# 一种基于属性与或矩阵和类型分析的 XACML 策略查询方法

韩道军[1,2] 原万里[2] 段晓宇[2] 张 磊[2]
(河南大学数据与知识工程研究所 河南 开封 475004)[1]
(河南大学计算机与信息工程学院 河南 开封 475004)[2]

**摘 要** 访问控制策略的描述与执行是信息系统资源保护的一种重要方式,影响到系统的业务化运行。针对目前评估效率较低的问题,研究人员提出了基于属性缓存和重排序等策略的评估方法,该方法提高了策略的评估效率,但尚未解决策略评估需要遍历所有相关规则的问题。针对此问题,在分析 XACML(eRxtensible Access Control Markup Language)描述特点的基础上,利用属性与或矩阵和类型分析,提出一种基于属性与或矩阵和类型分析的 XACML 策略查询方法,以减少策略评估实施时的规则匹配数量。该方法修改了现有 Context Handler 的处理过程,增加了一个访问控制规则匹配预处理环节,在该环节中计算得出每个规则属性的区分度,利用区分度和属性与或矩阵筛选掉与当前访问控制请求无关的规则,然后对筛选后的规则集合进行匹配,提高策略评估效率。最后通过实验验证了所提方法的有效性。

**关键词** XACML,属性与或矩阵,区分度,类型分析
**中图法分类号** TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.09.037

## XACML Policy Query Method Based on Attribute And/Or Matrix and Type Analysis

HAN Dao-jun[1,2] YUAN Wan-li[2] DUAN Xiao-yu[2] ZHANG Lei[2]
(Institute of Data and Knowledge Engineering, Henan University, Kaifeng, Henan 475004, China)[1]
(School of Computer and Information Engineering, Henan University, Kaifeng, Henan 475004, China)[2]

**Abstract** The description and execution of access control policy is an important way of information resource protection, which affects system's operational running. In view of the poor efficiency of evaluation, some researchers have proposed the policy evaluation methods based on attribute cache and reordering, which improve the efficiency of policy evaluation, but they still fail to solve the problem that the policy evaluation needs to traverse all relevant rules. To focus on this problem, after the analysis about the characteristics of the XACML policy description, a XACML policy query method based on attribute and/or matrix and type analysis was proposed in this paper, which can reduce the number of matching during policy evaluation. This method modifies the processing of the existing Context Handler, and adds a preprocessing phase which will match access control rule. During the preprocessing phase, the discriminations are calculated for each rule attributes. The irrelative rules for current access control request can be filtered by the attribute and/or matrix and the discriminations. The proposed method can improve the efficiency of policy evaluation by matching the filtered rule set. Experimental results verify its efficiency.

**Keywords** XACML, Attribute and/or matrix, Discrimination, Type analysis

## 1 引言

在信息安全问题较为突出的情况下,国际标准组织(OASIS)[1]定义的 5 项标准安全服务之一的访问控制作为该问题的有效解决方式,被广泛地应用于各类信息系统,其目的是使受保护的资源能被合理、受控地使用。访问控制策略描述语言是访问控制的一个重要组成部分,是资源受控管理的一种表达方式。与其他访问控制策略描述语言相比,XACML 具有通用性、可扩展性和功能强大的特点,已成为许多大型企业应用和商业化产品实现安全、授权功能的一种标准语言。

随着系统用户和资源数量的不断增加,访问控制策略中包含的规则数越来越多,结构越来越复杂,策略评估效率已成为制约系统可用性的关键瓶颈,此类问题引起了众多研究人员的关注[2-7]。在文献[2]提出的基于统计分析优化的高性能

到稿日期:2017-08-08 返修日期:2017-11-21 本文受国家自然科学基金资助项目(61272545,61402149),河南省科技攻关计划基金资助项目(142102210390),河南省教育厅科技攻关计划基金资助项目(14A520026),河南省博士后科研项目(2015036)资助。
**韩道军**(1979—),男,博士,副教授,主要研究方向为形式概念分析、空间数据处理、信息安全;**原万里**(1993—),男,硕士生,主要研究方向为访问控制技术,E-mail:2293978452@qq.com(通信作者);**段晓宇**(1992—),女,硕士生,主要研究方向为图像处理;**张 磊**(1981—),男,博士,讲师,主要研究方向为空间数据处理、信息安全。

XACML 策略评估引擎中，利用统计分析机制为频繁调用的属性、策略以及请求结果对建立缓存，解决了文献[3]中使用了缓存但缓存内容不一定被调用得最频繁的问题。但是，在实际应用中，为了保证频繁调用的规则始终在缓存中，该引擎需要在每次访问时做统计分析，以对规则进行调整排序，维护代价较高。文献[5]提出了一种自适应的策略/规则重排序方法，但其同样需要持续地计算优先级来满足“少部分策略/规则处理大部分请求”。基于这种情况，本文提出了一种基于属性与或矩阵和类型分析的 XACML 策略属性提取方法。该方法首先根据概率统计的数据分析方式修改现有的 Context Handler 的处理过程，增加了一个预处理过程，该过程把所有规则中的属性存放在属性与或矩阵中；然后根据属性的类型统计每一个属性的区分度，生成一个区分度集。当用户发送访问请求时，系统根据该区分度集来确定对规则集过滤时使用的属性的先后顺序，并对无效规则进行过滤，从而减少规则集的访问次数，提高系统的决策效率。与已有方法相比，本文方法具有以下特点：1）只有规则发生改变时才需要进行统计分析，计算量较小，代价较低；2）规则匹配速度较快，通过优化匹配条件减少了大量无效规则匹配过程中的计算，提高了访问控制策略的请求应答效率。

## 2　XACML 策略查询优化方法

针对传统方法中的遍历查询导致的无效规则的多次访问问题，本文提出了基于属性与或矩阵和类型分析的 XACML 策略查询方法。该方法通过将属性与或矩阵和区分度相结合，能够有效解决传统方法中存在的问题，实现策略信息的快速匹配，同时减少访问控制请求的决策响应时间。

### 2.1　策略查询优化方法

为了降低策略评估时的规则匹配数量，本文在上下文处理器（Context Handler）中增加了一个预处理过程。改进的 Context Handler 如图 1 灰色框中所示，在传统的处理器[8]的基础上，增加了一个虚线框所示的预处理过程。该过程的步骤及用途如下：1）将所有策略包含的规则中的属性存放在属性与或矩阵中，用于 PDP 生成决策结果时的规则过滤；2）基于属性与或矩阵的数据（规则集合），对规则中的主体、资源和动作等属性进行统计，计算得到各个属性的区分度，利用区分度可以确定在根据属性进行规则过滤时使用属性的先后顺序。

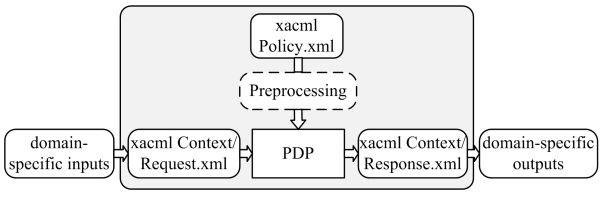


图 1　上下文处理器

Fig. 1　Context Handler

在传统方法中，系统的访问策略缺少预处理过程，PDP 需要读取大量未经分类和分析的访问控制规则，有可能需要多次遍历无效规则（这里的无效规则是指该规则最终不满足访问请求条件），从而造成决策时间过长、规则匹配效率低下（在最坏情况下，需要遍历所有的无效规则后才能给出应答）的问题。图 2 中的预处理过程可在后续规则过滤中优先使用区分度较大的属性对规则集进行过滤，最大程度地保证每次过滤的效果，即一个属性的区分度越大，将有越多的无效规则在属性匹配过程中被过滤掉；并且被过滤的规则不可参与下一次的规则匹配过程，减少了对一条无效规则的重复访问次数。改进后的方法减少了对无效规则在匹配过程中的重复访问，缩短了一次决策的响应时间，提高了访问控制请求的应答效率。

### 2.2　属性与或矩阵

规则的主要要素为：Subject，Resource，Action 和 Environment。属性与或矩阵将每个规则要素结构化，使用属性表示规则要素。通过属性与或矩阵表示规则集，才能够进行类型分析。

**定义 1**（属性与或矩阵）　假定 $M$ 为一个矩阵，$M$ 中的行（$r$）和列（$c$）分别表示为规则以及规则的属性。设对于 $M$ 中的任意两行 $r_i$ 和 $r_j$，$\exists\, c_{ik} \in r_i$，$\exists\, c_{il} \in r_i$，若 $(r_i \vee r_j) \wedge (c_{ik} \wedge c_{il})$ 成立，则称 $M$ 为规则属性与或矩阵，简称属性与或矩阵。

在构建属性与或矩阵时，需要对所有的规则集进行属性提取，即将每条规则的属性放入属性与或矩阵的对应位置。表 1 给出一个属性与或矩阵的示例，其中列包括规则的标识以及规则下的所有属性，一行则表示为一条规则。列与列之间，如 $R_1$ 中的 3 个属性之间是与关系，必须同时满足属性 $A$ 为 $a$，属性 $B$ 为 $b$，属性 $C$ 为 $c$，才是一个完整的 $R_1$。然而，行与行之间，如 $R_1$ 与 $R_2$ 是没有必然联系的或关系。

表 1　属性与或矩阵示例

Table 1　Example of attribute and/or matrix

| 规则 | 属性 $A$ | 属性 $B$ | 属性 $C$ |
|---|---|---|---|
| $R_1$ | $a$ | $b$ | $c$ |
| $R_2$ | $b$ | $a$ | $b$ |
| $R_3$ | $a$ | $c$ | $d$ |

本文使用示例 1 来说明根据每个属性的区分度对属性与或矩阵过滤无效规则能够提高策略评估效率。

示例 1　假设存在以下信息：全校有 1500 名学生，3 个年级（高一、高二、高三），每个年级 10 个班，每个班的人数相等，均为 50 人，男女生比例相同，各占 50%。查询目标：高二（9）班的男生。

根据以上 3 个条件，存在如表 2 所列的 6 种查找方案。

表 2　查找方案

Table 2　Inquiry schemes

| 方案 | 查找顺序 | | |
|---|---|---|---|
| 方案 1 | 高二 | 9 班 | 男 |
| 方案 2 | 高二 | 男 | 9 班 |
| 方案 3 | 9 班 | 高二 | 男 |
| 方案 4 | 9 班 | 男 | 高二 |
| 方案 5 | 男 | 9 班 | 高二 |
| 方案 6 | 男 | 高二 | 9 班 |

使用表 2 提供的方案依次对每个个体（学生）进行查询，当学生的属性与查找的子集属性（高二、9 班、男生）不符时，将该学生剔除。表 2 中的 6 种方案得到的最终结果是一样的，但查询次数是不相同的。

如表 3 所列，方案 3 为最佳的过滤方案，其原因在于每次

过滤时使用的属性在整体中所占比例最小，可以在一次属性遍历查询中最大限度地过滤掉不符合条件的个体。另外，从表 3 中方案 3 和方案 4 的第二次查找次数可看出：符合条件的子集在进行下次遍历时的查询次数将受到前一次过滤结果的影响。若每次查找所需的查询次数最少，则整体次数为最少，即为最佳的查找顺序。

表 3 不同方案的查询次数

Table 3 Inquiry times of different schemes

| 方案 | 查找次数 | | | |
|---|---|---|---|---|
| | 第 1 次查找 | 第 2 次查找 | 第 3 次查找 | 总次数 |
| 方案 1 | 1500 | 500 | 50 | 2050 |
| 方案 2 | 1500 | 500 | 250 | 2250 |
| 方案 3 | 1500 | 150 | 50 | 1700 |
| 方案 4 | 1500 | 150 | 75 | 1725 |
| 方案 5 | 1500 | 750 | 75 | 2325 |
| 方案 6 | 1500 | 750 | 250 | 2500 |

如图 2 所示，每个学生的信息存储在属性与或矩阵中，按照表 2 中的查找方案 3 实现对数据集的过滤。同样地，在使用属性与或矩阵对规则集进行过滤的过程中，需要事先确定过滤无效规则时的最优过滤次序，最优过滤次序的确定依赖于区分度的大小。如示例 1 所示，与‘9 班’这一属性相符的个体占比最小，同时它是区分度最大的一个属性，过滤次序为优先的。

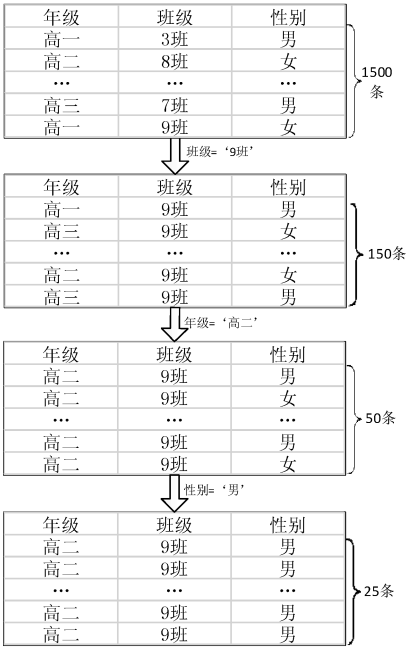


图 2 信息过滤过程示意图

Fig. 2 Diagram of information filtering process

通过以上示例 1 可以发现，通过使用区分度和属性与或矩阵，可以在属性的匹配中有效减少规则集的访问次数，从而提高决策生成的效率。

### 2.3 区分度的计算方法

在 2.2 节的示例 1 中，3 个属性均可以看作枚举类型。在实际中，属性的类型可分为枚举、数值、日期、字符串等，应根据属性类型的不同采用相应的方法计算区分度。区分度表示为：

$$Dis(A_n)=\frac{c^{(n)}}{c_{n_0}} \tag{1}$$

其中，属性 $c^{(n)}$ 为第 $n$ 个属性 $A_n$ 中属性值的总个数，$c_{n_0}$ 为事先划分的一个子区间包含的属性值的个数，针对不同的属性类型采用相应的划分子区间的方法。一般地，其他的属性类型可转化为枚举、数值、日期、字符串 4 种主要的属性类型，并按照以下方法来计算区分度。

1)枚举。设枚举类型的属性 $A_1$ 有 $N$ 个枚举值，总记录条数为 $S$，则每个枚举值的平均匹配记录为 $S/N$，$N$ 越大，$S/N$ 显然就越小；极端情况下，$S/N$ 为小数和整数。鉴于此，可以根据属性 $A_1$ 枚举值来划分子区间。一个枚举值可视为一个子区间，因此在枚举类型属性 $A_1$ 中，区分度为：

$$Dis(A_1)=\frac{c^{(1)}}{c_{n_0}}=N$$

$Dis$ 即为枚举值的个数。例如：属性 $Gender=\{$男，女$\}$，$Gender$ 取值仅限于‘男’‘女’两个，则区分度 $Dis(Gender)=2$。

2)数值类型。假设 $A_2$ 为数值类型属性，无论理论取值的范围有无限制，在有限的值中都可得到最大值和最小值。设取值范围为$\{x\mid a<x<b,x\in N^*\}$，将取值区间分为 $n_0$ 个子区间，对应的子区间分别为$(a,a+\frac{b-a}{n_0})$，$(a+\frac{b-a}{n_0},a+2\cdot\frac{b-a}{n_0})$，…，$((n_0-1)\cdot\frac{b-a}{n_0},b)$，$A_2$ 的属性值落在每个子区间内的策略个数记为：

$$c_1=GSN(a<x_1<a+\frac{b-a}{n_0})$$

$$c_2=GSN(a+\frac{b-a}{n_0}<x_2<a+2\cdot\frac{b-a}{n_0})$$

…

$$c_{n0}=GSN((n_0-1)\cdot\frac{b-a}{n_0}<x_{n_0}<b)$$

在确定 $A_2$ 属性的区分度时，确定 $A_2$ 属性的取值对应的子区间以及 $c_{n_0}$ $(n_0\in N^*)$，从而得到区分度为：

$$Dis(A_2)=\frac{c^{(2)}}{c_{n_0}}$$

其中，$C=GSN(a<x<b)$，$GSN$ 为 GetStrategyNumber 的缩写，表示在相应区间内属性取值的全部个数。

对数值类型属性范围划分子区间时，子区间的个数 $n_0$ 无固定值，可根据属性范围的大小、属性值的个数等因素设定适合的数值。$n_0$ 设置过小时，对无效规则过滤的效果明显；$n_0$ 设置过大时，由于子区间个数太多，在预处理阶段对子区间区分度的计算要消耗大量的计算代价。不失一般性，本文将数值类型属性的子区间个数设定为 10。

3)日期类型。对于日期类型属性 $A_3$，将数值类型作为对比：设取值范围为$\{x\mid a<x<b\}$，其中 $a$ 为 $A_3$ 中的最早日期，$b$ 为 $A_3$ 中的最晚日期。同样地，将取值区间分为 $n_0$ 个子区间，例如：$A_3$ 取值区间为一年，则可以按照月份将区间划分为 12 个子区间。

在确定 $A_3$ 属性区分度时，按照数值类型的处理方式确定 $A_3$ 属性取值对应的子区间以及 $c_{n_0}$ $(n_0\in N^*)$，得到的区分度为：

$$Dis(A_3)=\frac{c^{(3)}}{c_{n_0}}$$

4）字符串。对于字符串类型属性 $A_4$，其取值为数量和顺序随机的字符组合，其取值的个数在理论上是无限多的，且字符串类型属性的取值无法依据大小排序。设 $A_4$ 为字符串类型属性，在处理此类型的区分度问题上，假设：①$A_4$ 的取值是唯一的，可以最先查询；②可以找到符合查询条件的一条策略。

在一次策略查询中，暂定字符串类型属性 $A_4$ 的顺序为最先查询，即在一次策略查询的条件中，$A_4$ 属性的取值被认为在策略集中唯一存在，若在策略集中查找出符合查询条件中 $A_4$ 取值的一条策略，则认为找到了符合条件的策略。在假设前提①不存在时，需在②的前提下重新评估 $A_4$ 的区分度。

在策略集中，符合一个 $A_4$ 取值的策略可被看作是均匀分布的，当第 $n$ 次查找到符合 $A_4$ 取值的策略中的条件不符合查询的条件时，按照已查询的策略条数 $c_1$ 和策略集中策略的总数 $c^{(4)}$ 估测在策略集中符合 $A_4$ 取值的个数 $c=\frac{n\cdot c^{(4)}}{c_1}$。此时区分度被重新计算：

$$Dis(A_4)=\frac{c^{(4)}}{c}$$

利用以上 4 种属性类型的区分度计算方法可以确定属性与或矩阵中各个属性的区分度大小，而区分度大小则用来在属性与或矩阵进行规则过滤时判定使用属性的先后次序。一个属性的区分度越大，这个属性次序越靠前。

### 2.4　策略查询优化方法

本文方法将上述预处理过程与原有的 Context Handler 过程结合，即图 1 的 Context Handler 发送信息时会将预处理得到的区分度集一并发送到 PDP。需要注意的是，在规则集没有发生改变时，无需重新计算各属性的区分度。在 PDP 对访问控制请求决策时，须遵循以下步骤。

1）在区分度集 *DisSet* 选择当前最大区分度 $dis_i(j\in\{x\mid 0\leqslant x\leqslant n, x\in N\})$ 的属性 $Att_j$，对应的访问请求 *Req* 属性 $ReqAtt_j$ 作为过滤条件时，在属性与或矩阵中（即规则集）相应属性 $RuleAtt_j$ 作为过滤对象，过滤后得到一个规则子集 *parRuleSet*；

2）在 *DisSet* 中去除已被筛序的属性，继续执行步骤 1），直至查询到最后一个属性；

3）在过滤完毕的规则子集中，若有一条规则与访问控制请求中的各属性值相等，则以规则中的 Effect 作为决策结果，否则根据策略的合成方式发送相应的决策结果至上下文处理器。

上述过程的伪代码描述如算法 1 所示。

**算法 1**　规则集的优化查询算法

```
输入：规则集 RuleSet，区分度集 DisSet
输出：目标规则的 Effect
1. for each dis∈DisSet//DisSet 经过排序
2.   if Dis(ReqAtt)=dis//找到区分度相对应的访问控制请求属性
3.     Ij←ReqAtt.Interval;//获得对应属性所在区间
4. end if
5. for each rule∈RuleSet
6.       if rule.RuleAtt∉Ij
7.         RuleSet←RuleSet−Rule;
8.       end if
9.   end for
10. end for
11. for each rule∈RuleSet
12.  if Req=rule
13.    return rule.Effect;
14.  end if
15. end for
```

### 2.5　方法分析

在实际应用中，字符串可以用来表示姓名、学号这类具有高度唯一性的属性。在这种情况下，传统方法与本文所提方法的关系为：

$$x_1=(x_2-j+1)\times j \tag{2}$$

其中，$x_1$ 为传统方法的访问次数，$x_2$ 为本文方法的访问次数，$j$ 为规则集中所包含的属性的个数。在属性取值为随机均匀分布时，使用传统的方法需要访问的期望次数为：

$$E(x_1)=\frac{1}{n}\sum_{k=1}^{n}(j\cdot k) \tag{3}$$

其中，$n$ 为测试规则的数量，每个样本被选中为目标策略的概率为 $\frac{1}{n}$。在属性取值为随机均匀分布时，本文所提方法的期望值为：

$$E(x_2)=n+\sum_{l=1}^{j-1}\frac{n}{m\cdot l} \tag{4}$$

其中，$m$ 为划分数值类型和日期类型的子区间数量，不失一般性，在本文中 $m$ 值被设定为 10。

## 3　实验及分析

本节将按照 XACML 官方测试包构造的策略文件作为实验数据，通过实验验证对规则集进行访问时第 2 节提供的方法具有比传统遍历方法访问次数少的优势。本文实验环境如下：处理器型号为 Intel(R) Core(TM)2 Quad CPU Q9500 @ 2.83GHz；内存 8GB DDR3；操作系统为 Windows 7 旗舰版、SP1；Java JRE 1.8。

### 3.1　应用实例过程及分析

在构造的策略文件中有一条规则 *Rule* 为 permit←*sName*='pus'，*rDocID*='32'，*aRight*='write'，*cStaringDate*>'2016/4/3'，意为：'pus'可以阅读编号为'32'且更新日期为'2016/4/3'之后的文档。当有一个控制请求中的各个属性均符合 *Rule* 中的 Subject，Resource，Action 和 Condition 时，按照本文方法找到目标规则需要以下步骤：

1）属性提取，将所涉及规则中的各个属性值放入属性与或矩阵；

2）计算区分度，对属性与或矩阵的每列（即规则中的每个属性），根据属性类型，按照 2.3 节的计算方法计算各个属性的区分度；

3）按照各个属性的区分度大小，依次对现有规则集过滤，访问请求将在过滤后的规则集中匹配有效规则。

由数据集建立的属性与或矩阵如图 3 第一个表所示，其中，*RuleID* 为额外建立的一列，用于标识规则。属性与或矩

阵中的一行表示为一条规则，由于在策略文件中设定的 *Action* 均为 read 和 write，因此按照 read 和 write 将策略文件中的规则分成两条规则存入属性与或矩阵，100 个策略文件最后产生了 200 条规则。按照 2.3 节中提供的区分度计算方法可知，*sName* 为字符串类型，区分度首先被设定为最大；*rDocID* 取值类型为 int，分成 10 个子区间后，'32'位于 30 到 40 的子区间内，区分度为 12.5；*aRight* 为一个枚举类型，因为其取值 *aRight*={write，read}，区分度为 2；*cStartingDate* 为日期类型，取值区间范围为一年，将其划分为 12 个子区间，'2016/4/3'位于 2016/3－2016/4 区间内，区分度为 12.3。区分度大小排序为 *Dis*(*sName*)＞*Dis*(*rDocID*)＞*Dis*(*cStartingDate*)＞*Dis*(*aRight*)。首先依据 *sName* 属性对属性与或矩阵进行规则过滤，由于 *sName* 为字符串类型并且其指代的姓名具有高度唯一性，因此 *sName* 具有极高的区分度，在第一次使用 *sName* 属性过滤后，数据子集中的规则仅剩 2 条，使用 *aRight* 属性再次过滤后，得到目标规则。在实际使用中，若规则中的一个属性为字符串类型，则使用本文方法进行过滤时的效率更高。

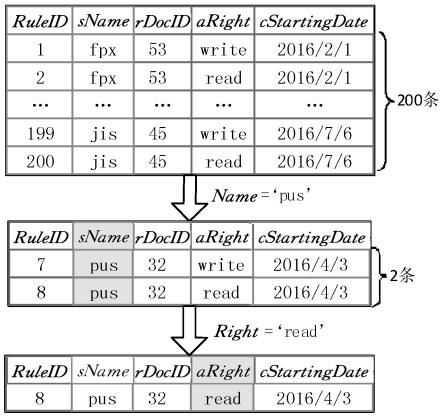


图 3 规则过滤过程示意图

Fig. 3 Diagram of rules filtering process

### 3.2 策略评估效率实验分析

字符串类型在实际应用中具有高度的唯一性，在实验中只能部分反映真实的对比情况。在官方测试数据集中[9]，主体为字符串类型，不具有其他场景，因此本次对比实验中使用的是按照 XACML 官方测试数据集的格式构造的数据集，分为两组。在数据集 1 中，Subject，Resource，Action 的类型分别为字符串类型、日期类型、枚举类型；在数据集 2 中，Subject，Resource，Action 的类型分别为数值类型、日期类型、枚举类型。两个数据集中各个属性的取值范围如下：1）数值类型，在 1～100 区间随机取值；2）字符串类型，每个字符在 26 个字母中随机取得，并组合为长度为 3 的随机字符串；3）日期类型，方法与数值类型相同，时间范围为一年；4）枚举类型，根据设定的枚举值随机抽取赋值。参照 XACML Implementation[10-11] 的策略评估过程，通过修改其中的 Context Handler 实现了本文实验。

图 4 中的传统方法是指可对无效规则多次遍历的传统策略查询方法。实验对比的依据为：在寻找相同的目标策略时，对矩阵访问的次数越多，则消耗的时间越长。

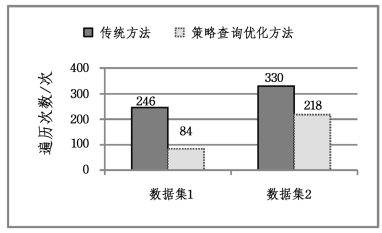


图 4 规则遍历次数的对比

Fig. 4 Comparison of traversal times for rules

2 个数据集在 3 次遍历中的访问次数如图 5 所示。在第一组数据集中，字符串类型具有高度的唯一性，在根据字符串类型属性进行筛选时，若此类型的属性值匹配，则会对整条规则进行匹配，可快速找到目标策略，在第一次遍历时无需对规则集全部访问，消耗的次数较少。在第二组数据集中，按照区分度的高低，依次对 Subject，Resource，Action 进行筛选。

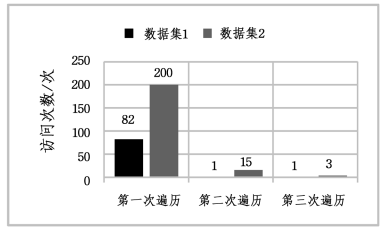


图 5 3 次遍历中的访问次数

Fig. 5 Traversal times in three traversals

如图 4 所示，在实验结果对比中，数据集 1 的实验结果对比与 2.5 节的方法分析一致，在数据集 2 中两种方法的期望分别为：$E(x_1)=\frac{1}{n}\sum_{k=1}^{n}(3\cdot k)=301.5$，$E(x_2)=n+\frac{n}{m}+\frac{n}{2m}=230$。实验结果表明，本文所提方法与传统遍历方法相比具有明显优势。

**结束语** XACML 策略规模庞大和遍历匹配策略模式是导致控制请求响应效率低的两大因素。本文从匹配策略模式入手，增加预处理过程，提出了基于属性与或矩阵和类型分析的策略属性提取方法，对策略集进行纵向的、有针对性的筛选。实验结果表明，本文提出的基于属性与或矩阵和类型分析的 XACML 策略属性提取方法减少了匹配运算量，提高了匹配速度。

## 参 考 文 献

[1] WANG Y Z，FENG D G. A Conflict and Redundancy Analysis Method for XACML Rules [J]. Chinese Journal of Computers，2009，32(3)：516-530. (in Chinese)
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.

[2] NIU D H，MA J F，MA Z，et al. HPEngine：high performance XACML policy evaluation engine based on statistical analysis [J]. Journal on Communications，2014，35(8)：206-215. (in Chinese)
牛德华，马建峰，马卓，等. 基于统计分析优化的高性能 XACML 策略评估引擎[J]. 通信学报，2014，35(8)：206-215.

[3] WANG Y Z，FENG D G，ZHANG L W，et al. XACML policy evaluation engine based on multi-level optimization technology

[J]. Journal of Software, 2011, 22(2): 323-338. (in Chinese)
王雅哲，冯登国，张立武，等. 基于多层次优化技术的 XACML 策略评估引擎[J]. 软件学报，2011，22(2)：323-338.

[4] QI Y, CHEN J, LI Q M. XACML policy evaluation optimization method based on reordering [J]. Journal of Nanjing University of Science and Technology, 2015, 39(2): 187-193. (in Chinese)
戚湧，陈俊，李千目. 一种基于重排序的 XACML 策略评估优化方法[J]. 南京理工大学学报，2015，39(2)：187-193.

[5] CHEN J. The research on XACML strategy optimization method [D]. Nanjing: Nanjing University of Science and Technology, 2015. (in Chinese)
陈俊. XACML 策略优化方法研究[D]. 南京：南京理工大学，2015.

[6] CHEN W H, WANG N N. Research on XACML policy evaluation optimization technology [J]. Application Research of Computer, 2013, 30(3): 900-905. (in Chinese)
陈伟鹤，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.

[7] QI Y, CHEN J, LI Q M, et al. XACML strategy optimization method based on redundancy elimination and attribute numericalization [J]. Journal of Computer Science, 2016, 43(2): 163-168. (in Chinese)
戚湧，陈俊，李千目. 基于冗余消除和属性数值化的 XACML 策略优化方法[J]. 计算机科学，2016，43(2)：163-168.

[8] eXtensible Access Control Markup Language(XACML) Version 3.0 [EB/OL]. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.doc.

[9] XACML 2.0 conformances tests [EB/OL]. http://www.oasis-open.org/committees/download.php/14846/xacml2.0-ct-v.0.4.zip.

[10] Sun's XACML Implementation [EB/OL]. http://sunxacml.sourceforge.net.

[11] Enterprise XACML Implementation [EB/OL]. http://sourceforge.net/projects/java-xacml.

(上接第 206 页)

**结束语**　为提高事故分析中多事件演化的准确性和事件因果分析的全面性，充分利用历史专家经验，本文基于 HMM 设计了一种面向故障树结构匹配的算法。构造的 HMM 模型充分考虑了故障树结构特征，利用维特比算法检测待匹配案例故障树，并完成算法的原型系统开发。实验结果表明，算法匹配具有较高的准确率。该方法不仅有助于分析人员在事故分析中充分利用已有经验的支持，而且可以逐步建立以故障树模型为中心的同类事故因果的汇总，能够更加全面、准确地分析事故形成的机理，从而有针对性地开展安全生产与事故预防工作。后续的主要研究工作是改进历史统计数据的完备性和节点中文表达的规范性，降低状态转移概率分布与观测概率分布值之间存在的偏差，以扩大事故案例样本，进一步提升文中算法的效率和准确性。

## 参 考 文 献

[1] XU B F, HUANG Z Q, HU J, et al. Time Property Analysis Method for State/Event Fault Tree[J]. Journal of Software, 2015, 26(2): 427-446. (in Chinese)
徐丙凤，黄志球，胡军，等. 一种状态事件故障树的时间特性分析方法[J]. 软件学报，2015，26(2)：427-446.

[2] TANAKA H, FAN L T, LAI F S, et al. Fault-Tree Analysis by Fuzzy Probability[J]. IEEE Transactions on Reliability, 2009, R-32(5): 453-457.

[3] CUI T J, MA Y D. Establishment of DSFT and determination of spatial distribution of probability of failure [J]. Systems Engineering-Theory & Practice, 2016, 36(4): 1081-1088. (in Chinese)
崔铁军，马云东. DSFT 的建立及故障概率空间分布的确定[J]. 系统工程理论与实践，2016，36(4)：1081-1088.

[4] ALIEE H, ZARANDI H R. A Fast and Accurate Fault Tree Analysis Based on Stochastic Logic Implemented on Field-Programmable Gate Arrays[J]. IEEE Transactions on Reliability, 2013, 62(1): 13-22.

[5] NI J, TANG W, XING Y. A Simple Algebra for Fault Tree Analysis of Static and Dynamic Systems[J]. IEEE Transactions on Reliability, 2013, 62(4): 846-861.

[6] LI Z F, REN Y, LIU L L, et al. Parallel algorithm for finding modules of large-scale coherent fault trees[J]. Microelectronics Reliability, 2015, 55(9/10): 1400-1403.

[7] DOOHWAN O H, WON WOO R O. High Performance Pattern Matching algorithm with Suffix Tree Structure for Network Security[J]. Pattern matching, 2014, 51(6): 110-116.

[8] LIU P. Tree structure matching pursuit based on Gaussian scale mixtures model[J]. Proc Spie, 2011, 8135(1): 813520-813520-8.

[9] ZHANG H W, JIE X F, DUAN Y Y, et al. A directed graph based subgraph matching query algorithm based on adaptive structure outline[J]. Chinese Journal of Computers, 2017, 40(1): 52-71. (in Chinese)
张海威，解晓芳，段媛媛，等. 一种基于自适应结构概要的有向标签图子图匹配查询算法[J]. 计算机学报，2017，40(1)：52-71.

[10] LI R Y, HONG L. A subgraph matching method based on inclusion degree[J]. Journal of Software, 2018, 29(6): 1792-1812. (in Chinese)
李瑞远，洪亮. 一种基于包含度的子图匹配方法[J]. 软件学报，2018，29(6)：1792-1812.

[11] LV Q, QIAO Y, ANSARI N, et al. Big Data Driven Hidden Markov Model Based Individual Mobility Prediction at Points of Interest[J]. IEEE Transactions on Vehicular Technology, 2017, PP(99): 1.

[12] CHOI K W, HOSSAIN E. Estimation of Primary User Parameters in Cognitive Radio Systems via Hidden Markov Model[J]. IEEE Transactions on Signal Processing, 2013, 61(3): 782-795.

[13] SOUALHI A, CLERC G, RAZIK H, et al. Hidden Markov Models for the Prediction of Impending Faults[J]. IEEE Transactions on Industrial Electronics, 2016, 63(5): 3271-3281.

[14] HAGENAUER J. Source-controlled channel decoding[J]. IEEE Transactions on Communications, 2002, 43(9): 2449-2457.

[15] NING A, LI X, WANG C. ASE Approach for Large Graphs Matching[J]. Information Technology Journal, 2013, 12(14): 2969-2974.