第38卷 第17期
Vol.38 No.17

计 算 机 工 程
Computer Engineering

2012年9月
September 2012

·软件技术与数据库·

文章编号：1000—3428(2012)17—0042—04 文献标识码：A 中图分类号：TP311

基于改进FSM的RBAC测试集约简方法

袁佳琳，高建华

(上海师范大学计算机科学与技术系，上海 200234)

摘 要：使用完备的有限状态机生成一致性测试集虽然有效，但数量庞大。针对该问题，考虑一般系统访问控制的基本需求，提出6种探索式方法对有限状态机(FSM)进行约简，有效避免状态爆炸的现象发生，简化了生成的一致性测试集大小。对基于FSM生成树进行实验，结果表明，改进FSM对缩小基于角色的访问控制系统一致性测试集是有效的。

关键词：有限状态机；权限控制；基于角色的访问控制；一致性测试集；试探法；错误覆盖率

Reduction Method of Role-based Access Control Test Suite Based on Improved Finite State Machine

YUAN Jia-lin, GAO Jian-hua

(Department of Computer Science and Technology, Shanghai Normal University, Shanghai 200234, China)

【Abstract】A test suite generated using complete Finite State Machine(FSM) has excellent fault detection ability, but it is astronomically large. This paper presents six heuristic method to reduce the size of the FSM model based on the general requirements from the access control system. These methods not only avoid the state explosion, but also simplify the generation of conformance test suite size. This paper designs experiment based on spanning tree of the FSM. Experimental result shows that using improved FSM to reduce the comformance test suite of Role-based Access Control(RBAC) system is effective.

【Key words】 Finite State Machine(FSM); authority control; Role-based Access Control(RBAC); conformance test suite; heuristic method; fault coverage rate

DOI: 10.3969/j.issn.1000-3428.2012.17.012

1 概述

权限控制是目前大部分软件都包含的功能，无论是操作系统软件、数据库软件或是行业应用软件都会使用一些角色策略来限制和授权用户访问应用功能、文件系统和数据等，软件将这些角色策略实现为应用前端，在用户请求访问系统资源前要求用户给予授权证明，之后再授权该用户使用已被激活的资源请求。这部分应用前端被称作基于角色的访问控制(Role-based Access Control, RBAC)模块[1]。对于软件测试，RBAC模块可以被单独取出并进行一致性测试，即测试软件权限控制的实现是否与权限需求定义相一致，访问控制的一致性测试也是系统功能性测试的组成之一。

使用有限状态机(Finite State Machine, FSM)是生成一致性测试集的常用方法，但是当用户、角色、权限之间的关系复杂时，有限状态机将是一个庞大的有向图，其生成的测试集也是非常庞大的，使测试的效率降低。本文结合一般系统访问控制的基本需求，将提出6种探索式方法[2]来简化有限状态机的规模，避免用户、角色、权限三者复杂关系产生的冗余，并通过实验比较这些方法的故障检测率。

2 背景及相关知识

2.1 RBAC模型

RBAC作为一种广泛使用且灵活的访问控制技术，该技术引入角色概念，将用户和权限解耦，所有的权限不是直接分配给用户，而是分配给不同的角色，用户通过角色获取相应的权限。根据NIST制定的标准RBAC模型[3]，可以分成3级，功能不断扩展，分别是基本模型RBAC0、角色层级模型RBAC1、角色约束模型RBAC2。

RBAC0定义了能构成一个RBAC控制系统的最小元素集合，包括5个基本元素：用户集，角色集，权限集，操作集，对象集。此外，RBAC0还包含一个会话集以及2种多对多的重要关系：用户与角色的关系，记为UA。UA⊆用户集×角色集，用户角色分配关系是多对多的映射；角色与权限的关系，记为PA。PA⊆权限集×角色集，角色

基金项目：国家自然科学基金资助项目(61073163)；上海市科委基金资助项目(09220503000)；上海市引进技术的吸收与创新计划基金资助项目(2010CH-014)

作者简介：袁佳琳(1986－)，女，硕士研究生，主研方向：软件可靠性设计理论与方法，软件测试技术；高建华，教授、博士

收稿日期：2011-10-19 **修回日期**：2011-12-18 **E-mail**：ivy_yuan@shnu.edu.cn

权限分配关系是多对多的映射。

RBAC1 在 RBAC0 的基础上引入了角色间的继承关系，记为 RH 关系，RH⊆角色集×角色集。角色间的继承关系可分为一般和受限继承关系。一般继承关系是一个任意偏序关系[4]，允许角色间的多继承，是一个图结构；受限继承关系要求角色继承关系是一个树结构，实现角色间的单继承。

RBAC2 在 RBAC1 的基础上引入了权责分离关系，记为 SoD(Separation of Duty)关系，用于避免角色间的冲突。在 RBAC2 中提出了 2 种权责分离解决方案，一种是静态权责分离，记为 SSD(Static Separation of Duty)，即当一个用户拥有一个角色，那么他就不能同时拥有与之冲突的另一个角色；另一种是动态权责，记为 DSD(Dynamic Separation of Duty)，即用户可以拥有多个相互冲突的角色，但是在一次会话中不能同时扮演 2 个冲突的角色。

本文基于 RBAC2 模型提出访问控制策略，将用户角色分配和激活分开处理与讨论。

2.2　传统 FSM 生成的一致性测试序列方法

基于模型的软件测试方法能有效地形式化被测系统的具体行为，生成测试用例，并判断检测被测系统是否满足实际需求。常用的模型包括 FSM(有限状态机)、马尔可夫链、UML 等。本文使用 FSM 模型形式化表示访问控制策略。

FSM 模型[5]是一个 5 元组：M=(有限输入集合 S，有限状态集合 E，有限输出集合 Y，状态转移函数 T，输出函数 O)，其中，T 为 S×E→E；O 为 S×E→Y，表示当有限状态机处于状态 $e_i(e_i \in E)$时，获得了一个输入请求 $s_i(s_i \in S)$，状态转移到下一个状态 $e_j(e_j \in E)$，同时输出 $y_i(y_i \in Y)$。

2.2.1　用户角色关系

FSM 模型一般用状态转移图来表示，本文将状态 E 用一连串二进制序列表示，每对二进制序列描述一组用户角色关系或角色权限关系。有限输入集合 S={AS, DS, AC, DC, AP, DP}。AS(Assign)和 DS(Deassign)分别表示用户分配和取消分配给某个角色；AC(Activate)和 DC(Deactivate)分别表示激活和取消激活用户属于某个角色；AP(Assign for Permissions-role assignments)和 DP(Deassign for Permissions-role assignments)分别表示权限分配和取消分配给某个角色。用户角色关系的一对二进制序列如表 1 所示。

表 1　用户角色关系

二进制序列	用户角色关系	
	分配	激活
00	否	否
10	是	否
11	是	是
01	不允许	不允许

假设给定 2 个用户，分别是 u_1 和 u_2，一个角色 r，使用有限状态机建模时，按表 1 所示的二进制序列对表示用户角色关系，2 个用户需用 2 对二进制序列，把这 2 对二进制序列连列作为用户角色关系的一个状态，即有 0000、0010、1000、1010、1100、0011、1110、1011、1111 这 9 种状态，集合 S_{UR}={AS, DS, AC, DC}即该 FSM 的输入集。部分 FSM 如图 1 所示。

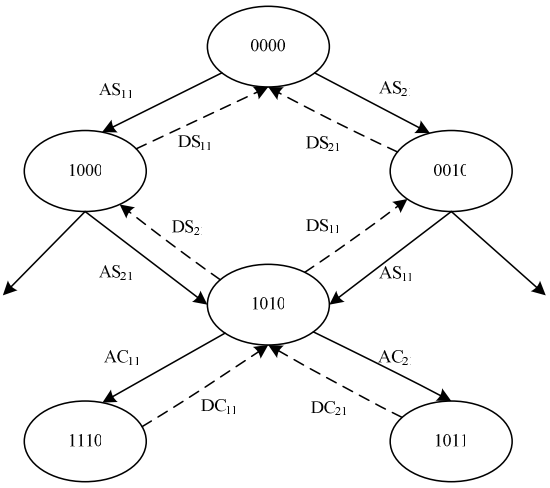

图 1　部分 FSM

假设给定多用户多角色，如 n 个用户、m 个角色，当使用有限状态机建模时，按表 1 所示的二进制序列对表示用户角色关系，需用 $n \times m$ 对二进制序列连列作为该用户角色关系的一个状态，即有 $3^{n \times m}$ 种状态。

2.2.2　角色权限关系

对于角色权限关系要比用户角色关系稍显简单，因为权限与角色没有激活关系，只有分配关系，所以可直接只用 1 个二进制位来表示单角色与单权限的关系，即 1 表示该角色已被分配了该权限，0 表示该角色还没有该权限。对于一个角色 r 和 2 个权限 p_1、p_2 使用有限状态机建模时只有 00、01、10、11 这 4 种状态，集合 S_{RP}={AP, DP}为该 FSM 的输入集合即可得到 FSM，如图 2 所示。

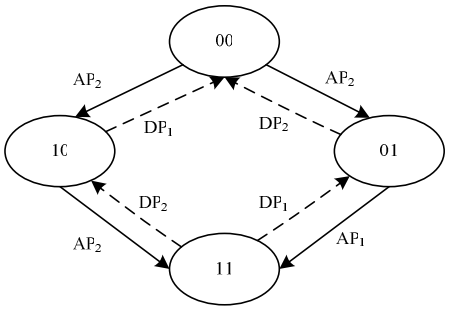

图 2　角色权限关系的 FSM

假设给定多角色多权限，如 m 个角色，k 个权限，使用有限状态机建模时，则需用 $m \times k$ 个二进制位表示一个状态，即该 FSM 共有 $2^{m \times k}$ 种状态。

得到用户角色关系、角色权限关系的 FSM 模型后就可以使用常用的基于 FSM 生成测试集方法来生成一致性测试集，常见的有周游法(T 方法)、区分序列法(D 方法)、特征序列法(W 方法)、唯一输入/输出法(UIO 方法)等[6]。

2.2.3 面临的问题与挑战

不难看出，当用户、角色、权限的数量不大，它们之间的关系不是很复杂时，传统的基于 FSM 生成一致性测试序列方法相当有效，但随着软件系统应用范围的扩大，用户、角色、功能、资源的数量急剧增大，之间的关系也日趋复杂，传统方法建立的 FSM 模型会是一个惊人的复杂有向图，从中生成一致性测试集的花销很大，效率也很低[6]。如表 1 所示，单用户与单角色的关系有 3 种状态，n 个用户与 m 个角色之间的关系需用一个长为 $2\times n\times m$ 的二进制序列表示一个状态，该用户角色关系有 $3n\times m$ 个状态，由此生成的 FSM 就有 $3n\times m$ 个结点，使用常用的基于 FSM 生成测试集方法的时间复杂度很大。

3 基于试探法的测试集约简方法

由于传统 FSM 生成一致性测试集的方法存在一定的问题，本文提出如下 6 种约束[7]来减少 FSM 复杂度以简化测试集生成的开销。

3.1 H1 方法(角色分配和激活的分开测试)

将用户角色的分配和激活关系分别构造 FSM，那么表示单用户与单角色的分配和激活关系都只需 1 位二进制位，分别只有 0 和 1 这 2 种状态。假设有 n 个用户与 m 个角色，那么表示用户角色的分配和激活关系的 FSM 模型分别有 $2^{n\times m}$ 种状态，考虑整个用户角色关系的状态数量就从 $3^{n\times m}$ 约简为 $2\times 2^{n\times m}$。

3.2 H2 方法(只允许单用户同时激活一个角色)

对于允许单用户拥有多个不同角色的系统而言，考虑 RBAC2 中的动态权责分离，为减少测试用 FSM 的大小，可以不仅不允许同时激活冲突的角色，而是只允许单用户同一时刻激活单角色。假设有 n 个用户与 m 个角色，那么表示用户角色关系的 FSM 模型有 $(m\times 2^{m-1})^n$ 种状态。

3.3 H3 方法(只允许为同一用户分配一个角色)

对于拥有多个相互不冲突的角色的系统而言，考虑 RBAC2 中的静态权责分离，为减少测试用 FSM 的大小，可以不仅不允许为单用户分配相互冲突的角色，而是只允许为单用户分配单角色，然而不同的用户可以拥有同样的角色。假设有 n 个用户与 m 个角色，那么表示用户角色关系的 FSM 模型有 $(3\times m)^n$ 种状态。

3.4 H4 方法(单个用户与角色的关系的测试)

对于每个用户，可以分别测试该用户与角色的关系。对于一个角色，与该用户有 3 种状态：未分配未激活(二进制位表示为 00)、已分配未激活(二进制位表示为 10)、已分配已激活(二进制位表示为 11)，则该用户与 m 个角色的关系会有 3^m 个结点。分别测试每个用户与角色的关系，那么 n 个用户总共有 $n\times 3^m$ 个结点。

3.5 H5 方法(单个角色与用户的关系的测试)

如同 3.4 节所述，对于每个角色，也可以分别测试该角色与用户的关系。由于每个用户与该角色同样也有 3 种关系：未分配未激活(二进制位表示为 00)，已分配未激活(二进制位表示为 10)，已分配已激活(二进制位表示为 11)，则该角色与 n 个用户的关系有 3^n 个结点。分别测试每个角色与用户的关系，那么 m 个角色总共有 $m\times 3^n$ 个结点。

3.6 H6 方法(用户组概念的创建)

对于拥有一些相同属性的用户，根据需求将其分成一组或多个组别，每个组别内的用户被分配相同的角色。此时，角色分配就不再是针对单个用户，而是针对一个用户组了。假设有 n 个用户与 m 个角色，n 个用户被分成 k 组，那么表示用户角色关系的 FSM 模型有 $3^{k\times m}$ 种状态。

4 实验与分析

上述 6 种约束方法，由于在测试时对用户和角色加上了一些限制，FSM 模型的缩小是不言而喻的，但是加上限制后对测试的有效性产生了影响也是不言而喻的，即加上的限制条件可能会使测试遗漏故障，本文的实验检验 H1~H6 能否能完整得测试出 RBAC 系统的权限故障，以及混合使用这 6 种方法对故障的检测遗漏率的影响。

4.1 实验环境

由于 H1~H6 产生的 FSM 不同，对于每种方法产生的 FSM 都要测试在每个结点状态下的输入和输出是否匹配，设计了如下测试环境，如图 3 所示。

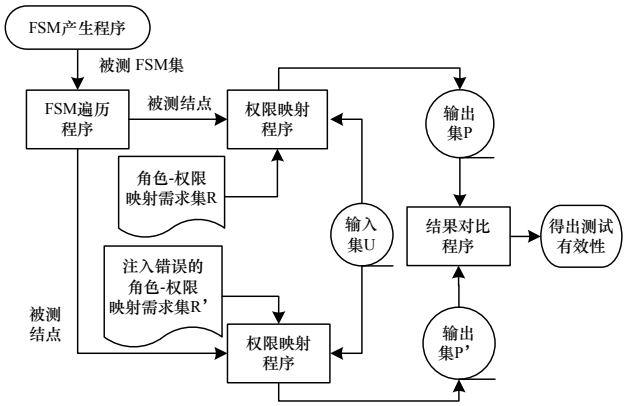

图 3 测试环境

FSM 产生程序每次产生一个 FSM 集，分别符合 H1~H6 的约束，每个 FSM 集中可能包含一个或多个 FSM，FSM 遍历程序分别遍历 FSM 集中每个 FSM 的所有状态结点，将该状态结点送入 2 个权限映射程序，这 2 个权限映射程序相同，但分别读取正确的角色-权限映射集 R 和注入了错误的角色-权限映射集 R'，输入相同的用户集 U 将得到 2 个输出权限集合 P 和 P'。根据 FSM 遍历程序和输入用户集 U，获知相应角色，再通过正确及注入了错误的 2 个角色-权限映射集，就能得到相应的输出权限集合，再由结果对比程序对比 2 个输出集合 P 和 P'得出测试有效性。为了结果对比程序能检测经过约简后的测试集的故障检测率，FSM 产生程序先生成一个完备的 FSM 集，再通过 H1~H6 的单独约束和组合约束，比较新生成的 FSM 集对测试有效性的最终影响。

4.2 实验方法

4.2.1 FSM 生成树

由于生成 FSM 的目的是遍历所有状态结点，验证每

个状态下角色-权限的正确性，而生成完备的 FSM 比较困难，并且完备的 FSM 包含大量附加边，即遍历图的所有结点后都不会经过的边，因此这里使用 FSM 的生成树来达到遍历 FSM 每个结点的目的。假设实验中有 2 个用户和 6 个角色，完备的 FSM 和受 H1~H6 限制的 FSM 生成树的结构如下述说明。

完备 FSM 的生成树是一颗完全三叉树。根结点为空结点，第 1 层表示第 1 个用户与第 1 个角色间的关系，根据表 1 所示，它们分别有 00、10、11 这 3 种关系，即根结点有 3 个孩子结点，第 2 层表示第 1 个用户与第 2 个角色的关系，也有 00、10、11 这 3 种关系，即第 1 层的 3 个结点分别有 3 个孩子结点，以此类推，至第 6 层表示第 1 个用户与第 6 个角色间的关系，到第 6 层为止第 1 个用户与所有角色的关系结束，第 7 层开始表示第 2 个用户和第 1 个角色的关系，则这棵树共有 12 层(用户数×角色数)，是一棵 12 层的完全三叉树，从根结点经过任意路径到叶子结点组成的序列就是完备 FSM 中的一个状态结点。

受 H1~H6 限制的 FSM 的生成树实际是完备 FSM 生成树的一棵缩减树或剪枝森林，即可能通过剪枝、取子树等操作的组合得到的树或森林。这里以 FSM_0 表示完备的 FSM，FSM_1~FSM_6 表示受 H1~H6 限制的 FSM，以 T_i 表示 FSM_i 的生成树或森林($0 \leqslant i \leqslant 6$)。

H1 分开测试角色分配与激活，测试激活时假设用户已属于所有角色，由于只有分配、未分配(激活、未激活) 2 种状态，因此测试分配和测试激活时都会生成一棵 12 层的完全二叉树，只需测试一棵即可，即 T_1 是一棵 12 层的完全二叉树。H2 只允许一个用户同时激活一个角色，那么将 T_0 中所有含有 2 个或以上的激活角色的分枝剪去即为 T_2。H3 只允许一个用户同时分配一个角色，那么将 T_0 中所有含有 2 个或以上的分配角色的分枝剪去即为 T_3，很明显 T_3 要比 T_2 简单。H4 分别测试单个用户与角色的关系，那么 T_4 是一个含有 2 棵 6 层完全三叉树的森林，这 2 个树分别表示一个用户与 6 个角色的关系，这 2 棵树对于测试权限正确性是等价的，所以，只需测试一棵树即可。H5 分别测试单个角色与用户的关系，那么 T_5 是一个含有 6 棵 2 层完全三叉树的森林，由于一棵树只表示所有用户与一个角色关系，显然对于权限验证这 6 棵树必须分别测试。H6 创建用户组，由于创建用户组的方法比较灵活，这里就以 2 个用户为同一组来测试，相当于将 2 个用户看成一个用户，T_6 是即使一棵 6 层的完全三叉树。

4.2.2　测试有效性的计算

图 1 中的 FSM 生成程序实际是 FSM 生成树的生成程序，FSM 遍历程序实际是 FSM 生成树的遍历程序，只需使用深度优先遍历该生成树或生成森林，遍历到叶子结点时输出当前路径产生的序列给予权限映射程序继续下一个步骤即可。

权限映射程序读取角色-权限映射表，将 FSM 遍历程序输出的状态结点序列映射成为当前用户的权限集，要注意的是只有激活的角色才会被分配到权限，那么实际的输出除了权限集还必须有用户被分配的角色集，只有角色集和权限集都相同才认为是和需求一致的。

实验中设置了 2 个角色-权限的配置，一个代表符合需求的映射配置，另一个在正确的配置中注入了许多错误，结果对比程序对比读取正确的角色-权限映射配置和注入了错误的角色-权限映射配置所生成的权限集 P 和 P'。完备的 FSM 得到的 P 和 P'对比结果包含所有的预计错误，这里就是用这个错误数表示预计错误数，受 H1~H6 限制的 FSM 得到的 P 和 P'对比结果所含错误数肯定小于或等于预计错误数。最后将比对 H1~H6 的输出集得出的错误数与预计错误数之比计为 H1~H6 的测试有效率。

4.3　实验结果

同样以 P_0 和 P_0'表示完备 FSM 得到正常权限集和注入错误的权限集，以 P_i 和 P_i'表示受 H_i 限制的 FSM 得到的正常权限集和注入错误的权限集($1 \leqslant i \leqslant 6$)，以 N_i 表示结果对比程序对比 P_i 和 P_i'后得出的存在错误的状态结点数($0 \leqslant i \leqslant 6$)，$N_{ia}$ 表示当前 FSM_i 的全部状态结点数，那么 $\lambda_i = \frac{N_i}{N_{ia}}$ 即该 H_i 测试出的故障率，它和完备 FSM 的故障率比值即该 H_i 方法的测试有效性。本次实验取 2 用户 6 角色得出的结果如表 2 所示。

表 2　有效性测量结果

FSM	状态结点数 N_{ia}	有错误的结点数 N_i	正确的结点数	故障率/(%)	测试有效性/(%)
完备	531 441	467 432	64 009	87.96	100.00
H1	4 096	3 007	1 089	73.41	83.47
H2	28 672	24 576	4 096	85.71	97.45
H3	169	114	55	67.46	76.69
H4	729	476	253	65.29	74.24
H5	54	30	24	55.56	63.16
H6	729	476	253	65.29	74.24

从表 2 中可以看出，完备的 FSM 共有 531 441 个状态结点，读取注入错误的角色权限映射配置后有 467 432 个状态结点的最终权限有错误，故障率达到 87.96%。受 H1~H6 限制的 FSM 能大幅度减少状态结点数，但存在测试遗漏的故障点，能检测出的故障与状态结点数基本呈正比关系。H2 的 FSM 状态结点数最多，测试有效性达到了 97.45%；H3 的 FSM 中包含较多没有激活任何角色的状态结点，这些结点不依赖于角色权限，都没有任何权限，所以使得有效性较高；H4 相当于每个用户单独与所有角色的关系进行排列，每个用户与角色的关系排列都是相同的，所以只要测试任一用户与角色的所有关系排列，但是遗漏了用户之间的关系；受本文实验的用户数限制，H5 的 FSM 只有 54 个结点，导致其测试有效性最差，若是深入分析用户数、角色数对 H5 的影响，应可以找出适当的用户数和角色数使得 H5 的有效性达到最佳；H6 旨在减少用户数，将多个用户合并成组，同样由于本文实验的用户数是 2，只能将 2 位用户分成一组，因此相当于只有单个用户与所有角色的关系进行排列，与 H4 的 FSM 相同，故结果也相同。

(下转第 48 页)