基于多值属性分量的XACML策略匹配算法

李冬辉 张 斌 费晓飞 刘 洋
(解放军信息工程大学 郑州 450001)

摘 要 针对多值属性分量的XACML策略和策略请求之间的匹配需求,分析多值属性策略匹配中策略规则与请求匹配时两者的对应属性关系,依据属性之间的包含关系和权限蕴含关系,给出3个关于策略匹配的定理并加以证明。根据策略匹配的定理,提出多值属性匹配算法。最后进行实验验证,结果表明该算法能够提高多值属性分量策略的匹配效率。

关键词 XACML,策略,多值属性,匹配定理,匹配算法
中图法分类号 TP393.08 **文献标识码** A

Algorithm of Matching to XACML-Policy Based on Component of Multi-valued Attribute

LI Dong-hui ZHANG Bin FEI Xiao-fei LIU Yang
(PLA Information Engineering University, Zhengzhou 450001, China)

Abstract Aiming at the demand of matching between the XACML policy and request based on the multi-valued attribute, this paper analyzed the corresponding attribute relationships between policy rule and the request. Three related theorems were proposed and proved based on the relationship between attributes implication and permissions implication. According to the three policy matching theorems, a matching algorithm was put forward. Finally, several experiments show that the algorithm enhances the matching efficiency.

Keywords XACML, Policy, Multi-valued attribute, Matching theorem, Matching algorithm

1 引言

随着各域组织间的业务协作、资源共享等新一代网络体系结构的出现,分布式资源共享、Web服务、域间协作等新兴业务需要制定大量的XACML策略来对资源进行细粒度访问控制[1-3]。在开放式的计算环境下,策略需要考虑的属性种类越来越多[4],通过对一种属性分量(主体属性、客体属性等)的多值描述能够实现细粒度的访问控制,例如策略中描述一个主体可能需要包含多个属性信息(职位、ID、年龄等)。

文献[5,6]给出了多值属性分量的策略描述和形式化描述,但没有提出相应的策略匹配规则和策略匹配算法。文献[7,8]提出了两种策略匹配算法,其能够提高策略的匹配效率,但没有考虑多值属性分量的XACML策略匹配。文献[9]提出的Melcoe PDP将策略存储到XML类型数据库中,根据原来给定的属性匹配列表减少策略检索空间,但其优化方法过于依赖数据库本身处理XML的效率,另一方面降低了XACML的灵活性。

本文通过深入分析多值属性策略匹配中策略规则与请求匹配时两者的对应属性关系、策略规则中属性个数n和请求中属性个数m对策略规则匹配的影响,并结合属性之间的包含关系与权限蕴含规则,给出基于多值属性分量的策略匹配定理并加以证明,然后根据策略匹配的定理提出多值属性匹配算法,最后通过实验验证该算法的性能。

2 多值属性分量策略匹配定理

2.1 多值XACML策略规则的形式化描述

策略中的目标(Target)[1]元素定义了策略规则适用的主体、客体、动作和环境集合。在实际的匹配过程中,规则中的Target元素是最终与访问请求进行匹配的最小策略单元。以下是对策略规则Target元素的形式化描述。

定义1 规则中Target元素和请求中的属性及其集合的形式化表示如表1所列。

表1 形式化描述

名称	形式化表示
目标中主、客体属性	SA_i、OA_i
目标中主、客体属性集合	$\bigcup_{i=1}^{n_1} SA_i$、$\bigcup_{i=1}^{n_2} OA_i$
目标中主、客体属性值	SA_i_val、OA_i_val
目标中主、客体属性值集合	$\bigcup_{i=1}^{n_1} SA_i\_val$、$\bigcup_{i=1}^{n_2} OA_i\_val$
目标中动作属性值	AC_T
请求中主、客体属性	SA_j、OA_j
请求中主、客体属性集合	$\bigcup_{j=1}^{m_1} SA_j$、$\bigcup_{j=1}^{m_2} OA_j$
请求中主、客体属性值	SA_j_val、OA_j_val
请求中主、客体属性值集合	$\bigcup_{j=1}^{m_1} SA_j\_val$、$\bigcup_{j=1}^{m_2} OA_j\_val$
请求中动作属性值	AC_R

到稿日期:2013-08-26 返修日期:2013-10-28 本文受国家863计划(2006AA01Z457)资助。
李冬辉(1988-),男,硕士生,主要研究方向为网络访问控制,E-mail:dhlee999@126.com;张 斌(1969-),男,博士,教授,主要研究方向为网络信息安全;费晓飞(1977-),男,博士,主要研究方向为SOA安全;刘 洋(1980-),男,博士生,主要研究方向为SOA安全。

定义 2 策略规则匹配满足状态。若规则 Rule1 中 Target 元素的主体属性、客体属性和动作与请求中的主体属性、客体属性和动作能够满足匹配规则，那么此规则与请求匹配成功，主体属性匹配成功表示为$(State_SA)^{match}$。匹配失败则表示为$(State_SA)^{notmatch}$。

XACML 策略与请求的匹配是依据两者的属性是否满足匹配规则来决定的。规则中的主体属性和请求中的属性是同一种属性信息，表示为 $SA_i=SA_j$。主体属性之间的包含关系蕴含着权限关系，例如一个属性值为 Teacher 的主体拥有对学校教育网的浏览权限，那么属性值是 English teacher 的主体也同样拥有该权限，但属性值是 English teacher 的主体拥有某项权限则不能推导出属性值是 Teacher 的主体也拥有该权限，那么称属性值 Teacher 包含 English teacher，表示为 $SA_1\_val \vdash\!\!\rightarrow SA_2\_val$，蕴含的权限关系[10]是：具有属性 SA_1_val 的主体的所有权限都会被具有属性 SA_2_val 的主体所继承。同理，客体属性的描述中也存在包含关系。由此可知，当请求中的属性描述包含于策略规则中的属性描述时，该策略规则能够匹配该请求。

2.2 多值 XACML 策略匹配定理

对于多值属性的规则和请求匹配规则，规则中的主体属性个数表示为 n，请求中主体属性个数表示为 m。本文将策略的匹配规则分为以下 3 种情况进行描述(客体、动作属性匹配定理同理)。

定理 1 $n=m$ 时，匹配成功的必要条件为：

$$(State\_SA)^{match}\Rightarrow\begin{cases}\bigcup_{i=1}^{n}SA_i=\bigcup_{j=1}^{m}SA_j\\ \bigcup_{i=1}^{n}SA_i\_val \vdash\!\!\rightarrow \bigcup_{j=1}^{m}SA_j\_val\end{cases}$$

$\bigcup_{i=1}^{n}SA_i\neq\bigcup_{j=1}^{m}SA_j$

通过证明该命题的逆否命题进行证明。

已知：$n=m$，$\bigcup_{i=1}^{n}SA_i\neq\bigcup_{j=1}^{m}SA_j$，$\bigcup_{i=1}^{n}SA_i\_val \vdash\!\!\rightarrow \bigcup_{j=1}^{m}SA_j\_val$ 不成立。

待证：$(State\_SA_i)^{notmatch}$

证明：$\because n=m$，$\bigcup_{i=1}^{n}SA_i\neq\bigcup_{j=1}^{m}SA_j$

$\therefore$对于任意的 $SA_j\in\bigcup_{j=1}^{m}SA_j$，$\forall SA_i\in\bigcup_{i=1}^{n}SA_i$ 满足 $SA_i\neq SA_j$

$\therefore(State\_SA_i)^{notmatch}$

$\because n=m$，$\bigcup_{i=1}^{n}SA_i\_val \vdash\!\!\rightarrow \bigcup_{j=1}^{m}SA_j\_val$ 不成立

$\therefore$对于任意的 $SA_j\in\bigcup_{j=1}^{m}SA_j\_val$，$\forall SA_i\_val\in\bigcup_{i=1}^{n}SA_i\_val$ 满足$(SA_i_val \vdash\!\!\rightarrow SA_j_val)$

$\therefore(State\_SA_i)^{notmatch}$

证毕。

定理 2 $n<m$ 时，匹配成功的必要条件为：

$$(State\_SA)^{match}\Rightarrow\begin{cases}\bigcup_{i=1}^{n}SA_i\subseteq\bigcup_{j=1}^{m}SA_j\\ \bigcup_{i=1}^{n}SA_i\_val \vdash\!\!\rightarrow \bigcup_{j=1}^{m}SA_j\_val\end{cases}$$

通过证明该命题的逆否命题进行证明。

已知：$n<m$，$\bigcup_{i=1}^{n}SA_i\not\subseteq\bigcup_{j=1}^{m}SA_j$，$\bigcup_{i=1}^{n}SA_i\_val \vdash\!\!\rightarrow \bigcup_{j=1}^{m}SA_j\_val$ 不成立。

待证：$(State\_SA_i)^{notmatch}$

证明：$\because n<m$，$\bigcup_{i=1}^{n}SA_i\not\subseteq\bigcup_{j=1}^{m}SA_j$

$\therefore\forall SA_i\in\bigcup_{i=1}^{n}SA_i$，使 $SA_i\cap\bigcup_{j=1}^{m}SA_j=\varnothing$

即存在 $SA_i\neq SA_j$

$\therefore(State\_SA_i)^{notmatch}$

$\because n<m$，$\bigcup_{i=1}^{n}SA_i\_val \vdash\!\!\rightarrow \bigcup_{j=1}^{m}SA_j\_val$ 不成立

同理可得$(State_SA_i)^{notmatch}$

证毕。

定理 3 $n>m\Rightarrow(State\_SA_i)^{notmatch}$。

由于$\forall SA_i\in\bigcup_{i=1}^{n}SA_i$ 满足 $SA_i\neq SA_j$ 且 $SA_i\_val \vdash\!\!\rightarrow SA_j\_val$ 不成立，匹配失败，得证。

3 多值属性策略匹配算法

3.1 多值属性策略匹配算法

由于策略集是树状结构，可以将策略集表示为策略树。图 1 所示为一个策略树。

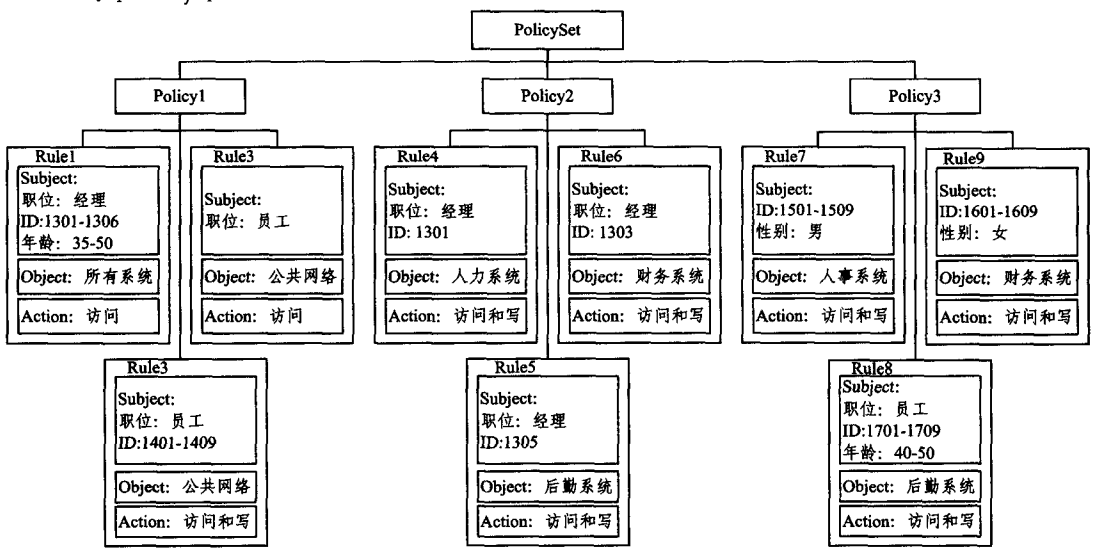

图 1 策略树

在 XACML 策略中，规则是策略的最基本单元，在规则 Rule 中包含了用于匹配策略请求的 Target 元素，Target 元素包含主体、客体和动作 3 个元素。这 3 个元素也可以表示为树形结构，其中 Rule1 中主体的属性结构表示为树形结构，如图 2 所示。

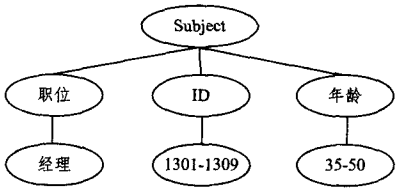

图 2　Rule1 的主体属性树

算法思想：根据策略匹配定理，给出策略匹配的一般过程：

(1)比较规则 Target 中主体属性数量 n 和请求中主体属性的数量 m，如果 $n \leqslant m$，根据定理 1 和定理 2 可知，需要保留该规则，然后依据定理内容需进行下一步的判断。如果 $n > m$，由定理 3 可知，该规则不能与策略请求匹配，则直接删除该规则。

(2)将请求中的主体属性与策略规则中的主体属性进行比较，若满足匹配定理 1 和定理 2 中定义的条件，则保留该规则，否则删除该规则。

(3)将请求中的主体属性值与规则中的主体属性值进行比较，若满足匹配定理中定义的条件，则保留该规则，否则删除该规则。

(4)按照上述方法对元素〈Resource〉和〈Action〉进行匹配，每次匹配的规则集都是经过优化后形成的新规则集合。

策略匹配算法中的主体属性匹配算法如下所示(客体和动作的匹配算法同理)：

Input：策略中规则的主体属性树：$tree_1$；请求中的主体属性树：$tree_2$。
Output：Match or Notmatch
Begin
Array T_1 = Trial($tree_1$)；
t_1 = T_1. get RootNode()；
$\bigcup_{i=1}^{n} SA_i$ = T_1. get ChildrenNode(t_1)，
$\bigcup_{i=1}^{n} SA_i\_val$ = T_1. get LeviesNode(t_1)；
Array T_2 = Trial($tree_2$)
t_2 = T_2. get RootNode()；
$\bigcup_{j=1}^{m} SA_j$ = T_2. get ChildrenNode(t_2)，
$\bigcup_{j=1}^{m} SA_j\_val$ = T_2. get LeviesNode(t_2)；
if(n=m)；
  if($\bigcup_{i=1}^{n} SA_i = \bigcup_{j=1}^{m} SA_j$)；
    if($\bigcup_{i=1}^{n} SA_i\_val \mid\!\rightarrow \bigcup_{j=1}^{m} SA_j\_val$)；
      return Match；
    else return Notmatch；
  else return Notmatch；
elseif(n<m)；
  if(($\bigcup_{i=1}^{n} SA_i \subseteq \bigcup_{j=1}^{m} SA_j$)；
    if(($\forall SA_i\_val, \exists SA_j\_val$) $\wedge$ ($SA_i\_val \mid\!\rightarrow SA_j\_val$))；
      return Match；
    else return Notmatch；
  else return Notmatch；
else(n>m)；
    return Notmatch；
end

假设有策略请求如下所示：

```
〈request〉
  〈target〉
    〈Subjects〉
      〈Subject〉职位：员工〈/Subject〉
      〈Subject〉ID：1405〈/Subject〉
    〈/Subjects〉
      〈Resource〉公共网络〈/Resource〉
      〈Action〉 read and write 〈/Action〉
  〈/target〉
〈/request〉
```

下面对图 1 所示的策略集按照匹配算法进行逐步优化。此时策略请求中的主体属性数量 $m=2$。Rule1 的 Target 中主体属性数量 $n=3$，根据策略匹配算法，$n>m$，此规则是无关规则，删除该规则。依次比较余下所有规则中的主体属性数量，Rule8 的主体属性数量也大于请求中主体属性数量，按照匹配算法对策略集进行裁剪，结果如图 3(a)所示。对比主体属性元素时，按照匹配定理，Rule7 和 Rule9 中的(ID，年龄)与请求中的(职位，ID)不匹配，删除这两个规则，策略集裁剪结果如图 3(b)所示。比较主体属性值时，Rule4、Rule5 和 Rule6 与请求不匹配，删除这 3 项规则，策略集裁剪结果如图 3(c)所示。对比元素〈Resource〉和〈Action〉后可知，只有 Rule2 能够匹配该请求，如图 3(d)所示。

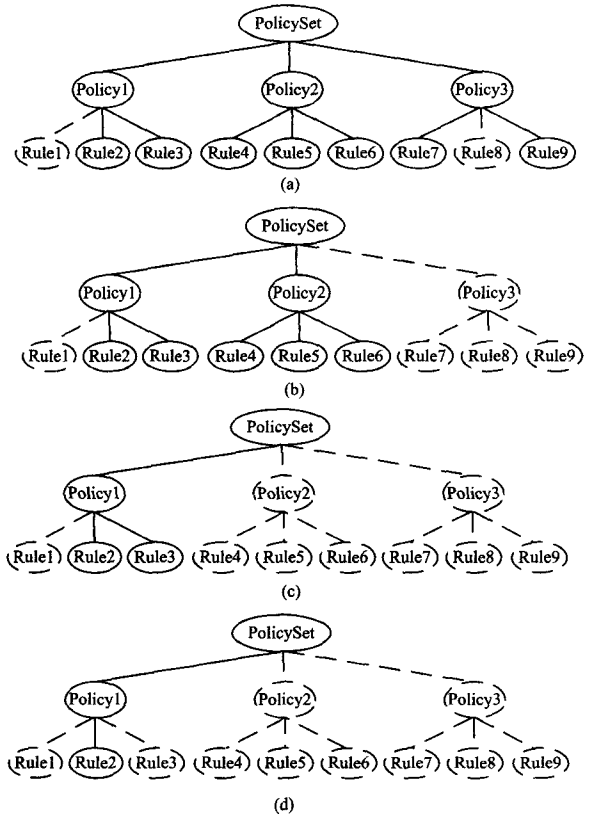

图 3　策略优化过程

假设策略集中有 k 个规则节点，每个规则节点中的主体

属性匹配时间 t_1、主体属性值匹配时间 t_2、客体属性匹配时间 t_3、客体属性值匹配时间 t_4 和动作的匹配时间 t_5 通过匹配算法对原策略集中的无关规则进行裁剪，则原有方法的匹配时间是 $(t_1+t_2+t_3+t_4+t_5)k$，通过每次判定将无关策略删除后分别为 k_1, k_2, k_3, k_4 和 k_5 ($k \geqslant k_1 \geqslant k_2 \geqslant k_3 \geqslant k_4 \geqslant k_5$)，匹配时间是 $(k_1t_1+k_2t_2+k_3t_3+k_4t_4+k_5t_5) \leqslant (t_1+t_2+t_3+t_4+t_5)k$，由此可知，根据匹配算法删除无效策略后，能够提高策略匹配的效率。

3.2 策略匹配算法的时间复杂度分析

经分析可知，匹配算法的时间复杂度与策略树的节点流大小成线性关系，策略规则与请求中的一个主体属性分量进行匹配时的时间复杂度为 $L=O(T_1+T_2+\cdots+T_D)$，当策略请求中主体属性分量值为 m 时，时间复杂度为 $O(L^m)$，经过每一步的策略集裁剪可知：$L_m \leqslant L_{m-1} \leqslant \cdots \leqslant L_1 = L$，其时间复杂度是 $O(L_1 \times L_2 \times \cdots \times L_m) \leqslant O(L^m)$。因此，本算法的时间复杂度较低，提高了匹配的效率。

3.3 多值属性匹配算法的实验分析

根据对策略匹配的分析可知，策略规则数量和请求的主体属性约束的个数 m 是影响策略匹配效率的两个因素。因此，从这两个方面来测试提出的多值属性匹配算法的性能。测试的策略规则数量分别是{50、100、150、200、300}，策略请求中主体属性约束的个数 m={1、2、3、4、5}。用 UMU XACML Editor 1.3.2 编写策略，采用 SUNX-ACML-1.2[11] 的策略评估引擎实现访问控制中对请求的决策。

用 Policy Finder.java 实现策略查找功能，通过 PolicyFinderResult=new PolicySet()实现策略的逐步优化，保证每一步的策略集都是经过前一步优化裁剪的。然后由 TestPDP.java 根据匹配到的策略进行权限决策，并返回策略决策结果。使用 TimeRangFunction.java 可以对策略决策时间进行监控。

实验选择 Sun PDP、Melcoe PDP、文献[8]提出的优先级合并算法的 Sun PDP 与本文提出的多值属性匹配算法进行性能比较。

测试中的请求主体属性分量 m 为定值 3，策略规则数量分别是{50、100、150、200、300}。实验结果如图 4 所示，从实验结果可以看出，使用多值属性匹配算法能够提高策略的决策效率。

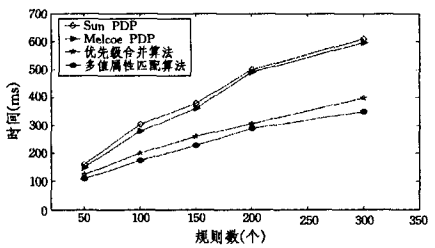

图 4 规则数量不同时策略匹配性能对比

假定其策略规则数量为定值 100，检测策略请求中主体属性约束的个数 m={1、2、3、4、5}时对策略匹配速度的影响，实验结果如图 5 所示。

根据实验结果可知，在策略规则一定时，策略请求的约束 m 值越大，匹配时需要的时间就越长。并且当请求的主体属性约束 m 值越小，本文提出的多值属性匹配算法对策略匹配效率的提升越明显。

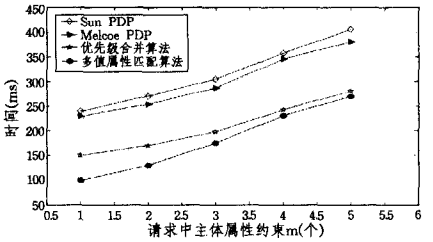

图 5 m 不同时策略匹配性能分析

结束语 本文针对多值属性分量的策略和请求之间匹配的需求，给出基于多值属性分量的 XACML 策略匹配的 3 个定理并加以证明，该匹配定理能够满足多值属性分量的策略和请求之间的匹配需求。并根据策略匹配定理，提出了一种针对多值属性分量策略匹配的属性匹配算法，其通过逐步优化，删除策略中无关规则节点，减少策略匹配的次数，提高访问控制策略决策效率。

参 考 文 献

[1] OASIS. Extensible Access Control Markup Language (XACML) V3.0[EB/OL]. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cd-1-en.pdf. April, 2009

[2] 李晓峰，冯登国，何永忠. XACML Admin 中的策略预处理研究[J]. 计算机研究与发展，2007，44(5)：730-736

[3] Bertolino A, Daoudagh S, Lonetti F. Automatic XACML requests generation for policy Testing[C]// 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012, 185：842-849

[4] Liu A X, Fei Chen. Designing Fast and Scalable XACML Policy Evaluation Engines[J]. IEEE Transactions on Computers, 2011, 12(60)：1802-1816

[5] Butler B, Jennings B, Botvich D. An experimental testbed to predict the performance of XACML Policy Decision Points [C]// 12th IFIP/IEEE International Symposium on Integrated Network Management. 2011：353-360

[6] Marouf S, Shehab M. Statistics and Clustering Based Framework for Efficient XACML Policy Evaluation [C]// IEEE International Symposium on Policy for Distributed Systems and Networks. 2009, 36：118-125

[7] 谢辉. 基于 UCON 改进模型的授权管理关键技术研究[D]. 郑州：解放军信息工程大学，2009：53-58

[8] 陈伟鹏，王娜娜. 基于 XACML 的策略评估优化技术的研究[J]. 计算机应用研究，2013，30(3)：900-905

[9] Jajodia S, Samarati P, Subrahmanian V S. A logical language for expressing authorizations [C]// Proceedings of the 1997 IEEE Symposium on Security and Privacy. Los Alamitos, California, USA, 1997：31-42

[10] 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)：516-527

[11] Sun Microsystems Inc. Sun XACML Policy Engine[OL]. http://sunxacml.sourceforge.net/guide.html