

Автоматизация процессов управления политикой разграничением доступа

Н.Ф. Богаченко
nfbogachenko@mail.ru

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

Аннотация

Описывается библиотека утилит, предназначенных для автоматизации процессов управления политикой разграничения доступа крупномасштабных информационных систем. Особое внимание уделяется форматам входных и выходных данных. Используется описание данных на основе языка XML. Также применяется формат GraphML.

1 Общее описание библиотеки утилит

Необходимость создания *библиотеки утилит управления разграничением доступа* (БУ УРД) обусловлена развитием и внедрением крупномасштабных информационных систем (КМИС) [1]. Использование БУ УРД позволит решать следующие задачи:

1. Локальная оптимизация иерархии ролей в соответствии с различными критериями оптимальности.
2. Построение и оптимизация иерархии ролей в соответствии с заданным распределением полномочий между пользователями.
3. Выбор оптимальной роли для авторизации пользователя.
4. Расчёт уровня критичности полномочий, исходя из анализа иерархии ролей.
5. Определение основных характеристик решётки ценностей.
6. Определение доступов субъектов к объектам по имеющимся прецедентам.
7. Совмещение ролевого и мандатного принципов разграничения доступа.
8. Совмещение мандатного и дискреционного принципов разграничения доступа.
9. Реализация последовательного доступа к иерархически организованным объектам.

БУ УРД предоставляет следующие новые возможности для администраторов КМИС:

1. Автоматизация процессов построения, оптимизации, администрирования и анализа ролевой политики разграничения доступа.
2. Автоматизация процессов построения и анализа мандатной политики разграничения доступа.
3. Автоматизация процессов построения и администрирования дискреционной политики разграничения доступа.
4. Автоматизация процессов построения правил доступа, удовлетворяющих требованиям нескольких политик разграничения доступа.

БУ УРД состоит из набора консольных утилит и графической оболочки, позволяющей интерактивно вызывать данные утилиты. Графическая оболочка и утилиты реализованы для операционных систем семейства Windows, начиная с версии Windows XP.

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: Sergey V. Belim, Nadezda F. Bogachenko (eds.): Proceedings of the Workshop on Data, Modeling and Security 2018 (DMS-2018), Omsk, Russia, October 2018, published at <http://ceur-ws.org>

В БУ УРД входят следующие консольные утилиты:

1. Утилита **RBACLO** (LO – Local Optimization) предназначена для преобразования иерархии ролей и выбора её оптимального варианта.
2. Утилита **RBACRM** (RM – Role Mining) предназначена для определения множества ролей и построения иерархии ролей.
3. Утилита **RBACOA** (OA – Optimal Authorization) предназначена для выбора оптимальной роли при авторизации пользователя.
4. Утилита **RBACPSL** (PSL – Permissions Severity Level) предназначена для анализа иерархии ролей с целью выявления полномочий, для которых возможная избыточность наиболее нежелательна.
5. Утилита **MACLM** (LM – Lattice Mining) предназначена для определения свойств решётки ценностей мандатной политики разграничения доступа.
6. Утилита **DACCBI** (CBI – Case-Based Interpolation) предназначена для определения доступов субъектов к объектам дискреционной политики разграничения доступа по явно заданным правилам доступа.
7. Утилита **RBACMAC** предназначена для совмещения правил ролевой и мандатной политики разграничения доступа.
8. Утилита **MACDAC** предназначена для совмещения правил мандатной и дискреционной политики разграничения доступа.
9. Утилита **НАС** (Hierarchy Access Control) предназначена для разграничения доступа к иерархически организованным объектам с возможностью только последовательного доступа по иерархии.

Графическая оболочка предназначена для вызова утилит в интерактивной форме. Графическая оболочка содержит кнопки, позволяющие вызывать каждую из утилит в отдельности. Если утилита требует ввода данных из командной строки, то графическая оболочка запрашивает необходимые данные у пользователя в диалоговых окнах. После этого формируется необходимая строка вызова и запускается утилита. Если утилита требует в качестве входного параметра имя файла, то открывается окно для выбора файла. Если утилита имеет собственную графическую оболочку, то данная утилита вызывается из оболочки, при этом сама оболочка остаётся в активном состоянии.

2 XML-представление основных структур данных

2.1 Представление ролевого графа

Все утилиты, затрагивающие ролевое разграничение доступа, используют представление ролевого графа в формате GraphML [2], который является языком описания графов на основе XML.

Граф задаётся элементом `<graph>`, вершины графа – элементами `<node>`, дуги графа – элементами `<edge>`. Роль, соответствующая данной вершине, определяется элементом `<data key="r">`. Метка вершины определяется элементом `<data key="p">`. Для ролевого графа метка – это вектор (строка), в котором координаты соответствуют полномочиям и принимают значения 0 или 1; единичные значения координат характеризуют полномочия, сопоставленные данной вершине-роли (см. пример 1).

Описание графа дополняется описанием элемента `<permissionsList>` – списка полномочий. Элемент `<permission>` является потомком элемента `<permissionsList>`, в качестве атрибута он содержит порядковый номер полномочия в списке. Потомки элемента `<permission>`: элемент `<number>` – номер полномочия; элемент `<name>` – имя полномочия. При описании ролевого графа элемент `<number>` соответствует индексу полномочия в векторе-метке, приписываемом вершине графа.

Пример 1. Далее приведён фрагмент GraphML-документа. Основные характеристики представленного ролевого графа: число вершин-ролей равно 3; число дуг, отражающих авторизацию ролей друг на друга, равно 2; общее число полномочий (длина вектора, определяющего метку вершины), равно 3.

```
<key id="r" for="node" attr.name="role" attr.type="string">
</key>
<key id="p" for="node" attr.name="permissions" attr.type="string"><default>000</default>
</key>
<graph id="G" edgedefault="directed">
  <node id="3"><data key="r">R3</data><data key="p">100</data></node>
  <node id="2"><data key="r">R2</data><data key="p">011</data></node>
  <node id="1"><data key="r">R1</data><data key="p">111</data></node>
```

```

    <edge source="1" target="2"/>
    <edge source="1" target="3"/>
</graph>
<permissionsList>
  <permission id="1">
    <number> 0 </number>
    <name> P1 </name>
  </permission>
  <permission id="2">
    <number> 1 </number>
    <name> P2 </name>
  </permission>
  <permission id="3">
    <number> 2 </number>
    <name> P3 </name>
  </permission>
</permissionsList>

```

В приведённом примере роль «R3» обладает полномочием «P1», роль «R2» – полномочиями «P2» и «P3», роль «R1» – полномочиями «P1», «P2» и «P3».

2.2 Представление матрицы

Для XML-представления матриц (и, как частного случая, векторов) используется подход, принятый в библиотеке OpenCV [3]: указывается число строк, число столбцов, тип элементов, а затем построчно перечисляются элементы матрицы, используя в качестве разделителей пробелы или иные символы табуляции.

Элемент `<matrix>` содержит описание матрицы: элемент `<rows>` определяет количество строк; элемент `<cols>` – количество столбцов; элемент `<dt>` – тип элементов матрицы; в элементе `<data>` перечислены элементы матрицы; элементы `<rowsNames>` и `<colsNames>` определяют имена строк и столбцов матрицы (их элементы-потомки `<row>` и `<col>` в качестве атрибутов содержат порядковые номера строк и столбцов, соответственно). Представление данных с использованием элемента `<matrix>` назовём форматом MatrixML (см. пример 2).

Пример 2. Далее в формате MatrixML представлена матрица

$$\begin{matrix} & A & B & C \\ \text{I} & \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \\ \text{II} & \end{matrix}$$

```

<matrix id="M">
  <rows> 2 </rows>
  <cols> 3 </cols>
  <dt> i </dt>
  <data> 1 2 3 4 5 6 </data>
  <rowsNames>
    <row id="1"> I </row>
    <row id="2"> II </row>
  </rowsNames>
  <colsNames>
    <col id="1"> A </col>
    <col id="2"> B </col>
    <col id="3"> C </col>
  </colsNames>
</matrix>

```

3 Утилита RBACLO

3.1 Общее описание

Утилита RBACLO предназначена для локальных эквивалентных преобразований иерархии ролей (ролевого графа) в рамках оптимизации ролевой политики разграничения доступа. Утилита RBACLO в качестве исходных данных принимает существующую иерархию ролей. Результатом работы утилиты RBACLO является один или несколько ролевых графов, эквивалентных исходной иерархии ролей с точки зрения ролевого разграничения доступа и локально оптимальных в соответствии с выбранным критерием.

3.2 Формат входных данных утилиты RBACLO

Исходный ролевой граф в формате GraphML (см. пример 1).

3.3 Формат выходных данных утилиты RBACLO

Результирующий ролевой граф в формате GraphML (см. пример 1).

3.4 Алгоритмы, реализованные в утилите RBACLO

В утилите RBACLO реализованы следующие алгоритмы [4]:

1. Преобразование ролевого графа в единичный листовой ролевой граф.
2. Преобразование ролевого графа в листовой ролевой граф.
3. Преобразование ролевого графа в *RP*-сокращенный ролевой граф.
4. Преобразование ролевого графа в древовидный ролевой граф.
5. Преобразование ролевого графа в транзитивно-сокращенный ролевой граф.

3.5 Формат вызова утилиты RBACLO

Вызов утилиты RBACLO осуществляется в следующем формате.

```
rbaclo.exe [ключ] [входной файл] [выходной файл]
```

Параметр [входной файл] определяет имя файла, в котором хранится исходный ролевой граф. Параметр [выходной файл] определяет имя файла, в который будет записан результирующий ролевой граф. Ключ вызова определяет алгоритм, используемый для преобразования ролевого графа. Возможные значения ключа:

- a – вызов алгоритма преобразования ролевого графа в единичный листовой ролевой граф;
- b – вызов алгоритма преобразования ролевого графа в листовой ролевой граф;
- c – вызов алгоритма преобразования ролевого графа в *RP*-сокращенный ролевой граф;
- d – вызов алгоритма преобразования ролевого графа в древовидный ролевой граф;
- e – вызов алгоритма преобразования ролевого графа в транзитивно-сокращенный ролевой граф.

Пример 3. Данный вызов преобразует исходный ролевой граф из файла graph1.graphml в *RP*-сокращенный ролевой граф и записывает результат в файл graph2.graphml.

```
rbaclo.exe -c graph1.graphml graph2.graphml
```

4 Утилита RBACRM

4.1 Общее описание

Утилита RBACRM предназначена для выявления множества ролей и получения иерархии ролей в рамках построения ролевой политики разграничения доступа. Утилита RBACRM в качестве исходных данных принимает дискретное отображение, задающее распределение полномочий между пользователями. Результатом работы утилиты RBACRM является ролевой граф, порождающий верхнюю полурешётку на множестве ролей, и дискретное отображение, задающее авторизацию пользователей на роли.

4.2 Формат входных данных утилиты RBACRM

Матрица распределения полномочий между пользователями – матрица, в которой строки соответствуют полномочиям; столбцы – пользователям; в ячейках записаны значения 0 или 1; единичные значения характеризуют наличие полномочия у пользователя. Матрица должна быть представлена в формате MatrixML (см. пример 4).

Пример 4. Пусть пользователь «U1» обладает полномочиями «P1», «P2» и «P3». Пользователь «U2» – полномочием «P1». Далее представлена матрица распределения полномочий между пользователями в формате MatrixML.

```
<matrix id="matrixPU">
  <rows> 3 </rows>
  <cols> 2 </cols>
  <dt> i </dt>
  <data>
    1 1
    1 0
    1 0
  </data>
  <rowsNames>
    <row id="1"> P1 </row>
    <row id="2"> P2 </row>
    <row id="3"> P3 </row>
  </rowsNames>
  <colsNames>
    <col id="1"> U1 </col>
    <col id="2"> U2 </col>
  </colsNames>
</matrix>
```

4.3 Формат выходных данных утилиты RBACRM

1. Ролевой граф в формате GraphML (см. пример 1).

2. Матрица авторизации пользователей на роли – матрица, в которой строки соответствуют пользователям; столбцы – ролям; в ячейках записаны значения 0 или 1; единичные значения задают авторизацию пользователя на роль. Матрица должна быть представлена в формате MatrixML (см. пример 5).

Пример 5. Пусть пользователь «U1» может быть авторизован на роли «R1» и «R2», а пользователи «U2» и «U3» – только на роль «R2». Далее представлена матрица авторизации пользователей на роли в формате MatrixML.

```
<matrix id="matrixUR">
  <rows> 3 </rows>
  <cols> 2 </cols>
  <dt> i </dt>
  <data>
    1 1
    0 1
    0 1
  </data>
  <rowsNames>
    <row id="1"> U1 </row>
    <row id="2"> U2 </row>
    <row id="3"> U3 </row>
  </rowsNames>
  <colsNames>
```

```

        <col id="1"> R1 </col>
        <col id="2"> R2 </col>
    </colsNames>
</matrix>

```

4.4 Алгоритм, реализованный в утилите RBACRM

В утилите RBACRM реализован алгоритм решения задачи разработки ролей, основанный на анализе формальных понятий [5].

4.5 Формат вызова утилиты RBACRM

Вызов утилиты RBACRM осуществляется в следующем формате.

```
rbacrm.exe [входной файл] [выходной файл 1] [выходной файл 2]
```

Параметр [входной файл] определяет имя файла, в котором хранится матрица распределения полномочий между пользователями. Параметр [выходной файл 1] определяет имя файла, в который будет записан ролевой граф. Параметр [выходной файл 2] определяет имя файла, в который будет записана матрица авторизации пользователей на роли.

Пример 6. Данный вызов по матрице распределения полномочий между пользователями из файла pu.xml строит ролевой граф, который записывает в файл graph.graphml, и определяет матрицу авторизации пользователей на роли, которую сохраняет в файл ur.xml.

```
rbacrm.exe pu.xml graph.graphml ur.xml
```

4.6 Вспомогательная утилита RBACPU

При применении утилиты RBACRM для оптимизации ролевого разграничения доступа может потребоваться автоматическое построение отображения, задающего распределение полномочий между пользователями, по существующей иерархии ролей и заданному отображению авторизации пользователей на роли. Для этих целей используется вспомогательная утилита RBACPU, вызов которой осуществляется в следующем формате.

```
rbacpu.exe [входной файл 1] [входной файл 2] [выходной файл]
```

Параметр [входной файл 1] определяет имя файла, в котором хранится ролевой граф в формате GraphML. Параметр [входной файл 2] определяет имя файла, в котором хранится матрица авторизации пользователей на роли в формате MatrixML (см. раздел «Формат выходных данных утилиты RBACRM»). Параметр [выходной файл] определяет имя файла, в который будет записана матрица распределения полномочий между пользователями в формате MatrixML (см. раздел «Формат входных данных утилиты RBACRM»).

Пример 7. Данный вызов по ролевому графу из файла graph.graphml и матрице авторизации пользователей на роли из файла ur.xml строит матрицу распределения полномочий между пользователями и записывает в файл pu.xml.

```
rbacpu.exe graph.graphml ur.xml pu.xml
```

5 Утилита RBASOA

5.1 Общее описание

Утилита RBASOA предназначена для выбора оптимальной роли для авторизации пользователя в рамках построения или администрирования ролевой политики разграничения доступа. Утилита RBASOA в качестве исходных данных принимает иерархию ролей и набор полномочий, необходимый некоторому пользователю для работы в системе. Результатом работы утилиты RBASOA является набор ролей, доступных для авторизации заданного пользователя и предоставляющих пользователю требуемые полномочия, линейно упорядоченный по убыванию (невозрастанию) уровня предпочтительности роли для авторизации.

5.2 Формат входных данных утилиты RBASOA

1. Ролевой граф в формате GraphML (см. пример 1).

2. Вектор полномочий – вектор, в котором координаты соответствуют полномочиям и принимают значения 0 или 1; единичные значения координат характеризуют полномочия, требующиеся пользователю. Вектор должен быть представлен в формате MatrixML (см. пример 8). Следует заметить, что вектор полномочий может формироваться из матрицы, описанной в разделе «Формат входных данных утилиты RBACRM», как транспонированный столбец, соответствующий указанному пользователю.

Пример 8. Пусть из трёх полномочий «P1», «P2» и «P3» пользователю «U1» для работы в системе требуются полномочия «P1» и «P2». Далее представлен вектор полномочий этого пользователя в формате MatrixML.

```
<matrix id="vectorP">
  <rows> 1 </rows>
  <cols> 3 </cols>
  <dt> i </dt>
  <data>
    1 1 0
  </data>
  <rowsNames><row id="1"> U1 </row></rowsNames>
  <colsNames>
    <col id="1"> P1 </col>
    <col id="2"> P2 </col>
    <col id="3"> P3 </col>
  </colsNames>
</matrix>
```

5.3 Формат выходных данных утилиты RBASOA

Список ролей доступных для авторизации – список, в котором каждый элемент содержит информацию о роли и её весовом коэффициенте характеризующем предпочтительность роли для авторизации заданного пользователя. Элементы списка упорядочены по невозрастанию значений весовых коэффициентов. Значения весовых коэффициентов принадлежат отрезку [0, 1]; сумма всех весовых коэффициентов равна 1. Список должен быть представлен в формате XML с использованием элемента <rolesList> (см. пример 9). Элемент <role> является потомком элемента <rolesList>, в качестве атрибута он содержит порядковый номер роли в списке. Потомки элемента <role>: элемент <name> – имя роли; элемент <weight> – весовой коэффициент.

Пример 9. Пусть для авторизации доступны три роли «R2», «R4» и «R7», их весовые коэффициенты равны 0.3, 0.5 и 0.2, соответственно. Далее представлен список ролей доступных для авторизации в формате XML с использованием элемента <rolesList>.

```
<rolesList>
  <role id="1">
    <name> R4 <\name>
    <weight> 0.5 <\weight>
  </role>
  <role id="2">
    <name> R2 <\name>
    <weight> 0.3 <\weight>
  </role>
  <role id="3">
    <name> R7 <\name>
    <weight> 0.2 <\weight>
  </role>
</rolesList>
```

5.4 Алгоритм, реализованный в утилите RBASOA

В утилите RBASOA реализован алгоритм решения задачи авторизации пользователя в системе с ролевым разграничением доступа, основанный на методе анализа иерархий [6].

5.5 Формат вызова утилиты RBASOA

Вызов утилиты RBASOA осуществляется в следующем формате.

```
rbasoa.exe [входной файл 1] [входной файл 2] [выходной файл]
```

Параметр [входной файл 1] определяет имя файла, в котором хранится ролевой граф. Параметр [входной файл 2] определяет имя файла, в котором хранится вектор полномочий. Параметр [выходной файл] определяет имя файла, в который будет записан список ролей доступных для авторизации.

Пример 10. Данный вызов по ролевому графу из файла graph.graphml и вектору полномочий из файла ur.xml формирует список ролей доступных для авторизации и записывает в файл ur.xml.

```
rbasoa.exe graph.graphml ur.xml ur.xml
```

6 Утилита RBACPSL

6.1 Общее описание

Утилита RBACPSL предназначена для ранжирования полномочий по уровню критичности в рамках анализа ролевой политики разграничения доступа. Утилита RBACPSL в качестве исходных данных принимает иерархию ролей. Результатом работы утилиты RBACPSL является линейно упорядоченное множество полномочий. Упорядочивание происходит по убыванию (невозрастанию) уровня критичности.

6.2 Формат входных данных утилиты RBACPSL

Ролевой граф в формате GraphML (см. пример 1).

6.3 Формат выходных данных утилиты RBACPSL

Список полномочий – список, в котором каждый элемент содержит информацию о полномочии и его весовом коэффициенте, характеризующем уровень критичности полномочия. Элементы списка упорядочены по невозрастанию значений весовых коэффициентов. Значения весовых коэффициентов принадлежат отрезку $[0, 1]$; сумма всех весовых коэффициентов равна 1. Список должен быть представлен в формате XML с использованием элемента `<permissionsList>` (см. раздел «Представление ролевого графа»). Добавлен элемент `<weight>` – весовой коэффициент (см. пример 11).

Пример 11. Пусть весовые коэффициенты полномочий «P1», «P2», «P3» равны 0.2, 0.35 и 0.45, соответственно. Далее представлен список полномочий в формате XML с использованием элемента `<permissionsList>`.

```
<permissionsList>
  <permission id="1">
    <name> P3 <\name>
    <weight> 0.45 <\weight>
  </permission>
  <permission id="2">
    <name> P2 <\name>
    <weight> 0.35 <\weight>
  </permission>
  <permission id="3">
    <name> P1 <\name>
    <weight> 0.2 <\weight>
  </permission>
</permissionsList>
```

6.4 Алгоритм, реализованный в утилите RBACPSL

В утилите RBACPSL реализован алгоритм расчёта уровня критичности полномочий в системе с ролевым разграничением доступа, основанный на методе анализа иерархий [7].

Для работы алгоритма требуется древовидная единичная листовая иерархия ролей. Для выполнения этих требований использована утилита `rbaclo.exe` с ключом `-d`, затем с ключом `-a`.

6.5 Формат вызова утилиты RBACPSL

Вызов утилиты RBACPSL осуществляется в следующем формате.

```
rbacpsl.exe [входной файл] [выходной файл]
```

Параметр [входной файл] определяет имя файла, в котором хранится ролевой граф. Параметр [выходной файл] определяет имя файла, в который будет записан список полномочий.

Пример 12. Данный вызов по ролевому графу из файла `graph.graphml` формирует список полномочий и записывает в файл `p.xml`.

```
rbacpsl.exe graph.graphml p.xml
```

7 Утилита MACLM

7.1 Общее описание

Утилита MACLM предназначена для выявления характеристик решётки ценностей по известному ориентированному графу в рамках построения и анализа мандатной политики разграничения доступа. Утилита MACLM в качестве исходных данных принимает ориентированный граф, задающий разрешённые в системе информационные потоки. Результатом работы утилиты MACLM является ответ на вопрос, соответствует ли оргграф решётке ценностей. И если «да», то утилита осуществляет попытку определения типа решётки: линейная $LS(n)$, решётка подмножеств $XS(n)$ или мультиуровневая решетка $MLS(n, m) = XS(n) \times LS(m)$, и её количественных параметров: n , m .

7.2 Формат входных данных утилиты MACLM

Матрица смежности ориентированного графа в формате MatrixML (см. пример 13).

Пример 13. Далее представлена матрица смежности ориентированного графа с множеством вершин $\{v1, v2, v3\}$ и множеством дуг $\{(v1, v2), (v1, v3)\}$ в формате MatrixML.

```
<matrix>
  <rows> 3 </rows>
  <cols> 3 </cols>
  <dt> i </dt>
  <data>
    1 1 1
    0 1 0
    0 0 1
  </data>
  <rowsNames>
    <row id="1"> v1 </row>
    <row id="2"> v2 </row>
    <row id="3"> v3 </row>
  </rowsNames>
  <colsNames>
    <col id="1"> v1 </col>
    <col id="2"> v2 </col>
    <col id="2"> v3 </col>
  </colsNames>
</matrix>
```

7.3 Формат выходных данных утилиты MACLM

В текстовый файл выводится одно из пяти сообщений:

- 1) «no» – оргграф не соответствует алгебраической решётке;
- 2) «yes» – оргграф соответствует алгебраической решётке, тип решётки не определён;
- 3) «yes, LS(n), n = ...» – оргграф соответствует линейной решётке;
- 4) «yes, XS(n), n = ...» – оргграф соответствует решётке подмножеств;
- 5) «yes, MLS(n, m), n = ..., m = ...» – оргграф соответствует *MLS*-решётке.

7.3.1 Алгоритм, реализованный в утилите MACLM

В утилите MACLM реализован алгоритм восстановления характеристик решётки ценностей по известному ориентированному графу, задающему разрешённые в системе информационные потоки [8].

7.4 Формат вызова утилиты MACLM

Вызов утилиты MACLM осуществляется в следующем формате.

maclm.exe [входной файл] [выходной файл]

Параметр [входной файл] определяет имя файла, в котором хранится матрица смежности ориентированного графа. Параметр [выходной файл] определяет имя файла, в который будет записано текстовое сообщение о результатах поиска соответствия ориентированного графа решётке ценностей.

Пример 14. Данный вызов по матрице смежности ориентированного графа из файла graph.xml осуществляет попытку восстановления решётки ценностей и записывает результирующее текстовое сообщение в файл text.txt.

maclm.exe graph.xml text.txt

8 Утилита DACCBI

8.1 Общее описание

Утилита DACCBI предназначена для автоматического определения доступов субъектов к объектам по имеющимся прецедентам (явно заданным правилам доступа субъектов к объектам) в рамках построения и администрирования дискреционной политики разграничения доступа. Утилита DACCBI в качестве исходных данных принимает матрицу атрибутов субъектов, матрицу атрибутов объектов, матрицу доступов, последовательность координат прецедентов и новый прецедент. Результатом работы утилиты DACCBI является новая матрица доступов и пополненная последовательность координат прецедентов.

8.2 Формат входных данных утилиты DACCBI

1. Матрица атрибутов субъектов – матрица, строки которой соответствуют субъектам, столбцы – атрибутам безопасности субъекта, в ячейке $[S_i, A_j]$ содержится значение атрибута A_j субъекта S_i . Столбцы упорядочены по убыванию в соответствии с введённым линейным порядком на множестве атрибутов безопасности субъекта. Матрица должна быть представлена в формате MatrixML (см. пример 15).

Пример 15. Пусть в системе определено два атрибута безопасности субъекта $A1$ и $A2$. На множестве атрибутов введён линейный порядок: $A1 > A2$. Значение атрибута $A1$ выбирается из множества $\{a11, a12, a13\}$, значение атрибута $A2$ – из множества $\{a21, a22\}$. И пусть в системе действует три субъекта $S1, S2, S3$. Значения атрибутов безопасности субъектов заданы следующим образом.

$$S1 : A1 = a12, A2 = a22; \quad S2 : A1 = a13, A2 = a21; \quad S3 : A1 = a11, A2 = a21.$$

Далее представлена матрица атрибутов субъектов в формате MatrixML.

```
<matrix id="matrixAttrS">
  <rows> 3 </rows>
  <cols> 2 </cols>
  <dt> s </dt>
```

```

<data>
  a12 a22
  a13 a21
  a11 a21
</data>
<rowsNames>
  <row id="1"> S1 </row>
  <row id="2"> S2 </row>
  <row id="3"> S3 </row>
</rowsNames>
<colsNames>
  <col id="1"> A1 </col>
  <col id="2"> A2 </col>
</colsNames>
</matrix>

```

2. Матрица атрибутов объектов – матрица, строки которой соответствуют объектам, столбцы – атрибутам безопасности объекта, в ячейке $[O_i, B_j]$ содержится значение атрибута B_j объекта O_i . Столбцы упорядочены по убыванию в соответствии с введённым линейным порядком на множестве атрибутов безопасности объекта. Матрица должна быть представлена в формате MatrixML аналогично матрице атрибутов субъектов.

3. Исходная матрица доступов, содержащая список координат прецедентов. Матрица доступов определяется стандартным образом: строки соответствуют субъектам, столбцы – объектам, в ячейке $[S_i, O_j]$ содержится маска доступа субъекта S_i к объекту O_j . Маска доступа – целое неотрицательное число

$$m_{ij} = p_1 \cdot 2^0 + p_2 \cdot 2^1 + \dots + p_n \cdot 2^{n-1}, \quad (1)$$

где $p_k = 1$, если субъект S_i обладает k -м правом доступа к объекту O_j , и $p_k = 0$ – в противном случае; $k = 1, \dots, n$; n – общее число прав доступа. Матрица должна быть представлена в формате MatrixML (см. пример 16). Описание матрицы дополняется описанием элемента `<permissionsList>` – списка прав доступа (см. раздел «Представление ролевого графа»). Описание матрицы дополняется описанием элемента `<casesList>` – списка координат прецедентов (координат ячеек матрицы доступов, для которых доступ определён явно). Элемент `<case>` является потомком элемента `<casesList>`, в качестве атрибутов он содержит три идентификатора: `id` – порядковый номер прецедента в списке; `r` – идентификатор строки; `c` – идентификатор столбца.

Пример 16. Пусть в системе определено два субъекта $S1$ и $S2$, два объекта $O1$ и $O2$, три права доступа «P1», «P2», «P3». Субъект $S1$ не имеет прав на объект $O1$ и обладает правом «P2» на объект $O2$. Субъект $S2$ обладает правами «P1» и «P3» на объект $O1$ и не имеет прав на объект $O2$. При этом администратором безопасности явно заданы доступы субъекта $S1$ к объекту $O2$ и субъекта $S2$ к объекту $O1$. Далее представлена матрица доступов в формате MatrixML, содержащая список координат прецедентов.

```

<matrix id="matrixAccess">
  <rows> 2 </rows>
  <cols> 2 </cols>
  <dt> i </dt>
  <data>
    0 2
    5 0
  </data>
  <rowsNames>
    <row id="1"> S1 </row>
    <row id="2"> S2 </row>
  </rowsNames>
  <colsNames>
    <col id="1"> O1 </col>
    <col id="2"> O2 </col>
  </colsNames>
</matrix>

```

```

</colsNames>
<permissionsList>
  <permission id="1">
    <number> 0 </number>
    <name> P1 </name>
  </permission>
  <permission id="2">
    <number> 1 </number>
    <name> P2 </name>
  </permission>
  <permission id="3">
    <number> 2 </number>
    <name> P3 </name>
  </permission>
</permissionsList>
<casesList>
  <case id="1" r="1" c="2"/>
  <case id="2" r="2" c="1"/>
</casesList>
</matrix>

```

4. Новый прецедент (новый доступ) – тройка «субъект – объект – маска доступа». Прецедент должен быть определён в формате XML с использованием элемента `<access>` (см. пример 17). Потомки этого элемента: `<subject>` – субъект; `<object>` – объект, `<mask>` – маска доступа (см. формулу 1).

Пример 17. Пусть администратором безопасности явно определяется доступ субъекта *S1* к объекту *O1* в виде набора прав {«P2», «P3»}. Далее представлено XML-описание этого прецедента.

```

<access>
  <subject> S1 </subject>
  <object> O1 </object>
  <mask> 6 </mask>
</access>

```

8.3 Формат выходных данных утилиты DACSVI

Результирующая матрица доступов, содержащая список координат прецедентов (см. пример 16).

8.4 Алгоритмы, реализованные в утилите DACSVI

В утилите DACSVI реализованы алгоритмы формирования правил доступа субъектов к объектам по имеющимся прецедентам [9, 10]:

1. Частичная интерполяция матрицы доступов.
2. Последовательная интерполяция матрицы доступов.

8.5 Формат вызова утилиты DACSVI

Вызов утилиты DACSVI осуществляется в следующем формате.

```

dacsbi.exe [ключ] [входной файл 1] [входной файл 2] [входной файл 3]
[входной файл 4] [выходной файл]

```

Параметр [входной файл 1] определяет имя файла, в котором хранится матрица атрибутов субъектов. Параметр [входной файл 2] определяет имя файла, в котором хранится матрица атрибутов объектов. Параметр [входной файл 3] определяет имя файла, в котором хранится исходная матрица доступов, содержащая список координат прецедентов. Параметр [входной файл 4] определяет имя файла, в котором хранится новый прецедент. Параметр [выходной файл] определяет имя файла, в который будет записана

результатирующая матрица доступов, содержащая список координат прецедентов. с использованием элемента `<casesList>`. Ключ вызова определяет алгоритм, используемый для формирования правил доступа. Возможные значения ключа:

- a – вызов алгоритма частичной интерполяции матрицы доступов.
- b – вызов алгоритма последовательной интерполяции матрицы доступов.

Пример 18. Данный вызов по матрице атрибутов субъектов из файла `atrs.xml`, матрице атрибутов объектов из файла `atro.xml`, и новому прецеденту из файла `access.xml` преобразует исходную матрицу доступов, содержащую список координат прецедентов, из файла `matr1.xml` согласно алгоритму частичной интерполяции и записывает результирующую матрицу доступов вместе с обновлённой последовательностью координат прецедентов в файл `matr2.xml`.

```
daccbi.exe -a atrs.xml atro.xml matr1.xml access.xml matr2.xml
```

9 Утилита RBASMAC

9.1 Общее описание

Утилита RBASMAC предназначена для построения правил доступа, удовлетворяющих требованиям двух действующих политик разграничения доступа в рамках совмещения ролевого и мандатного принципов разграничения доступа. Утилита RBASMAC в качестве исходных данных принимает существующую иерархию ролей и ориентированный граф, определяющий решётку ценностей мандатного разграничения доступа. Результатом работы утилиты RBASMAC является решёточная иерархия ролей (соответствующий ролевой граф порождает как отношение порядка на множестве ролей, так и решётку ценностей на множестве меток безопасности).

9.2 Формат входных данных утилиты RBASMAC

1. Ролевой граф в формате GraphML (см. пример 1).
2. Матрица смежности орграфа, задающего решётку ценностей, в формате MatrixML (см. пример 13).

9.3 Формат выходных данных утилиты RBASMAC

Результирующий ролевой граф в формате GraphML (см. пример 1).

9.4 Алгоритм, реализованный в утилите RBASMAC

В утилите RBASMAC реализован алгоритм совмещения ролей и полномочий ролевой политики разграничения доступа и меток безопасности мандатной политики разграничения доступа [11].

Для работы алгоритма требуется, чтобы исходный ролевой граф являлся решёточным. Для выполнения этих требований использована утилита `rbaslo.exe` с ключом `-d`. Возможно использование утилиты `rbasgm.exe`.

9.5 Формат вызова утилиты RBASMAC

Вызов утилиты RBASMAC осуществляется в следующем формате.

```
rbasmac.exe [входной файл 1] [входной файл 2] [выходной файл]
```

Параметр [входной файл 1] определяет имя файла, в котором хранится ролевой граф. Параметр [входной файл 2] определяет имя файла, в котором хранится матрица смежности орграфа, задающего решётку ценностей. Параметр [выходной файл] определяет имя файла, в который будет записан результирующий ролевой граф.

Пример 19. Данный вызов по ролевому графу из файла `graph1.graphml` и матрице смежности орграфа, задающего решётку ценностей, из файла `graph2.xml` формирует ролевой граф, совмещающий исходные роли, полномочия и метки безопасности, и записывает результат в файл `graph3.graphml`.

```
rbasmac.exe graph1.graphml graph2.xml graph3.graphml
```

10 Утилита MACDAC

10.1 Общее описание

Утилита MACDAC предназначена для построения правил доступа, удовлетворяющих требованиям двух действующих политик разграничения доступа в рамках совмещения мандатного и дискреционного принципов разграничения доступа. Утилита MACDAC в качестве исходных данных принимает коэффициент доминирования мандатной политики над дискреционной; максимальное значение уровня разрешения; ориентированный граф, определяющий решётку ценностей мандатного разграничения доступа; функцию безопасности мандатного разграничения доступа; матрицу доступов дискреционного разграничения доступа; запрос на доступ. Результатом работы утилиты MACDAC является уровень разрешения доступа.

10.2 Формат входных данных утилиты MACDAC

1. Коэффициент доминирования мандатной политики над дискреционной r – вещественное положительное число. Задаётся в строке вызова утилиты.
2. Максимальное значение уровня разрешения T – натуральное число. Задаётся в строке вызова утилиты.
3. Матрица смежности орграфа, задающего решётку ценностей, в формате MatrixML (см. пример 13).
4. Функция безопасности – списки меток безопасности субъектов и объектов. Оба списка должны быть представлены в формате XML с использованием элемента `<securityList>` (см. пример 20). Элемент `<elem>` является потомком элемента `<securityList>`. Потомки элемента `<elem>`: элемент `<name>` – имя субъекта или объекта; элемент `<label>` – метка безопасности (метки безопасности должны совпадать с именами строк и столбцов матрицы смежности орграфа, задающего решётку ценностей).

Пример 20. Далее представлена функция безопасности в формате XML с использованием элемента `<securityList>`.

```
<securityList id="subjects">
  <elem id="1">
    <name> S1 <\name>
    <label> v1 <\label>
  </elem>
  <elem id="2">
    <name> S2 <\name>
    <label> v3 <\label>
  </elem>
</securityList>
<securityList id="objects">
  <elem id="1">
    <name> 01 <\name>
    <label> v2 <\label>
  </elem>
  <elem id="2">
    <name> 02 <\name>
    <label> v3 <\label>
  </elem>
</securityList>
```

5. Матрица доступов в формате MatrixML (см. пример 16 (без элемента `<casesList>`)).
6. Запрос на доступ (тройка «субъект – объект – маска доступа») в формате XML с использованием элемента `<access>` (см. пример 17).

10.3 Формат выходных данных утилиты MACDAC

В текстовый файл выводится уровень разрешения – вещественное число в интервале $[-T, T]$.

10.4 Алгоритм, реализованный в утилите MACDAC

В утилите MACDAC реализован алгоритм совмещения ролевой и мандатной политик разграничения доступа [12].

10.5 Формат вызова утилиты MACDAC

Вызов утилиты MACDAC осуществляется в следующем формате.

```
macdac.exe [r] [T] [входной файл 1] [входной файл 2]  
[входной файл 3] [входной файл 4] [выходной файл]
```

Параметр [r] – вещественное положительное число – определяет коэффициент доминирования мандатной политики над дискреционной. Параметр [T] – натуральное число – определяет максимальное значение уровня разрешения. Параметр [входной файл 1] определяет имя файла, в котором хранится матрица смежности орграфа, задающего решётку ценностей. Параметр [входной файл 2] определяет имя файла, в котором хранится функция безопасности. Параметр [входной файл 3] определяет имя файла, в котором хранится матрица доступов. Параметр [входной файл 4] определяет имя файла, в котором хранится запрос на доступ. Параметр [выходной файл] определяет имя файла, в который будет записано текстовое сообщение, содержащее информацию об уровне разрешения.

Пример 21. Данный вызов по матрице смежности орграфа, задающего решётку ценностей, из файла graph.xml, функции безопасности из файла labels.xml, матрице доступов из файла matr.xml вычисляет уровень разрешения для доступа из файла access.xml при коэффициенте доминирования мандатной политики над дискреционной равном 2 и максимальном значении уровня разрешения равном 5. Результат вычислений записывается в файл level.txt.

```
macdac.exe 2 5 graph.xml labels.xml matr.xml access.xml level.txt
```

11 Утилита НАС

11.1 Общее описание

Утилита НАС предназначена для разграничения доступа к объектам, на множестве которых определено отношение порядка (задана иерархия объектов) и разрешён только последовательный доступ от доминирующего объекта к подчинённому. Утилита НАС в качестве исходных данных принимает орграф, задающий иерархию объектов, объект и его ключ доступа (если вместо объекта указано значение «0», то ключ является инициализирующим). Результатом работы утилиты НАС является набор ключей доступа к объектам, которые достижимы из заданного объекта (либо все объекты иерархии в случае знания инициализирующего ключа).

11.2 Формат входных данных утилиты НАС

1. Орграф иерархии объектов в формате GraphML. В отличие от ролевого графа (см. пример 1) атрибутами вершины такого орграфа являются имя объекта (элемент `<data key="o">`) и идентификатор объекта (элемент `<data key="i">`), а также отсутствует элемент `<permissionsList>` (см. пример 22).

Пример 22. Далее приведён фрагмент GraphML-документа, описывающий орграф иерархии объектов.

```
<key id="o" for="node" attr.name="object" attr.type="string"></key>  
<key id="i" for="node" attr.name="identifier" attr.type="string"></key>  
<graph id="G" edgedefault="directed">  
  <node id="3"><data key="o">03</data><data key="i">id3</data></node>  
  <node id="2"><data key="o">02</data><data key="i">id2</data></node>  
  <node id="1"><data key="o">01</data><data key="i">id1</data></node>  
  <edge source="1" target="2"/>  
  <edge source="1" target="3"/>  
</graph>
```

2. «Точка доступа» – пара «объект – ключ доступа» (либо пара «0 – ключ доступа»). Первый элемент пары – имя объекта – задаётся в строке вызова утилиты. Второй элемент пары – ключ доступа – задаётся в строке вызова утилиты.

11.3 Формат выходных данных утилиты НАС

Список ключей доступа – список пар «объект – ключ доступа», начиная с объекта из «точки доступа» в порядке, определяемом обходом в ширину подграфа иерархии объектов, порождённого всеми объектами, достижимыми из заданного. Список должен быть представлен в формате XML с использованием элемента `<keysList>` (см. пример 23). Элемент `<key>` является потомком элемента `<keysList>`. Потомки элемента `<key>`: элемент `<object>` – имя объекта; элемент `<value>` – значение ключа доступа к объекту.

Пример 23. Далее представлен список ключей доступа к объектам иерархии, начиная с объекта *O1*, в формате XML с использованием элемента `<keysList>`.

```
<keysList>
  <key id="1">
    <object> 01 <\object>
    <value> AbCdEfGh <\value>
  </key>
  <key id="2">
    <object> 02 <\object>
    <value> IjKlMnOp <\value>
  </key>
  <key id="3">
    <object> 03 <\object>
    <value> QrStUvWx <\value>
  </key>
</keysList>
```

11.4 Алгоритм, реализованный в утилите НАС

В утилите НАС реализован алгоритм распределения ключей доступа к иерархически организованным объектам [13].

В работе алгоритма использована утилита `gbaclo.exe` с ключом `-d`.

11.5 Формат вызова утилиты НАС

Вызов утилиты НАС осуществляется в следующем формате.

```
has.exe [алгоритм] [объект] [ключ доступа] [входной файл] [выходной файл]
```

Параметры [объект] и [ключ доступа] – два строковых значения – определяют «точку доступа». Параметр [входной файл] определяет имя файла, в котором хранится оргграф иерархии объектов. Параметр [выходной файл] определяет имя файла, в который будет записан зашифрованный список ключей доступа. Параметр [алгоритм] определяет алгоритм, используемый для шифрования данных в выходном файле. В качестве ключа алгоритма шифрования используется значение параметра [ключ доступа]. Возможные значения параметра [алгоритм]:

AES – используется алгоритм шифрования AES.

GOST – используется алгоритм шифрования ГОСТ.

Пример 24. Данный вызов по ключу доступа «AbCdEfGh» к объекту *O1* и оргграфу иерархии объектов из файла `graph.graphml` вычисляет ключи доступа к объектам, достижимым из объекта *O1*, и записывает зашифрованный алгоритмом AES список ключей доступа в файл `keys.xml`.

```
has.exe AES 01 AbCdEfGh graph.graphml keys.xml
```

Список литературы

- [1] N.F. Bogachenko. The Analysis of Problems of Access Control Administration in Large-Scale Information Systems. *Mathematical Structures and Modeling*, 2(46):135–152, 2018 (In Russian).
- [2] URL: <http://graphml.graphdrawing.org/primer/graphml-primer.html>.

- [3] URL: https://docs.opencv.org/2.4/modules/core/doc/old_xml_yaml_persistence.html.
- [4] N.F. Bogachenko. Lokal'naya optimizaciya politiki rolevogo razgranicheniya dostupa (Local Optimization of the Role-Based Access Control Policy). *CEUR Workshop Proceedings*, Vol. 1965, 2017. URL: <http://ceur-ws.org/Vol-1965/paper14.pdf>.
- [5] S.V. Belim, N.F. Bogachenko. The Use of a Concept Lattice for Constructing the Role-Based Access Control. *Information Science and Control Systems*, 1(55):16–28, 2018 (In Russian).
- [6] S.V. Belim, S.Yu. Belim, N.F. Bogachenko, A.N. Kabanov. User Authorization in a System with a Role-Based Access Control on the Basis of the Analytic Hierarchy Process. *IEEE Dynamics of Systems, Mechanisms and Machines (Dynamics)*, 14–16 Nov., 2017. URL: <http://ieeexplore.ieee.org/document/8239432/>.
- [7] S.V. Belim, N.F. Bogachenko. Using a Hierarchy Analysis Method to Assess Permission Leakage Risks in Systems with a Role Based Access Control. *Information and Control Systems*, 6:67–72, 2013 (In Russian).
- [8] S.V. Belim, N.F. Bogachenko. The Check of the Correspondence of the Directed Graph to the Algebraic Lattice. *Applied Discrete Mathematics*, 41:54–65, 2018 (In Russian).
- [9] S.V. Belim, N.F. Bogachenko, Yu.S. Rakitskiy. Use Case Approach for Creating Adaptive Discretionary Security Policy. *Information and Control Systems*, 2(48):4–16, 2016 (In Russian).
- [10] S.V. Belim, N.F. Bogachenko. Creation of a Discretionary Security Policy on Partial Set of Data. *Information Security Questions*, 2:23–28, 2016 (In Russian).
- [11] S.V. Belim, N.F. Bogachenko, Yu.S. Rakitskiy. Theoretical-Graph Approach to the Problem of Combining Role-Based and Mandatory Security Policies. *Information Security Problems. Computer Systems*, 2:9–17, 2010 (In Russian).
- [12] S.V. Belim, N.F. Bogachenko, A.N. Kabanov, Yu.S. Rakitskiy. Using the Decision Support Algorithms Combining Different Security Policies. *IEEE Dynamics of Systems, Mechanisms and Machines (Dynamics)*, 15–17 Nov., 2016. URL: <http://ieeexplore.ieee.org/document/7818976/>.
- [13] 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>.

The Automation of the Management Processes by the Access Control Policy

Nadezda F. Bogachenko

The library of the utilities which is intended for the automation of the management processes by the access control policy in the large-scale complex IT systems is described. Special attention is paid to formats of the input and output data. The data description on the basis of the XML is used. The GraphML format is applied too.