文章编号:1001-9081(2011)01-0115-03 doi:10.3724/SP.J.1087.2011.00115

隐私安全策略中的变更影响分析

王 强,刘 峤,秦志光
(电子科技大学 计算机科学与工程学院,成都 610054)
(wangq@uestc.edu.cn; liuqiao@uestc.edu.cn; qinzg@uestc.edu.cn)

摘 要:为了解决Web分布式系统中的隐私安全策略在制定和变更中的错误很难被发现的问题,提出了策略变更中各种情况的相应变更影响分析算法。对以可扩展访问控制标记语言(XACML)为代表的隐私安全策略语言中的变更理论进行了研究,定义了变更分析中的相关概念,通过把策略中的字符串元素转化成对应整数值建立一个优化的树形数据结构,利用树的特征分析变更后果。这使得一个管理员可以在正式应用策略变更前检验即将实施的变更是否符合自己的真正意图,从而大大增强系统安全性。最后实现了一个原型系统,并可以应用到其他标准策略语言。

关键词:可扩展访问控制标记语言;隐私安全策略;变更影响分析;树形结构
中图分类号:TP309.2;TP302.7 **文献标志码**:A

Change impact analysis in authorization policies

WANG Qiang, LIU Qiao, QIN Zhi-guang
(*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*)

Abstract: Due to the lack of tools for analyzing policies, most authorization policies on the Internet have been plagued with policy errors. A policy error either creates security vulnerabilities that will compromise the security of information technology system. A major source of policy errors stems from policy changes. Authorization policies often need to be changed as networks evolve and new requests emerge. The theory and algorithms for authorization policy change-impact analysis were presented. Algorithms in this paper took an authorization policy and a proposed change as input, and then output the accurate impact of the change. Thus, an administrator can verify a proposed change before committing it. A prototype was built to demonstrate the use of the algorithms.

Key words: eXtensible Access Control Markup Language (XACML); authorization policy; change impact analysis; tree structure

0 引言

随着Internet越来越广泛的使用,企业及众多组织机构怎样在基于Web的分布式应用环境中准确有效地制定安全隐私策略就变得越来越重要。为了解决上述问题,统一标准的隐私安全策略被提了出来,比如P3P、EPAL、XACML[1]等。

由于安全策略的独立和统一性,任何策略规则的变动都可能会对全局产生影响。假设一个企业把数据库中只有经理以上级别人员能够查看的敏感信息发布到了公开的企业门户网站上,管理员发现了问题,并迅速修改了其中一条安全策略禁止了敏感信息外流。但管理员很难知道这一个安全策略的修改是否影响到了其他相关用户,目前看来禁止了敏感信息外流,有可能用户组中某些级别的用户也一并受到了影响。当管理员在制定修改安全策略时,需要对自己的变更有明确的认识,规则的添加、删除和修改是否得到了和计划中相同的结果。为了确保安全策略的定制能达到预期的结果,变更影响分析变得尤为重要。

目前已有的安全策略的研究成果主要集中在其本身安全特性的应用以及规则优化、冲突检测等方面。文献[2]提出了一种XACML规则冲突及冗余分析方法,没有对规则变更做阐述;文献[3]讲述了一种规则的组合算法。变更分析的概念[4]也被应用到传统的软件开发中,但未见针对目前大规模Web分布式环境的安全策略变更分析。

本文主要使用XACML作为例子来做变更影响分析,用一个变化作为输入,输出精确的变更影响分析结果,告知管理员规则变化的影响以及预知变更后果,以提高策略制定的准确性和正确性。

1 XACML实例

由于篇幅限制,本文使用文字描述一个简化的学校学生管理系统XACML策略实例以及策略包括的若干规则,真实的XACML例子可以参见文献[5]。借助文献[5]的方法,首先把XACML内的各个元素转化为数字方式的表达方式,每一个元素均对应一个整数值。XACML共有四种规则匹配算法,这里选用最先应用,即遇到的第一条匹配的相关规则的评估结果作为最终授权决策的评估结果。其余三种组合算法可以同样运用本文的方法进行分析,只需要根据相应算法做简单参数调整即可。如表1所示,XACML各元素均和一个整数值一一对应。

三条规则的文字描述如下。

规则一 学生和行政人员不能修改分数。

规则二 教授、讲师或行政人员可以修改或者读取分数

收稿日期:2010-06-30;**修回日期**:2010-08-16。 **基金项目**:国家863计划项目(2009AA01Z4222)。

作者简介:王强(1981-),男,四川富顺人,博士研究生,主要研究方向:安全策略、Web安全; 刘峤(1974-),男,四川成都人,博士研究生,主要研究方向:人工智能、数据挖掘、入侵检测; 秦志光(1956-),男,四川成都人,教授,博士生导师,主要研究方向:信息系统安全、智能交通、电子商务。

和履历。

规则三　学生可以修改或者读取履历。

缺省规则　拒绝任何不匹配以上规则的请求。

表1　策略元素对应表

主体	资源	动作
学生:0	分数:0	改变:0
行政人员:1	履历:1	读取:1
教授:2		
讲师:3		

三条规则的数字化表示形式如下：

$R_1: S \in [0,1] \wedge R \in [0,0] \wedge A \in [0,0] \rightarrow$Deny

$R_2: S \in [1,3] \wedge R \in [0,1] \wedge A \in [0,1] \rightarrow$Permit

$R_3: S \in [0,0] \wedge R \in [1,1] \wedge A \in [0,1] \rightarrow$Permit

$R_\infty: S \in [0,3] \wedge R \in [0,1] \wedge A \in [0,1] \rightarrow$Deny

其中 S 代表主体，R 代表资源，A 代表动作。最后一条规则 R_∞ 是每个策略必须包含的一个完整性说明，确保每一个请求都有相应的结果返回，在不影响所有规则语义的情况下确保规则的完整性和安全性。

假设有一个请求为“行政人想改变分数”，此请求的数字化表达式为 $Q(1,0,0)$，在实际运用中，此请求会被送入XACML的策略评估模块PDP(Policy Decision Point)，然后PDP根据规则匹配情况得出结论是拒绝还是接受这个请求。

2　规则变更分析

本文主要分析三种对隐私安全策略的变更模式，分别是：规则删除、规则插入和规则修改。根据上一章的实例，可以看出，假如删除规则一，对于请求 Q 的结论将发生变化，删除前PDP拒绝 Q，删除后PDP接受 Q。下面将分析更多更复杂的规则变更。

首先定义三种变更。

定义1　假设一个隐私安全策略有 n 条规则。

规则删除是指删除规则 R_i，其中 $1 \leqslant i \leqslant n-1$；

规则插入是指在规则 R_i 和 R_{i+1} 在之间插入规则 R，其中 $1 \leqslant i \leqslant n-1$；

规则修改是指修改规则 R_i 为 R_i'，其中 $1 \leqslant i \leqslant n-1$。

定义2　全匹配集合和匹配集合。在一个策略 P 中，如果请求 q 匹配规则 R_i，这些请求的集合就是 R_i 的全匹配集合，表示为 $M(R_i)$。如果请求 q 匹配规则 R_i 但不匹配任意 $R_j(j<i)$，这些请求的集合就是 R_i 的匹配集合，表示为 $R(R_i,P)$。

接着讨论三个变更分析理论，这里使用 $r.D$ 来表示一条规则的结论，使用 $p(q)$ 表示请求 Q 在策略 P 里第一条匹配规则的结论。

定理1　若策略 P 包含 n 条规则$\langle R_1,R_2,\cdots,R_n\rangle$。假设删除 R_i 得到策略 $P'\langle R_1,\cdots,R_{i-1},R_{i+1},\cdots,R_n\rangle$，用 g 表示策略$\langle R_{i+1},R_{i+2},\cdots,R_n\rangle$，对于任意请求 q，有如下两种情况：

1)假如 $q \in R(R_i,P)$，则 $p(q)=r_i.D$ 且 $p'(q)=g(q)$，P 和 P' 对请求 q 可能有不同结论；

2)假如 $q \notin R(R_i,P)$，则 $p(q)=p'(q)$，P 和 P' 对请求 q 有相同结论。

定理2　若策略 P 包含 n 条规则$\langle R_1,R_2,\cdots,R_n\rangle$。假设插入规则 R 得到策略 $P'\langle R_1,\cdots,R_{i-1},R_i,R_{i+1},\cdots,R_n\rangle$，用 g 表示策略$\langle R_{i+1},\cdots,R_n\rangle$，对于任意请求 q，有如下两种情况：

1)假如 $q \in R(R_i,P)$，则 $p(q)=g(q)$ 且 $p'(q)=r_i.D$，P 和 P' 对请求 q 可能有不同结论。

2)假如 $q \notin R(R_i,P)$，则 $p(q)=p'(q)$，P 和 P' 对请求 q 有相同结论。

定理3　若策略 P 包含 n 条规则$\langle R_1,R_2,\cdots,R_n\rangle$。假设修改 R_i 得到策略 $P'\langle R_1',\cdots,R_{i-1}',R_i',R_{i+1}',\cdots,R_n'\rangle$，用 g 表示策略$\langle R_{i+1},R_{i+2},\cdots,R_n\rangle$，对于任意请求 q，有如下四种情况：

1)假如 $q \in R(R_i,P) \cap R(R_i',P')$，则 $p(q)=r_i.D$ 且 $p'(q)=p(q)$；

2)假如 $q \in R(R_i,P)-R(R_i',P')$，则 $p(q)=r_i.D$ 且 $p'(q)=g(q)$；

3)假如 $q \in R(R_i',P')-R(R_i,P)$，则 $p(q)=g(q)$ 且 $p'(q)=r_i'.D$；

4)假如 $q \notin R(R_i,P) \cup R(R_i',P')$，则 $p(q)=p'(q)$，P 和 P' 对请求 q 有相同结论。

3　变更分析算法

根据上一章变更分析的定理，本章讨论相应的变更分析算法。每种算法的输出为变更分析的结果，表示为：

〈请求中各元素取值区间〉→〈变更前结果〉VS〈变更后结果〉

3.1　规则删除

为了得到删除 R_i 的变更影响，首先计算匹配集合 $R(R_i,P)$。匹配集合可以被表示成一系列不相交的规则$\{e_1,e_2,\cdots,e_k\}$[6]。这一系列不相交的规则也可以称为 R_i 的有效集合。对于任意规则 R 有效集合满足以下三个条件：1)$R(R,P)=\bigcup_{i=1}^{k} M(e_i)$；2)$M(r_i') \cap M(r_j')=\varnothing(1 \leqslant i<j \leqslant k)$；3)每个 e_i 和 R 有相同结论。

R_1、R_2 和 R_3 的有效集合分别表示为 E_1、E_2 和 E_3。

三条规则的有效表达形式：

$$E_1: S \in [0,1] \wedge R \in [0,0] \wedge A \in [0,0] \rightarrow \text{Deny}$$

$$E_2: \begin{cases} S \in [1,1] \wedge R \in [0,0] \wedge A \in [1,1] \rightarrow \text{Permit} \\ S \in [1,1] \wedge R \in [1,1] \wedge A \in [0,1] \rightarrow \text{Permit} \\ S \in [2,3] \wedge R \in [0,1] \wedge A \in [0,1] \rightarrow \text{Permit} \end{cases}$$

$$E_3: S \in [0,0] \wedge R \in [1,1] \wedge A \in [0,1] \rightarrow \text{Permit}$$

假设删除 R_1 后求变更影响。根据定理1，问题可以转换成求满足 E_1 但被只含有$\langle R_2,R_3\rangle$ 的策略 P' 拒绝的请求结果。P' 转换成不相交的等价策略，此策略包含如下规则：

$R_1': S \in [1,3] \wedge R \in [0,1] \wedge A \in [0,1] \rightarrow$Permit

$R_2': S \in [0,0] \wedge R \in [1,1] \wedge A \in [0,1] \rightarrow$Permit

$R_\infty': S \in [0,3] \wedge R \in [0,1] \wedge A \in [0,1] \rightarrow$Deny

为了更直观说明问题，引入一个树形结构来描述等价策略的规则，非叶子节点 S、R、A 分别代表主体、资源和动作，树的边代表每个元素的取值区间。叶子节点为符合不相交等价策略中此区间值的规则结论。如图1所示。

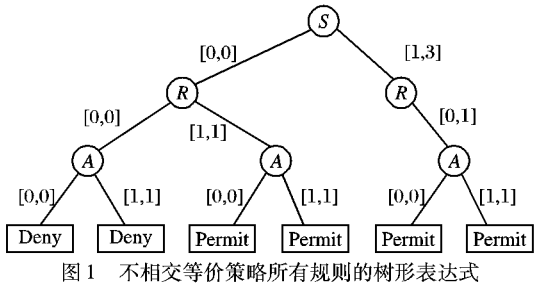

图1　不相交等价策略所有规则的树形表达式

遍历图1中从顶点到叶子节点路径，对于满足 $R_1: S \in$

$[0,1] \wedge R \in [0,0] \wedge A \in [0,0]$ 的所有请求,在删除 R_1 前这些请求的结论为 Deny,删除 R_1 后属于 $S \in [0,0] \wedge R \in [0,0] \wedge A \in [0,0]$ 的请求结果仍然为 Deny,但属于 $S \in [1,1] \wedge R \in [0,0] \wedge A \in [0,0]$ 的请求结果变为 Permit。输出删除规则 R_1 变更分析结果如下:

$S \in [0,0] \wedge R \in [0,0] \wedge A \in [0,0] \rightarrow$
Deny VS Deny

$S \in [1,1] \wedge R \in [0,0] \wedge A \in [0,0] \rightarrow$
Deny VS Permit

3.2　规则插入

根据定理 2,计算规则插入的影响类似于规则删除。若策略 P 包含 n 条规则 $\langle R_1, R_2, \cdots, R_n \rangle$,假设插入规则 R 得到策略 $P' \langle R_1, \cdots, R_{i-1}, R_{i+1}, \cdots, R_n \rangle$,为计算插入的影响,首先计算规则 R 在 P' 中的有限集合。然后构造 $\langle R_{i+1}, R_{i+2}, \cdots, R_n \rangle$ 类似如图 1 的树形结构($\langle R_{i+1}, R_{i+2}, \cdots, R_n \rangle$ 也可以构成一个独立策略,因为最后一条规则匹配所有前面规则未曾匹配的请求)。最后遍历树形结构检查哪一个路径修改前后的结论不同然后得知变更分析结果。

3.3　规则修改

根据定理 3,计算策略中规则 R_i 修改成 R_i' 后的变更影响必需先计算 $R(R_i,P) \cap R(R_i',P')$ 和 $R(R_i,P) - R(R_i',P')$。

接下来讨论怎么计算它们。给定两个匹配集合 R_a 和 R_b 分别可表示成两个有效集合 $\{e_1, e_2, \cdots, e_m\}$ 和 $\{\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n\}$。可得:

$$R_a \cap R_b = \bigcup_{i=1}^{m} \bigcup_{j=1}^{l} (M(e_i) \cap M(\varepsilon_j))$$

其中 $M(e_i) \cap M(\varepsilon_j)$ 可由 $(Se_i \cap S\varepsilon_j) \wedge (Re_i \cap R\varepsilon_j) \wedge (Ae_i \cap A\varepsilon_j)$ 求得,可以看做是规则 e_i 和 ε_j 在主体、资源和动作分别的交集。

给定两个匹配集合 R_a 和 R_b 分别可表示成两个有效集合 $\{e_1, e_2, \cdots, e_m\}$ 和 $\{\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n\}$,$R_\infty$ 为完整性说明规则。$R_a - R_b$ 为策略 $\{e_1, \cdots, e_m, \varepsilon_1, \cdots, \varepsilon_n, R_\infty\}$ 中每一个 e_i 的有效集合的并集。

所以规则修改的影响可以用以下步骤计算。

1)计算 R_i 在 P,R_i' 在 P' 中的有效集合。

2)假如 R_i 和 R_i' 有相同结论,直接跳到第 3) 步。若结论不同,计算 $R(R_i,P) \cap R(R_i',P')$,如果 $R(R_i,P) \cap R(R_i',P') \neq \varnothing$,输出变更分析结果。

3)计算 $R(R_i,P) - R(R_i',P')$。

4)由规则 $\langle R_{i+1}, \cdots, R_n \rangle$ 构造树图。遍历树图,检查每个路径的结论是否合变更前冲突,若有则输出变更分析结果。

4　算法实现

算法采用 Java 语言实现了一个原型系统,策略评估引擎采用 SUN PDP,设置策略导入功能以及统一变更输入界面,针对规则删除、插入、修改的变更分析算法采用独立的功能模块,并把结果输出到统一界面。测试平台采用一台 2.56 GHz 的 AMD 双核 CPU、8 GB 内存并运行 Windows Server 2003 操作系统的 PC 机。对一个有 1 000 条规则的策略中一条规则的变更分析运行 20 次,平均耗时 100 ms。

5　结语

本文分析了 XACML 策略定制中变更影响的各种可能性,归纳出变更分析理论基础,并提出规则删除、规则插入和规则修改的变更分析算法,对可能的变更结果做出预测输出分析结果,为系统管理员提供直观的决策参考,降低了策略定制中出现安全纰漏的几率。相关方法结论还可以应用到其他安全隐私策略语言。实验证明本文算法在保证正确准确情况下耗时短,可应用于各种策略语言。

参考文献:

[1] ANDERSON A. A comparison of two privacy policy languages: EPAL and XACML[C]// Proceedings of the 3rd ACM Workshop on Secure Web Services. New York: ACM, 2006: 53 - 60.

[2] 王雅哲,冯登国. 一种 XACML 规则冲突及冗余分析方法[J]. 计算机学报,2009,32(3):516 - 530.

[3] PIETRO M, BRUNO C, SWAMINATHAN S, *et al*. XACML policy integration algorithms[J]. ACM Transactions on Information and System Security, 2008, 11(1):1 - 29.

[4] RYDER B, TIP F. Change impact analysis for object-oriented programs[C]// Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. New York: ACM, 2001: 46 - 53.

[5] LIU A, CHEN F, HWANG J, *et al*. Xengine: A fast and scalable XACML policy evaluation engine[C]// Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York: ACM, 2008: 265 - 276.

[6] LIU A, GOUDA M. Complete redundancy detection in firewalls [C]// Proceedings of the 19th Annual IFIP Conference on Data and Applications Security. Berlin: Springer-Verlag, 2005: 196 - 209.

(上接第 84 页)

[2] COELLO C A C, PULIDO G T, LECHUGA M S. Handling multiple objectives with particle swarm optimization [J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 256 - 279.

[3] LEONG W-F, YEN G G. PSO-based multiobjective optimization with dynamic population size and adaptive local archives[J]. IEEE Transactions on Systems, Man, and Cybernetics——Part B: Cybernetics, 2008, 38(5): 1270 - 1293.

[4] YEN G G, LEONG W-F. Dynamic multiple swarms in multiobjective particle swarm optimization [J]. IEEE Transactions on Systems, Man, and Cybernetics——Part A: Systems and Humans, 2009, 39(4): 890 - 911.

[5] MARTINEZ S Z, COELLO C A C. Hybridizing an evolutionary algorithm with mathematical programming techniques for multi-objective optimization [C]// Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference. New York: ACM, 2008: 769 - 770.

[6] STACEY A, JANCIC M, GRUNDY I. Particle swarm optimization with mutation[C]// CEC '03: Proceedings of IEEE Congress on Evolutionary Computation. Washington, DC: IEEE, 2003: 1425 - 1430.

[7] DEB K, GOYAL M. A combined genetic adaptive search (GeneAS) for engineering design [J]. Computer Science and Informatics, 1996, 26(4):30 - 45.

[8] DEB K, PRATAP A, AAGRWALl S, *et al*. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002,6(2): 182 - 197.