Локальная оптимизация политики ролевого разграничения доступа

Н.Ф. Богаченко
nfbogachenko@mail.ru

Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

Аннотация

В статье описываются структурные модификации иерархии ролей в политике ролевого разграничения доступа. Строятся и обосновываются алгоритмы преобразования ролевого графа с учетом различных критериев его оптимальности. При этом рассматриваются такие характеристики, как принципы распространения полномочий в системе, отсутствие дублирующих ролей, особенности ролевого графа, в частности, его древовидность. Основным результатом является доказательство возможности получения нескольких ролевых графов, соответствующих эквивалентным политикам ролевого разграничения доступа.

1 Ролевой граф

Ролевое разграничение доступа (РРД) основывается на запрещении или разрешении действий в системе в целом, без привязки к отдельным объектам системы. Возможность осуществлять такие действия называется *полномочием* (правом, привилегией). Каждая *роль* представляет собой некоторый типовой набор полномочий. Роли ассоциируются с субъектами (*пользователями*) компьютерной системы. При этом одному пользователю может приписываться несколько ролей, и наоборот, на одну роль может быть авторизовано несколько пользователей. В процессе авторизации пользователя на роль он получает и набор полномочий, закрепленный за выданной ролью.

Выделим основные элементы модели РРД, которые понадобятся для дальнейшего описания. Основные множества:

1. $P = \{p_1, \ldots, p_m\}$ – множество полномочий (прав доступа, привилегий) на действия в системе.
2. $R = \{r_1, \ldots, r_n\}$ – множество ролей, возможных в системе.
3. $U = \{u_1, \ldots, u_s\}$ – множество пользователей системы.

Основные отображения:

1. $RP : R \to 2^P$. $RP(r)$ предоставляет набор полномочий роли r. При этом $\forall\, p \in P\ \exists\, r \in R :\ p \in RP(r)$.
2. $RR : R \to 2^R$. $RR(r)$ определяет роли, на которые должна быть авторизована роль r.

In: Sergey V. Belim, Nadezda F. Bogachenko (eds.): Proceedings of the Workshop on Data, Modeling and Security 2017 (DMS-2017), Omsk, Russia, October 2017, published at http://ceur-ws.org

3. $UR : U \to 2^R$. $UR(u)$ определяет набор ролей, на которые может быть авторизован пользователь u. Возможно существование ролей, на которые не авторизован ни один пользователь.

4. $UP : U \to 2^P$. $UP(u)$ задает набор полномочий, необходимых пользователю u для работы в системе.

Заметим, что так как множества P, R и U конечны, то все рассматриваемые дискретные отображения могут быть представлены булевыми матрицами по правилу: если $F : X \to 2^Y$, то элемент матрицы $[\mathbf{F}]_{ij} = 1 \iff y_j \in F(x_i)$.

Ведущую роль в списке основных отображений играет отображение авторизации ролей друг на друга RR, которое должно сформировать *иерархию ролей* – отношение нестрогого частичного порядка на множестве R (*отношение авторизации*). Отношение авторизации обозначим оператором доминирования/подчинения ролей «$\succ$» («$\prec$»): $r_i \succ r_j$ ($r_j \prec r_i$) $\iff r_j \in RR(r_i)$ – в этом случае роль r_i является доминирующей (старшей), а роль r_j – подчиненной (младшей). Рефлективность, антисимметричность и транзитивность отношения авторизации определяют следующие свойства отображения RR:

1. $r_i \in RR(r_i)$.

2. Если $r_j \in RR(r_i) \wedge r_i \in RR(r_j)$, то $r_i = r_j$.

3. Если $r_j \in RR(r_i) \wedge r_k \in RR(r_j)$, то $r_k \in RR(r_i)$.

Множество $RR(r)$ называется *множеством достижимости* роли r.

Основные требования РРД заключаются в том, что отображения RR и RP должны гарантировать выполнение условия *наследования полномочий*:

$$(r_j \prec r_i) \Rightarrow (RP(r_j) \subseteq RP(r_i)), \tag{1}$$

а отображения RR и UR должны обеспечивать выполнение условия *наследования авторизации*:

$$(r_j \prec r_i) \wedge (r_i \in UR(u)) \Rightarrow (r_j \in UR(u)), \tag{2}$$

Заметим, что описанные отображения должны быть *взаимнокорректны*, то есть связаны соотношением:

$$UP(u) \subseteq \bigcup_{r \in UR(u)} RP(r). \tag{3}$$

Рассмотрим более подробно отношение порядка, введенное на множестве ролей. Традиционно для анализа структуры частично упорядоченного множества используется *диаграмма Хассе* – теоретико-графовое представление отношения порядка «$\succ$», в котором дуга (r_i, r_j) существует тогда и только тогда, когда $r_i \succ r_j$ и не существует ориентированного пути $\rho(r_i, r_j)$ такого, что его длина строго больше единицы (в противном случае дуга (r_i, r_j) называется *транзитивной*). Диаграмма Хассе имеет наименьшее число дуг среди всех графов, порождаемых заданным отношением порядка. Диаграмму Хассе называют *транзитивным сокращением* отношения порядка или *транзитивной редукцией* ориентированного графа. В рамках модели РРД также будем использовать графовую интерпретацию иерархии ролей.

Определение 1. Помеченный ориентированный граф $G = (R, E, RP)$ назовем *ролевым графом*, если множество его вершин определяется множеством ролей R; множество дуг порождается отношением авторизации, заданным на множестве ролей (отображением RR); метки вершин определяются отображением RP.

Исходя из вышесказанного, ролевой граф – это граф, порождаемый отображениями RR и RP, и обладающий следующими *характеристиками*:

- Ориентированный.
- Бесконтурный (отсутствуют ориентированные циклы) – в силу антисимметричности отношения порядка.
- Помеченный.
- Выполнено условие наследования меток – условие (1).

При решении ряда задач требуется выполнение еще одного ограничения – ролевой граф должен быть связным. Это свойство возможно получить путем добавления фиктивной вершины, связанной дугами со всеми источниками (вершинами без входящих дуг) ролевого графа.

Заметим, что ролевой граф не обязан являться диаграммой Хассе. Одной и той же иерархии ролей в общем случае можно сопоставить несколько ролевых графов. Такие графы назовем *RR-эквивалентными*. Очевидно, *RR*-эквивалентные ролевые графы отличаются друг от друга множеством дуг. Транзитивные дуги могут появляться в ходе эквивалентных преобразований политики РРД.

Определение 2. Ролевой граф назовем *транзитивно-сокращенным*, если он является диаграммой Хассе.

В большинстве работ посвященных РРД принято считать, что иерархия ролей имеет вид *ориентированного дерева* – бесконтурного ориентированного графа, у которого полустепень захода (число входящих дуг) любой вершины не больше 1 и существует ровно одна вершина (корень), полустепень захода которой равна 0.

Определение 3. Древовидный ролевой граф будем называть *ролевым деревом* и обозначать $T = (R, E, RP)$.

При иерархическом отношении ролей особое внимание уделяется процессу построения отображения RP. Важным является вопрос: возможно ли назначение одного и того же набора полномочий двум ролям, находящимся в иерархическом подчинении. Для построения используется ролевой граф и применяется механизм наследования «снизу – вверх»: расстановка меток (распределение полномочий) начинается с листьев или стоков (вершин без исходящих дуг) ролевого графа. Пусть R_L ($R_L \subseteq R$) – множество стоков, $Ch(r)$ – множество вершин-сыновей вершины r. Возможны два подхода к расстановке меток, порождающих, в свою очередь, две различные *RP-характеристики* ролевого графа.

Определение 4. *Листовой* ролевой граф (*листовая RP*-характеристика). Каждому стоку r отображение RP сопоставляет набор полномочий $RP(r)$ так, чтобы

$$\bigcup_{r \in R_L} RP(r) = P.$$

Метки остальных вершин наследуются:

$$\forall r \in R \setminus R_L : RP(r) = \bigcup_{r' \in Ch(r)} RP(r').$$

Если $\forall r_i, r_j \in R_L$: $RP(r_i) \cap RP(r_j) = \varnothing$, то листовой ролевой граф называется *таксономическим*, иначе – *нетаксономическим*. Если $\forall r \in R_L : |RP(r)| = 1$, то листовой ролевой граф называется *единичным*, иначе – *общим*.

Определение 5. *Охватный* ролевой граф (*охватная RP*-характеристика). Допускается включение в наборы полномочий вершин «добавочных» полномочий, ненаследуемых от сыновей. Это означает, что

$$\bigcup_{r \in R_L} RP(r) \subseteq P.$$

Определение 6. Роли r_i и r_j назовем *RP-эквивалентными*, если они наделены одинаковыми наборами полномочий: $\forall r_i, r_j \in R : (r_i \overset{RP}{\backsim} r_j) \Longleftrightarrow (RP(r_i) = RP(r_j))$. Тем самым множество ролей R разбивается на классы эквивалентности – *RP-классы*.

Определение 7. Ролевой граф назовем *RP-сокращенным*, если каждый его *RP*-класс содержит ровно одну роль (метки всех вершин различны).

Обобщая вышесказанное, можно выделить следующие *дополнительные характеристики* ролевого графа:

- Листовой (в том числе единичный и/или таксономический).
- *RP*-сокращенный.

- Древовидный.
- Транзитивно-сокращенный.

Последнее свойство непосредственно касается только ролевого графа и никак не характеризует иерархию ролей.

Для разработки алгоритмов преобразования ролевого графа и оценки их трудоемкости необходимо учитывать особенности представления (описания) графов.

Для задания наборов полномочий ролей (меток вершин ролевого графа) удобно использовать векторное представление: метка вершины r – это m-мерный битовый вектор (массив, строка) $r.p$:

$$[r.p]_k = \begin{cases} 1, & p_k \in RP(r) \\ 0, & p_k \notin RP(r) \end{cases}, \ (k = 1, \ldots, m). \tag{4}$$

Рассматривая способы описания графа, необходимо различать файловое и программное (внутреннее) представления. При программно-технической реализации политики РРД возникает подзадача файлового представления ролевого графа, одновременно удобного и для машинной обработки, и для восприятия человеком. Возможны два подхода: разработка собственного языка представления помеченного ориентированного графа или выбор одного из общепризнанных стандартов для представления теоретико-графовых моделей. Предпочтение следует отдать второму подходу, так как использование стандартизованного языка для описания ролевого графа позволит сократить время разработки и обеспечит совместимость со многими прикладными программами и библиотеками для работы с графами. В большинстве стандартов граф задается двумя списками [1, 2]:

1. Список вершин графа; каждой вершине приписана метка – битовая строка длины m (см. формулу 4).
2. Список дуг графа; каждая дуга задана начальной и конечной вершинами.

Формальное описание алгоритмов на графах удобно вести в терминах абстрактного типа данных (АТД) «Граф». Интерфейс этого АТД должен включать следующие операции (методы) работы с графом:

- добавить / удалить вершину;
- добавить / удалить ребро;
- задать метку вершины;
- «склеить» / «стянуть» вершины [1] и т. д.

Кроме того, для обработки всех вершин, смежных с указанной, как правило используется АТД «Итератор». Если I_i – итератор, созданный для вершины r_i графа G, то: метод $I_i.begin()$ возвращает номер первой вершины, смежной с вершиной r_i; метод $I_i.next()$ переходит к следующей вершине, смежной с r_i, и возвращает ее номер; метод $I_i.end()$ возвращает 1, если есть непройденные вершины, смежные с r_i, и 0 – в противном случае. Тогда, например, операция «обойти все вершины, смежные с вершиной r_i» может быть реализована в виде цикла:

$$for(j = I_i.first(); \ !I_i.end(); \ j = I_i.next())$$

Реализация интерфейса АТД «Граф» зависит от способа представления данных.

Одним из стандартов программного (внутреннего) представления графа являются *списки смежности*. Пусть n – число вершин графа, S – n-мерный вектор (массив) списков смежности. Элемент $[S]_i$ вектора S соответствует вершине r_i графа и содержит список тех вершин, в которые ведут дуги из вершины r_i (список смежности). На этом уровне детализации операция «обойти все вершины, смежные с вершиной r_i» реализуется как обход (просмотр) списка смежности $[S]_i$.

Матрица смежности графа $\mathbf{M}$ – другой способ представления данных внутри АТД «Граф». Матрица имеет размерность $n \times n$. Элемент $[\mathbf{M}]_{ij}$ матрицы смежности равен 1, если в орграфе существует дуга (r_i, r_j), и 0 – в противном случае. Для одних алгоритмов эффективнее списочная реализация графа, для других – матричная.

[1] Здесь и далее операции «склейки» и «стягивания» вершин понимаются в соответствии с определениями теории графов.

Еще одной часто используемой структурой является *матрица достижимости* графа $\mathbf{M}^+$ (или *транзитивное замыкание* матрицы смежности). Эта матрица также имеет размерность $n \times n$. Напомним, что элемент $[\mathbf{M}^+]_{ij}$ матрицы достижимости равен 1, если в орграфе существует ориентированный путь из вершины r_i в вершину r_j, и 0 – в противном случае. Для построения матрицы достижимости используется алгоритм Уоршелла [3]. Несложно заметить, что $\mathbf{M}^+$ представляет собой матрицу $\mathbf{RR}$ отображения авторизации ролей RR.

Часть алгоритмов работы с ролевым графом будет описана на уровне АТД «Граф», часть – на уровне внутреннего представления данных[2].

2 Локальная оптимизация РРД

2.1 Эквивалентные преобразования ролевого графа

Под оптимизацией РРД будем понимать преобразования подсистемы безопасности информационной системы, повышающие эффективность ее функционирования. Эффективность может определяться одним или несколькими параметрами, такими как производительность, риски утечки полномочий и т.д. Оптимизация РРД может быть достигнута с помощью преобразования информационных структур, связанных с политикой разграничения доступа. При этом данные изменения должны быть прозрачными для пользователя. Отдельно взятый пользователь должен получать одни и те же полномочия до и после преобразований. Если две политики РРД предоставляют пользователям одни и те же полномочия, то они могут считаться эквивалентными. Более строго данный принцип может быть сформулирован следующим образом.

Предположение 1. *Две политики РРД эквивалентны, если у них совпадают множества P и U, а также отображения UP изоморфны.*

Если в результате преобразований РРД будет получена политика разграничения доступа эквивалентная исходной, то можно говорить об *эквивалентном преобразовании* РРД. Следует отметить, что эквивалентность двух политик РРД накладывает ограничения только на отображение UP. Отображения RP, RR и UR могут отличаться. В реальных системах из требований непрерывности функционирования информационной системы как правило присутствуют ограничения и на изменения отображений RP, RR и UR. Наиболее распространенным является требование минимизации изменений вносимых в данные отображения.

Определение 8. Эквивалентное преобразование РРД, вносящее минимальные изменения в отображения RP, RR и UR и направленное на оптимизацию РРД, будем называть *локальной оптимизацией*.

Локальная оптимизация РРД по сути представляет собой преобразование ролевого графа. Критерий «минимальные изменения отображений RP, RR и UR» является эвристическим и трудно поддается формализации. В дальнейшем будем руководствоваться следующими рассуждениями.

Одним из основных требований к построению политики РРД является выполнение условия (2), касающееся отображения UR: вместе с заданной ролью пользователь должен быть авторизован и на все подчиненные роли. Основными каналами утечки информации в политике РРД являются «избыточные полномочия», получаемые пользователем. Поэтому в процессе локальной оптимизации РРД:

1. Изменение подмножества ролей, подчиненных текущей роли, точнее семейства наборов полномочий подчиненных ролей, по возможности должно быть минимальным.

2. Не должно возникнуть ситуации, при которой пользователю для получения требуемого отображением UP набора полномочий пришлось бы авторизоваться на новую роль с более широкими возможностями.

Введем ряд обозначений и определений. Пусть F – преобразование ролевого графа, определяющее изменение множества ролей и ролевой иерархии:

$$F(R, RR, RP, UR) = (R', RR', RP', UR').$$

[2] Дальнейшая детализация типов данных и методов работы с ними будет зависеть от особенностей объектной модели выбранного языка программирования. Примеры реализации АТД «Граф» на языке C++ можно найти в работе [4].

Определение 9. Преобразование ролевого графа G в ролевой граф G' назовем *RP-допустимым*, если:

1. $\forall r \in R\ \exists r' \in F(R)$: $RP(r) = RP'(r')$. Другими словами, множества RP-классов этих графов связаны отношением включения: $(G/\overset{RP}{\backsim}) \subseteq (G'/\overset{RP'}{\backsim})$.

2. Для любого ориентированного пути $\rho(r_i, r_j)$ в графе G найдется ориентированный путь $\rho(r_u, r_v)$ в графе G' такой, что $RP(r_i) = RP'(r_u)$ и $RP(r_j) = RP'(r_v)$.

Иногда возможно построить ролевой граф G' удовлетворяющий более сильному требованию.

Определение 10. RP-допустимое преобразование ролевого графа G в ролевой граф G' назовем *RP-эквивалентным*, если:

1. Множества RP-классов этих графов совпадают: $(G/\overset{RP}{\backsim}) = (G'/\overset{RP'}{\backsim})$.

2. Для любого ориентированного пути в одном из графов найдется ориентированный путь в другом графе такой, что совпадают метки начальных вершин и совпадают метки конечных вершин.

Из условия (3) о взаимной корректности основных отображений РРД следует:

Утверждение 1. *RP-допустимое (RP-эквивалентное) преобразование ролевого графа приводит к построению эквивалентной политики РРД.*

Исходя из вышесказанного, будем считать, что преобразование F ролевого графа G в ролевой граф G' представляет собой локальную оптимизацию РРД, если:

1. Для графа G' выполнены требования выбранного критерия оптимальности.
2. F – RP-эквивалентное (или RP-допустимое) преобразование.
3. Число вершин и/или дуг ролевого графа либо не увеличилось, либо это увеличение минимально.

Согласно выделенным в разделе 1 дополнительным характеристикам ролевого графа, рассмотрим следующие критерии локальной оптимизации.

1. «[Единичный] листовой РГ»: [единичный] листовой ролевой граф является оптимальным.
2. «RP-сокращенный РГ»: RP-сокращенный ролевой граф является оптимальным.
3. «Древовидный РГ»: древовидный ролевой граф является оптимальным.
4. «Транзитивно-сокращенный РГ»: транзитивно-сокращенный ролевой граф является оптимальным.

Далее будут представлены методики и алгоритмы локальной оптимизации РРД в соответствии с указанными критериями. Эти подходы частично изложены в работах [5, 6]. В данной статье они будут расширены и систематизированы.

2.2 Листовой ролевой граф

Теорема 1. *Существует RP-допустимое преобразование ролевого графа в единичный листовой ролевой граф.*

Доказательство. Пусть G – произвольный ролевой граф. Построим ролевой граф G' по следующему алгоритму. Все вершины, дуги и полномочия графа G перенесем в граф G'.

Далее осуществим обход вершин графа G. Под обходом графа понимается просмотр всех его вершин в некоторой последовательности. В данном алгоритме обход графа произволен. При этом будем различать листовые (стоковые) и нелистовые вершины. Пусть листовой вершине r_i сопоставлен набор полномочий $P_i = \{p_{i1}, \ldots, p_{im_i}\}$. Если $|P_i| = m_i > 1$, то в графе G' к этой вершине присоединим m_i листовых вершин, каждая из которых будет наделена полномочием p_{ij} ($j \in \{1, \ldots, m_i\}$). Каждую нелистовую вершину r в графе G' пополним сыновьями-листьями по числу полномочий из набора $RP(r)$ графа G, которые не были унаследованы (каждой новой вершине припишем соответствующее полномочие).

Граф G' является ролевым по построению. В графе G' каждая нелистовая вершина не получает ни одного полномочия непосредственно, а лишь наследует их от сыновей. Каждой листовой вершине графа G' приписано одно единственное полномочие. Следовательно, G' – единичный листовой ролевой граф. Так как G является подграфом графа G', то представленное преобразование ролевого графа является RP-допустимым ■

Следствие 1.1. *Существует RP-допустимое преобразование ролевого графа в листовой ролевой граф.*

Доказательство. Принцип построения листового ролевого графа аналогичен алгоритму, представленному в доказательстве предыдущей теоремы. Отличие заключается в том, что листовые вершины сыновьями не пополняются, а каждая нелистовая вершина при необходимости пополняется одним сыном-листом с полномочиями, которые не были унаследованы. ■

Следствие 1.2. *Рассмотренные в теореме 1 и следствии 1.1 преобразования ролевого графа являются локальной оптимизацией РРД.*

Доказательство. Из доказательства теоремы следует, что изменения, вносимые в иерархию ролей, ведут к минимальному «разрастанию» графа. ■

Замечание 1. Доказательства теоремы 1 и следствия 1.1 конструктивны и определяют алгоритмы локальной оптимизации РРД в соответствии с критерием «[единичный] листовой РГ».

Замечание 2. Очевидно, алгоритмы локальной оптимизации РРД в соответствии с критерием «[единичный] листовой РГ» сохраняют древовидность ролевого графа.

Заметим, что алгоритмы построения [единичного] листового ролевого графа не единственны. В представленном подходе избавление от «охватности» распределения полномочий происходит только за счет добавления новых вершин, метки которых соответствуют неунаследованным полномочиям. Можно было бы попытаться унаследовать эти полномочия от уже имеющихся вершин. Но такой подход также не однозначен и может привести не только к потере древовидности, но и к существенном изменению множества *RP*-классов.

2.3 *RP*-сокращенный ролевой граф

В процессе локальной оптимизации РРД в соответствии с критерием «[единичный] листовой РГ» может увеличиться не только количество ролей и *RP*-классов, но и мощность самих *RP*-классов. Последнее свидетельствует о наличии в системе «дублирующих» ролей. В ряде случаев требование отсутствия «дублирующих» ролей является существенным. Тогда необходимо гарантировать *RP*-сокращенность ролевого графа, быть может отказавшись от листового принципа распределения полномочий или от древовидной структуры иерархии ролей.

Теорема 2. *Существует RP-эквивалентное преобразование ролевого графа в RP-сокращенный ролевой граф.*

Доказательство. В ролевом графе G достаточно «склеить» вершины-роли, попадающие в один *RP*-класс, если они не соединены дугами, либо попарно «стянуть», если такие дуги имеются. Полученный граф G' будет ролевым по построению. Множество *RP*-классов останется прежним, но граф G' будет *RP*-сокращенным. Очевидно, что для любого ориентированного пути в одном из графов найдется ориентированный путь в другом графе такой, что совпадают метки начальных вершин и совпадают метки конечных вершин. Следовательно, представленное преобразование ролевого графа является *RP*-эквивалентным. ■

Следствие 2.1. *Рассмотренное в теореме 2 преобразование ролевого графа является локальной оптимизацией РРД.*

Доказательство. В процессе «склейки» и «стягивания» вершин графа число вершин и дуг не увеличивается. ■

Замечание 3. Доказательство теоремы 2 конструктивно и определяет алгоритм локальной оптимизации ролевого графа в соответствии с критерием «*RP*-сокращенный РГ».

Замечание 4. Несложно понять, что алгоритм локальной оптимизации ролевого графа в соответствии с критерием «*RP*-сокращенный РГ» сохраняет листовой принцип распределения полномочий. Более того, если исходный ролевой граф был единичным листовым, то результирующий станет единичным таксономическим листовым. Поэтому данный алгоритм целесообразно применять после алгоритма построения единичного листового ролевого графа (который может привести к появлению вершин с одинаковыми метками).

Теорема 3. *Существует RP-допустимое преобразование ролевого графа в единичный таксономический листовой RP-сокращенный ролевой граф.*

Доказательство. С учетом замечания 4 достаточно последовательно применить преобразования, описанные в теоремах 1 и 2. ■

Следствие 3.1. *Рассмотренное в теореме 3 преобразование ролевого графа является локальной оптимизацией РРД.*

2.4 Ролевое дерево

Для доказательства следующей теоремы требуется, чтобы ролевой граф имел ровно один источник (вершину без входящих дуг). В противном случае ролевой граф может быть пополнен фиктивной вершиной-супериcточником s, которая соединяется дугами со всеми существующими источниками $s_1, \ldots, s_k$. Метка супериcточника формируется по правилу наследования: $RP(s) = RP(s_1) \cup \ldots \cup RP(s_k)$. Заметим, что так как в ролевом графе отсутствуют ориентированные циклы, то требование единственности источника влечет за собой связность ролевого графа.

Теорема 4. *Существует RP-эквивалентное преобразование ролевого графа с единственным источником в ролевое дерево.*

Доказательство. Пусть дан ролевой граф G. RP-эквивалентное ему ролевое дерево T будем формировать последовательно. В начале каждому стоку r_{t_j} графа G сопоставляем $d^+(r_{t_j})$ листьев в графе T: «оригинал» и $(d^+(r_{t_j}) - 1)$ «ярлыков» (здесь и далее $d^+(r)$ – полустепень захода вершины r). Эта операция называется *расщеплением* вершины. Если полустепень захода равна единице, то имеется только «оригинал».

Далее, двигаясь по графу G от нижних ярусов к источнику в порядке возрастания длины самого протяженного из путей от текущей вершины до стоков, последовательно расщепляем все вершины. «Оригинал» и «ярлыки» наделяем теми же полномочиями, что были у вершины их образующей. К «оригиналу» присоединяем уже существующие вершины графа T из тех, что не имеют входящих дуг, восстанавливая сыновей расщепляемой вершины графа G (такие вершины в T всегда найдутся по построению). К каждому «ярлыку» добавляем вершины и дуги так, чтобы подграф, порожденный «ярлыком», представлял собой копию поддерева, порожденного «оригиналом».

Очевидно, что построенный таким образом граф T является ролевым деревом и имеет те же RP-классы, что и исходный ролевой граф G. Кроме того для любого ориентированного пути в одном из графов найдется ориентированный путь в другом графе такой, что совпадают метки начальных вершин и совпадают метки конечных вершин. Таким образом рассмотренное преобразование ролевого графа является RP-эквивалентным. ■

Следствие 4.1. *Рассмотренное в теореме 4 преобразование ролевого графа является локальной оптимизацией РРД.*

Доказательство. Предложенный в доказательстве порядок обхода вершин графа G позволит минимизировать увеличение числа вершин и дуг. ■

Следствие 4.2. *Число вершин n' результирующего ролевого дерева T не превосходит $\mathrm{O}(n^3)$, где n – число вершин исходного ролевого графа.*

Доказательство. Пусть F – рассматриваемое преобразование ролевого графа. По построению

$$n' = n + \sum_{r \in R} \left(d^+(r) - 1\right) |RR'(r')|,$$

где $r' \in F(r)$ – произвольна. Так как поддеревья, присоединяемые к «ярлыкам», восстанавливают сыновей расщепляемой вершины исходного графа, то $|RR'(r')| \leq n$. Следовательно:

$$n' \leq n + \sum_{r \in R} \left(d^+(r) - 1\right) n \leq n + (n-1)\, n^2 = \mathrm{O}(n^3).$$

■

Замечание 5. Доказательство теоремы 4 конструктивно и определяет алгоритм локальной оптимизации ролевого графа в соответствии с критерием «древовидный РГ».

Замечание 6. Несложно понять, что алгоритм локальной оптимизации ролевого графа в соответствии с критерием «древовидный РГ» сохраняет [единичный] листовой принцип распределения полномочий. Учитывая замечание 2, алгоритмы локальной оптимизации в соответствии с критериями «древовидный РГ» и «[единичный] листовой РГ» можно применять в любом порядке и получать [единичное] листовое ролевое дерево. Очевидно, что алгоритм построения ролевого дерева имеет большую трудоемкость, поэтому его целесообразно применять в первую очередь.

Теорема 5. *Существует RP-допустимое преобразование ролевого графа с единственным источником в [единичное] листовое ролевое дерево.*

Доказательство. С учетом замечания 6 достаточно последовательно применить преобразования, описанные в теоремах 4 и 1. ■

Следствие 5.1. *Рассмотренное в теореме 5 преобразование ролевого графа является локальной оптимизацией РРД.*

2.5 Транзитивно-сокращенный ролевой граф

Построение транзитивно-сокращенного ролевого графа заключается в получении графа, описывающего исходную иерархию ролей но не содержащего транзитивные дуги. Отсюда следует:

Теорема 6. *Переход к транзитивно-сокращенному ролевому графу является RP-эквивалентным преобразованием.*

Следствие 6.1. *Переход к транзитивно-сокращенному ролевому графу является локальной оптимизацией РРД.*

Замечание 7. Очевидно, что если ролевой граф древовидный, то он не имеет транзитивных дуг и является транзитивно-сокращенным. В общем случае переход к транзитивно-сокращенному ролевому графу не изменяет метки вершин. Поэтому для упрощения ролевой иерархии оптимизация ролевого графа в соответствии с критерием «транзитивно-сокращенный РГ» должна предшествовать другим видам оптимизации. Вместе с тем, оптимизация в соответствии с критерием «RP-сокращенный РГ» может привести к появлению транзитивных дуг. В этом случае следует вновь перейти к транзитивно-сокращенному графу.

2.6 Алгоритмы и оценка трудоемкости

По-прежнему метку вершины r будем представлять m-мерным битовым вектором $r.p$ (см. формулу (4)). Пусть для битовых векторов перегружены следующие операции: $\vee$ – побитовая дизъюнкция, $\oplus$ – побитовое сложение по mod 2. Нулевой вектор обозначим $\vec{0}$. Заметим, что вектор $r_i.p \oplus r_j.p$ предоставляет полномочия, которыми различаются вершины r_i и r_j. Следовательно, вектор

$$r.p \oplus \{ \bigvee_{r' \in Ch(r)} r'.p \} \tag{5}$$

определяет полномочия, которые вершина r получает непосредственно, а не за счет наследования ($Ch(r)$ – множество вершин-сыновей вершины r). Для проверки того, что ролевой граф является RP-сокращенным, удобно использовать свойство меток:

$$r_i.p = r_j.p \Leftrightarrow r_i.p \oplus r_j.p = \vec{0}. \tag{6}$$

Пусть ролевой граф задан n-мерным вектором списков смежности S. Элемент $S[i]$ вектора S соответствует вершине r_i графа и содержит следующие поля[3]:

1) $S[i].list$ – целочисленный список номеров тех вершин, в которые ведут дуги из вершины r_i (список смежности);

2) $S[i].p$ – метка вершины r_i.

[3] Для i-й координаты вектора V наряду с обозначением $[V]_i$ будем использовать запись $V[i]$. Аналогично для матриц: $[\mathbf{M}]_{ij} = \mathbf{M}[i,j]$.

2.6.1 Алгоритм оптимизации ролевого графа по критерию «единичный листовой РГ»

Создать копию S' вектора S. Последовательно просмотреть все элементы вектора S. Если список смежности $S[j].list$ очередного элемента пуст (вершина r_j является стоком или листом) и вектор $S[j].p$ содержит более одной единичной координаты, то:

– в S' добавить новые элементы по числу единичных координат вектора $S[j].p$,

– приписать новым элементам метки, каждая из которых определяется одной единичной координатой вектора $S[j].p$,

– пополнить список смежности $S'[j].list$ новыми элементами.

Если список смежности $S[j].list$ очередного элемента не пуст (вершина r_j нелистовая), то сформировать m-мерный битовый вектор V в соответствии с формулой (5): при этом просмотр всех сыновей вершины r_j заключается в просмотре списка смежности $S[j].list$. Если $V \neq \vec{0}$, то:

– в S' добавить новые элементы по числу единичных координат вектора V,

– приписать новым элементам метки, каждая из которых определяется одной единичной координатой вектора V,

– пополнить список смежности $S'[j].list$ новыми элементами.

Утверждение 2. *Трудоемкость алгоритм оптимизации ролевого графа по критерию «единичный листовой РГ» не превосходит* $\mathrm{O}(m \cdot n^2))$*, где* n *– число ролей,* m *– число полномочий.*

Доказательство. Число шагов алгоритма оценивается сверху величиной $n(\mathrm{O}(m) + \mathrm{O}(m \cdot n)) = \mathrm{O}(m \cdot n^2)$. ■

Алгоритм оптимизации ролевого графа по критерию «листовой РГ» отличается тем, что просматриваются только те элементы $S[j]$ вектора S, для которых список смежности $S[j].list$ не пуст. И если найден элемент с ненулевым вектором V, то: в S' добавляется один новый элемент, ему приписывается метка V, список смежности $S'[j].list$ пополняется этим новым элементом. Очевидно, что трудоемкость этого алгоритма имеет ту же оценку, что и для алгоритма построения единичного листового ролевого графа.

2.6.2 Алгоритм оптимизации ролевого графа по критерию «RP-сокращенный РГ»

Пока в S найдется два элемента $S[i]$ и $S[j]$ такие, что $S[i].p \oplus S[j].p = \vec{0}$ (см. формулу (6)), выполнять:

Если индекс i содержится в списке $S[j].list$ или индекс j содержится в списке $S[i].list$ (вершины r_i и r_j смежны), то для вершин r_i и r_j выполнить операцию «склеить» АТД «Граф».

Иначе для этих вершин выполнить операцию «стянуть» АТД «Граф».

Утверждение 3. *Трудоемкость алгоритма оптимизации ролевого графа по критерию «RP-сокращенный РГ» не превосходит* $\mathrm{O}(m \cdot n^4))$*, где* n *– число ролей,* m *– число полномочий.*

Доказательство. Число шагов алгоритма оценивается сверху величиной $n^2 \cdot (\mathrm{O}(m) \cdot \mathrm{O}(n^2)) = \mathrm{O}(m \cdot n^4)$. ■

2.6.3 Алгоритм оптимизации ролевого графа по критерию «древовидный РГ»

Для реализации алгоритма необходимо сформировать два вектора. D – n-мерный целочисленный вектор. Элемент $D[i]$ вектора D – это полустепень захода (число входящих дуг) вершины r_i ролевого графа. Алгоритм формирования вектора D:

1. Вектор D инициализировать нулями.

2. Для каждого $i = 1, \ldots, n$ обойти список смежности $S[i].list$.

3. Для каждого j – очередного элемента i-го списка смежности: $D[j] := D[j] + 1$.

Q – n-мерный вектор. Элемент $Q[i]$ вектора Q состоит из двух целочисленных полей:

1) $Q[i].r$ – номер вершины графа;

2) $Q[i].l$ – длина самого протяженного из ориентированных путей от вершины с номером $Q[i].r$ до стоков (для дерева – до листовых вершин).

Алгоритм формирования вектора Q:

1. Реализовать модифицированный алгоритм Флойда [3]: на вход алгоритму подается вектор списков смежности S, на выходе формируется матрица $\mathbf{H}$ размерности $n \times n$ такая, что $\mathbf{H}[i, j]$ есть длина самого

протяженного ориентированного пути из вершины r_i в вершину r_j (под длиной маршрута понимается число входящих в него дуг; если вершина r_j не достижима из вершины r_i, то $\mathbf{H}[i,j] := -1$).

2. Начальная инициализация: $Q[i].r := i$, $Q[i].l := 0$ $(i = 1, \ldots, n)$.

3. Для каждого $i = 1, \ldots, n$: $Q[i].l := \max_{(j=1,\ldots,n)\wedge(S[j].list=0)}\{\mathbf{H}[i,j]\}$.

4. Упорядочить элементы вектора Q по неубыванию величины $Q[i].l$.

Очевидно, что вершины-стоки (листья) расположены в начале вектора Q, далее следуют вершины, связанные со стоками только дугой, и т.д.

Описание алгоритма удобно провести в теоретико-графовой постановке, а затем пояснить особенности программной реализации. Пусть задан ролевой граф с одним источником G. RP-эквивалентное ему ролевое дерево T формируется последовательно.

1. На каждом шаге строится ориентированный ациклический граф T^i. На начальном этапе граф T^0 пуст. Число шагов алгоритма равно порядку n графа G, то есть $T^n = T$. В процессе построения дерева в определенной последовательности обходятся все вершины орграфа G и на каждом шаге i $(i = 1, \ldots, n)$ граф T^i пополняется вершинами и дугами.

2. В начале просматриваются стоки графа G. Каждому стоку r_{t_j} сопоставляется $d^+(r_{t_j})$ листьев в T: вершина-оригинал и $(d^+(r_{t_j}) - 1)$ вершин-ярлыков. Оригиналу и ярлыкам приписывается метка вершины r_{t_j}. После просмотра всех l стоков $r_{t_1}, \ldots, r_{t_l}$ графа G, граф T^l представляет собой $\sum_{j=1}^{l} d^+(r_{t_j})$ изолированных вершин.

3. Далее обход вершин графа G осуществляется в порядке возрастания длины самого протяженного из путей от текущей вершины до стоков $r_{t_1}, \ldots, r_{t_l}$. Пусть r_i – очередная вершина графа G с меткой $r_i.p$, просматриваемая на i-том шаге алгоритма, и $d^+(r_i) = k_i$. В графе T^i ей сопоставляется k_i вершин: вершина-оригинал o_i^1 и $(k_i - 1)$ ярлыков $o_i^2, \ldots, o_i^{k_i}$. Вершинам $o_i^1, \ldots, o_i^{k_i}$ приписывается метка $r_i.p$.

4. Вершина-оригинал o_i^1 соединяется дугами с уже существующими вершинами графа T^i по правилу: дуга (o_i^1, o_s^j) присоединяется к графу T^i тогда и только тогда, когда $d^+(o_s^j) = 0$ и вершине o_s^j соответствует такая вершина r_s в графе G, что в G существует дуга (r_i, r_s). Другими словами среди вершин графа T^i выбираются вершины, соответствующие непосредственным потомкам вершины r_i в графе G, которые и становятся сыновьями вершины o_i^1.

5. Для каждого ярлыка $o_i^2, \ldots, o_i^{k_i}$ добавляются новые вершины-ярлыки и дуги так, чтобы подграф $T_{o_i^j}$ $(j = 2, \ldots, k_i)$, порожденный ярлыком o_i^j и всеми его потомками, представлял собой копию поддерева $T_{o_i^1}$, порожденного вершиной-оригиналом o_i^1 и всеми ее потомками; метки вершин заменены порядковыми номерами, в графе T вершины-оригиналы закрашены, вершины-ярлыки – нет).

Обход графа G следует вести в порядке, определяемом вектором Q: индекс очередного элемента вектора S лежит в $Q[i].r$ $(i = 1, \ldots, n)$. Для удобства далее очередной индекс будем обозначать i.

Дерево T также представляется вектором списков смежности ST. Но набор полей элементов вектора ST следует расширить. Каждый j-й элемент вектора ST должен содержать:

1) $ST[j].list$ – целочисленный список номеров тех вершин, в которые ведут дуги из j-й вершины (список смежности);

2) $ST[j].p$ – метка вершины – m-мерный битовый вектор.

3) $ST[j].ind$ – индекс вершины исходного графа G, которой была сопоставлена данная вершина в графе T;

4) $ST[j].d$ – полустепень захода вершины.

Кроме того, вектор ST должен динамически наращивать размерность, так как заранее число вершин дерева T не известно. Изначально вектор ST пуст. Процесс сопоставления вершине r_i графа G вершины-оригинала и ярлыков в графе T будет заключаться в следующем.

Для построения вершины-оригинала o_i^1 в вектор ST добавляется очередной элемент j:

$$ST[j].p := S[i].p,\ ST[j].ind := i,\ ST[j].d := 0,\ ST[j].list := 0$$

Чтобы построить исходящие из вершины-оригинала дуги, необходимо для каждого элемента k списка смежности $S[i].list$ пройти по вектору ST, чтобы найти индекс l такой, что $ST[l].ind = k$ и $ST[l].d = 0$. Найденный индекс l надо добавить в список смежности $ST[j].list$, а также $ST[l].d := 0$.

Для построения поддеревьев с корневыми вершинами-ярлыками формируется $D[i]-1$ новых векторов списков смежности $ST_2, \ldots, ST_{D[i]}$, подобных вектору ST. Далее можно реализовать обход «в ширину» всех вершин, достижимых из вершины $ST[j]$, и сформировать векторы $ST_2, \ldots, ST_{D[i]}$ как копии элементов, пройденных в векторе ST. Заполненные векторы $ST_2, \ldots, ST_{D[i]}$ присоединяются к вектору ST с учетом перенумерации новых вершин.

Более детальное описание алгоритма зависит от выбранного языка программирования и от способа реализации динамических структур данных.

Утверждение 4. *Трудоемкость алгоритм оптимизации ролевого графа по критерию «древовидный РГ» полиномиально зависит от числа вершин и числа полномочий.*

Доказательство. Обоснованием является следствие 4.2. ■

2.6.4 Алгоритм оптимизации ролевого графа по критерию «транзитивно-сокращенный РГ»

Для реализации этого алгоритма используется матричное представление ролевого графа. Известно, что транзитивное сокращение $\mathcal{R}^-$ отношения порядка $\mathcal{R}$ можно найти, используя его транзитивное замыкание $\mathcal{R}^+$ и операции разности «$-$» и композиции «$\circ$» отношений: $\mathcal{R}^- = \mathcal{R} - (\mathcal{R} \circ \mathcal{R}^+)$. Переходя от отношений порядка к ориентированным графам, получаем, что матрица смежности $\mathbf{M}^-$ диаграммы Хассе ролевого графа может быть вычислена через матрицы смежности $\mathbf{M}$ и достижимости $\mathbf{M}^+$ исходного ролевого графа:

$$\mathbf{M}^- = \mathbf{M} - (\mathbf{M} \circ \mathbf{M}^+).$$

При этом композиция и разность понимаются как «булево произведение» и «булева разность» бинарных матриц:

$$(\mathbf{M_1} \circ \mathbf{M_2})[i,j] = \mathbf{M_1}[i,1] \wedge \mathbf{M_2}[1,j] \vee \ldots \vee \mathbf{M_1}[i,n] \wedge \mathbf{M_2}[n,j],$$
$$(\mathbf{M_1} - \mathbf{M_2})[i,j] = \mathbf{M_1}[i,j] \wedge \overline{\mathbf{M_2}[i,j]}.$$

Утверждение 5. *Трудоемкость алгоритма оптимизации ролевого графа по критерию «транзитивно-сокращенный РГ» не превосходит* $\mathrm{O}(n^3)$*, где* n *– число ролей.*

Доказательство. Очевидно, что искомая трудоемкость складывается из трудоемкости алгоритма построения матрицы достижимости и трудоемкости операций композиции и разности матриц. Алгоритм Уоршелла, используемый для построения матрицы достижимости, имеет трудоемкость $\mathrm{O}(n^3)$ [3]. Трудоемкость операций композиции и разности матриц равна $\mathrm{O}(n^3)$ и $\mathrm{O}(n^2)$, соответственно. ■

2.7 Выводы

Удалось показать, что одной и той же политике РРД может соответствовать несколько ролевых графов. Причем возможна проверка эквивалентности ролевых графов с точки зрения распределения полномочий. Этот факт является существенным в развитии модели РРД, так как позволяет строить различные представления иерархии ролей, оптимальные в различных смыслах (см. табл. 1).

Таблица 1: Алгоритмы локальной оптимизации

Критерий	Оптимальный ролевой граф	Сохраняется
1	Листовой	Древовидность
2	*RP*-сокращенный	Листовое распределение полномочий
3	Древовидный	Листовое распределение полномочий
4	Транзитивно-сокращенный	Вся иерархия
1+2	Листовой + *RP*-сокращенный	
3+1	Листовой + Древовидный	

Список литературы

[1] URL: `http://graphml.graphdrawing.org/primer/graphml-primer.html`.

[2] N. F. Bogachenko. The Mutual Exclusion Relation on a Set of Roles in Access Control Models. *CEUR Workshop Proceedings*, 1732, 2016. URL: `http://ceur-ws.org/Vol-1732/paper4.pdf`.

[3] A. V. Aho, J. E. Hopcroft, J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1987.

[4] URL: http://www.intuit.ru/studies/courses/12181/1174/lecture/25264

[5] S. Belim, N. Bogachenko, E. Ilushechkin. An Analysis of Graphs that Represent a Role-Based Security Policy Hierarchy *Journal of Computer Security*, 23(5):641–657, 2015. URL: http://content.iospress.com/articles/journal-of-computer-security/jcs532.

[6] S. V. Belim, N. F. Bogachenko Distribution of Cryptographic Keys in Systems with a Hierarchy of Objects. *Automatic Control and Computer Sciences*, 50(8):777–786, 2016. URL: http://link.springer.com/article/10.3103/S0146411616080071.

Local Optimization of the Role-Based Access Control Policy

Nadezda F. Bogachenko

Modifications of the role hierarchy in the role-based access control policy are described in this paper. The role-graph conversion algorithms wich use different optimality criterions are constructed and proved. The principles of the permission distribution, the absence of the roles-counterparts and the role-graph acyclicity are considered as the optimality criterions. A possibility to obtain the several role-graphs wich correspond to the equivalent role-based access control models is prooved.