma\Rightarrow\Delta|\Gamma}$;
- $\dfrac{\Delta|B,\Gamma\Rightarrow\Delta|\Gamma}{\Delta|A\wedge B,\Gamma\Rightarrow\Delta|\Gamma}$.

定义 13(R-$\vee$规则):

$$\frac{\Delta|A,\Gamma\Rightarrow\Delta|\Gamma\quad\Delta|B,\Gamma\Rightarrow\Delta|\Gamma}{\Delta|A\vee B,\Gamma\Rightarrow\Delta|\Gamma}.$$

定义 14(R-$\to$规则):

$$\frac{\Delta|\neg A,\Gamma\Rightarrow\Delta|\Gamma\quad\Delta|B,\Gamma\Rightarrow\Delta|\Gamma}{\Delta|A\to B,\Gamma\Rightarrow\Delta|\Gamma}.$$

定义 15(R-$\forall$规则):

$$\frac{\Delta\left|A\left[\frac{t}{x}\right],\Gamma\Rightarrow\Delta|\Gamma\right.}{\Delta|\forall xA(x),\Gamma\Rightarrow\Delta|\Gamma}.$$

定义 16(R-$\exists$规则):

$$\frac{\Delta\left|A\left[\frac{y}{x}\right],\Gamma\Rightarrow\Delta|\Gamma\right.}{\Delta|\exists xA(x),\Gamma\Rightarrow\Delta|\Gamma}.$$

定义 17(R 删除规则-I):

$$\frac{\Gamma_1, A, \Gamma_2 \vdash \boldsymbol{C}\ A \to_{\Gamma} \boldsymbol{C}\ \Delta | \boldsymbol{C}, \Gamma_2 \Rightarrow \Delta | \Gamma_2}{\Delta | \Gamma_1, A, \Gamma_2 \Rightarrow \Delta | \Gamma_1, \Gamma_2}.$$

定义 18(R 删除规则-II):

$$\frac{\Gamma_1, A \vdash B\ A \to_{\Gamma} B\ B, \Gamma_2 \vdash \boldsymbol{C}\ \Delta | \boldsymbol{C}, \Gamma_2 \Rightarrow \Gamma_2}{\Delta | \Gamma_1, A, \Gamma_2 \Rightarrow \Delta | \Gamma_1, \Gamma_2}.$$

苏开乐[19,20]进一步提出建立 R 反驳的概念,不仅能祛除受反驳的规则,还能在受反驳的规则中祛除不协调值的同时,保留协调的值.

定义 19. 对任意两个 A 的 U-事实反驳 M_0,M_1:

- 我们说 M_1 优先 M_0(记为 $M_1 \leqslant^* M_0$ 或 $M_0 \geqslant^* M_1$)当且仅当$|M_0| \subset |M_1|$,且对任何Γ中公式 $B_{(x)}$和任何个体 $\alpha \in |M_0|$,若 $M_0 \models B_{(\alpha)}$,则 $M_1 \models B_{(\alpha)}$;
- 我们说 M_0 等价于 M_1(记为 $M_1 \approx M_2$),如果 $M_0 \leqslant^* M_1$ 且 $M_1 \leqslant^* M_0$.

定义 20(理想事实反驳). 我们称 A 的 U-事实反驳 M 为 A 的 U-理想事实反驳,如果 M 为上述优先关系的极小元,即:不存在 A 的 U-事实反驳 M',使得 $M' \leqslant^* M$ 且 $M' \neq M$.

Γ^*为一般的假说,即,一非空的相容的公式(可含自由变元 x)的集.A 为Γ^*推出的一句子,U 为一可数无穷的个体域.模型 M 为 A 的 U-事实反驳,如果 M 的全域为 U 的子集,且 $M \models \neg A$.对给定的一个 A 的 U-理想事实反驳 M,我们如下给出一个关于Γ的 M 的 R-重构 RM.令 $TH(\bar{M})$为 M 的等价类中每个模型都成立的句子集合,R_M 可从原假说Γ^*做出以下修改而得到:

(1) 把$\neg A$ 加在Γ中;

(2) 删去使得 $TH(\bar{M}) \vdash (\forall)(\neg B(x))$成立的$\Gamma^*$中的公式 $B(x)$;

保留使得使得 $TH(\bar{M}) \vdash (\forall)(B(x))$成立的$\Gamma^*$中的公式 $B(x)$;

(4) 对不符合条件(2)、条件(3)的Γ^*中的公式 $B(x)$改换成$\vee S \to B(x)$形式的公式,其中,$\vee S$ 表示是中公式的析取,S 表示使得 $TH(\bar{M}) \vdash (x)(P(x) \to B(x))$成立的公式 $P(x)$的集合.若$\vee S$ 不能规约为一个公式 $Q(x)$,我们 则 把$\vee S \to B(x)$看成 $P_0(x) \to B(x), \ldots, P_i(x) \to \pi(x), \ldots$等无穷个公式,其中,$P_i(x) \in S$.

定义 21. Γ^*关于 A 的事实反驳的 R-重构集为 $R(\Gamma^*, A)=\{RM | M$ 为 A 的 U-理想事实反驳$\}$.

例如对$\Gamma^*=\{P(x)\}$,$A \equiv (x)P(x)$,关于 A 的事实反驳的 R-重构仅有一个为:$\{(\exists x)(\neg P(x)), \exists y(\neg P(y)) \& x \neq y \to P(x)\}$.

R 反驳计算具有以下属性:

定理 1(R 可达性). R 演算是 R 可达的.

它表明:形式系统Γ关于Δ反驳,总可以通过有限次递归调用 R 规则演算,排除Γ中与Δ不协调的规则,形成与Δ和谐的新形式系统Γ'.

定理 2(R 可靠性). R 演算是 R 可靠的.

它表明:形式系统Γ关于Δ反驳,总存在Γ的 R 反驳模型 M,使得 $M \models \Delta$并且$\Gamma_{M(\Delta)}=\Gamma'$成立.

定理 3(R 完备性). R 演算是 R 完备的.

它表明:形式系统Γ关于Δ反驳,模型 M 是Γ关于Δ的 R 反驳模型,总存在 R 转换序列,使得$\Delta | \Gamma \Rightarrow^* \Delta | \Gamma_{M(\Delta)}$.

附录 B: 基于 R 反驳计算消解策略冲突精化方法正确性证明

定理 4. 基于 R 反驳计算消解原子策略冲突、修正策略之间关联属性的策略精化是可计算的,精化计算结果保持完备性和一致性.

证明:

(1) 可计算性

系统策略是有限的;精化(不考虑平台无法实现的情况)规则保证策略总可以精化生成唯一精化树;原子策略冲突判断是可计算的;精化规则属于 R 反驳规则计算范围,R 反驳计算的可达性、完备性、可靠性保证精化

树基于叶子结点冲突的 R 反驳计算是可计算,所以 CIM 层、PIM 层、PSM 层的精化各个步骤都是可计算的,因此,基于 R 反驳计算消解策略冲突的精化方法是可计算的.

(2) 策略之间符合精化完备性要求

完备性指系统策略精化经过 R 反驳计算,上层策略在下层存在支撑策略.本文不考虑平台无法实现的情况,所以从精化树构建可以知道,上层策略作为精化树根结点必然能够精化生成下层叶子结点支撑策略.策略精化中,R 反驳计算包括 CIM 层、PIM 层和 PSM 层反驳计算.

a) CIM 层叶子结点策略形式可以统一表示为

$$p(S,O,a)=\forall(s_i,o_i)\in SOp(s_i,o_i,a).$$

如果叶子结点为根结点,则不存在上层策略.

如果是组合策略$\wedge\{p_1,...,p_m\}$的叶结点,由于精化树生成是唯一,所以下层策略与上层策略唯一对应关系是待消解策略及与其具有“与”关系的兄弟结点策略对应的父结点策略.根据定义 9、定义 10,下层支撑策略唯一对应的必要前提是其上层对应的父结点策略.由消解算法 5、算法 1、算法 2、算法 4 可以知道:反驳计算消去冲突策略本身、冲突策略同层相关“与”兄弟策略以及“与”兄弟策略的后代策略,上层唯一对应的父结点策略,保留精化树其他策略上下层对应精化关系,保持策略的组合、互斥以及精化计算等策略之间关系.

保留的 CIM 层叶结点策略精化生成 PIM 层支撑策略.

b) PIM 层叶子结点策略形式可以统一表示为:

$$\forall(s_i,o_i)\in SO\wedge_{k=1}^{n_j}p((s_i,c_j),o_i,a_{j,k}),j\in\{1,2,...,n\}.$$

如果叶子结点为根结点,则不存在上层策略.

如果是 PIM 层组合策略$\wedge\{p_1,...,p_m\}$的叶结点,由于精化树生成是唯一,所以下层策略与上层策略唯一对应关系是待消解策略及与其具有“与”关系的兄弟结点策略对应的父结点策略.根据定义 9、定义 10,下层支撑策略唯一对应的上层父结点策略是其必要前提.由消解算法 5、算法 1、算法 2、算法 4 可以知道:反驳计算消去冲突策略本身、冲突策略同层相关“与”兄弟策略以及“与”兄弟策略的后代策略,上层唯一对应的父结点策略,保留精化树其他策略上下层对应精化关系,保持策略的组合、互斥以及精化计算等策略之间关系.

如果是 CIM 层精化而来的叶结点,由于精化树生成是唯一,所以下层策略与上层策略唯一对应关系是待消解策略及与其具有“或”关系的兄弟结点策略对应的父结点策略.根据定义 9、定义 10,下层支撑策略唯一对应的上层父结点策略是其必要前提.而下层只要具有“或”关系的兄弟结点策略中任何一个兄弟结点策略成立,上层策略就存在下层支撑策略,只有当所有兄弟结点策略失效时,上层策略失去下层支撑策略.由消解算法 5、算法 1、算法 3 可以知道:反驳计算消去冲突策略本身以及本结点具有“与”关系的策略,如果并非所有“或”兄弟策略失效,$p_{or}\neq0$,结束计算;如果所有“或”兄弟策略失效,$p_{or}=0$,算法 4 消去它们唯一对应的 CIM 层父结点策略,算法 5 递归 R 反驳计算如前 CIM 层冲突消解,保留精化树其他策略上下层对应精化关系,保持策略的组合、互斥以及精化计算等策略之间关系.所以,PIM 层反驳计算保留精化树其他策略上下层对应精化关系,保持策略的组合以及精化计算等策略之间关系.

保留 PIM 层策略生成 PSM 层支撑策略.

c) PSM 层叶子结点策略形式可以统一表示为

$$j\in\{1,2,...,n\},\forall((s_i,c_j,ds_i),(o_i,do_i))\in SC_jPO_t\wedge_{k=1}^{n_z}p((s_i,c_j,ds_{t,z}),(o_i,do_{t,z}),b_{t,z,k}),z=1,2,...,m_t.$$

如果叶子结点为根结点,则不存在上层策略.

如果是 PSM 层组合策略$\wedge\{p_1,...,p_m\}$的叶结点,由于精化树生成是唯一,所以下层策略与上层策略唯一对应关系是待消解策略及与其具有“与”关系的兄弟结点策略对应的父结点策略.根据定义 9、定义 10,下层支撑策略唯一对应的上层策略是其必要前提.由消解算法 5、算法 1、算法 2、算法 4 可以知道:反驳计算消去冲突策略本身、冲突策略同层相关“与”兄弟策略以及“与”兄弟策略的后代策略,上层唯一对应的父结点策略,保留精化树其他策略上下层对应精化关系,保持策略的组合、互斥以及精化计算等策略之间关系.

如果是 PIM 层精化而来的叶结点,由于精化树生成是唯一,所以下层策略与上层策略唯一对应关系是待消

解策略及与其具有“或”关系的兄弟结点策略对应的父结点策略.根据定义 9、定义 10,下层支撑策略唯一对应的上层父结点策略是其必要前提.而下层只要具有“或”关系的兄弟结点策略中任何一个兄弟结点策略成立,上层策略就存在下层支撑策略,只有当所有兄弟结点策略失效时,上层策略失去下层支撑策略.由消解算法 5、算法 1、算法 3 得知:反驳计算消去冲突策略本身以及本结点具有“与”关系的策略,如果并非所有“或”兄弟策略失效,$p_{or}\neq 0$,结束计算;如果所有“或”兄弟策略失效,$p_{or}=0$,算法 4 消去它们唯一对应的 PSM 层父结点策略.再根据定义 9、定义 10,该 PSM 层策略还存在唯一父结点策略是其 PIM 层必要前提.由消解算法 5、算法 4 可以知道:反驳计算消去其唯一对应的上层父结点策略,PIM 层父结点策略被消解,算法 5 递归 R 反驳计算原理如前 PIM 层、CIM 层冲突消解,直到消去所有被消解下层策略对应的上层策略.所以,PSM 层反驳计算保留精化树其他策略上下层对应精化关系,保持策略的组合、互斥以及精化计算等策略之间关系.

综上可知:CIM 层、PIM 层和 PSM 层基于 R 反驳计算消解冲突策略、修正精化树策略关联属性的精化计算,均保证策略之间满足精化完备性要求.所以经过精化计算,系统策略之间满足精化完备性要求,精化树保持策略的组合、互斥以及精化计算等策略之间关系.

(3) 系统原子策略之间一致,没有冲突

a) 对于系统每层叶子结点,由原子策略冲突分析判定和冲突类型分析顺序保证保留的叶子结点原子策略之间没有冲突;

b) 由 CIM 层、PIM 层、PSM 层递归计算原子策略冲突,消解原子策略冲突以及失效的关联策略,R 反驳计算可达性、可靠性、完备性保证保留的叶子结点与所在精化树其他结点策略协调;

c) 保留的叶子结点原子策略 p 与其他精化树非叶子结点策略存在 3 种关系:

- 其他精化树非叶子结点推出 $p.p$ 与精化推出 p 的策略协调,没有冲突;
- 其他精化树非叶子结点推出$\neg p.\neg p$ 与 p 冲突,原子策略冲突分析必然定义$\neg p$ 为冲突策略,叶子结点 R 反驳计算必然删除$\neg p$,同时删除相关推导前提策略,所以必然与 p 最终保持协调;
- 其他精化树非叶子结点既不推出 p,也不推出$\neg p$,与 p 无冲突.

所以,系统同层以及不同层原子策略之间一致,没有冲突. □

吴迎红(1966－),女,江苏南京人,高级工程师,主要研究领域为信息安全.

曾庆凯(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全.

黄皓(1957－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全.

张迪明(1986－),男,讲师,主要研究领域为操作系统,形式化验证.

吕庆伟(1985－),男,助理工程师,主要研究领域为信息安全.