САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-Механический факультет
Кафедра системного программирования

Одеров Роман Сергеевич

Управление политиками контроля доступа на основе ролевой модели

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д.ф.-м.н., профессор Терехов А. Н.

Научный руководитель:
ст. преподаватель, Баклановский М.В.

Рецензент:
первый зам. Ген. директора, зам. Ген. конструктора
ОАО «НПО Машиностроения» по ИТ,
Мартынов В.И.

Санкт-Петербург
2014

SAINT-PETERSBURG STATE UNIVERSITY

Mathematics & Mechanics Faculty
Software Engineering Chair

Roman Oderov

Managing access control policies based on RBAC-model

Graduation Thesis

Admitted for defence
Head of the Chair:
Professor Andrey Terekhov

Scientific advisor:
Sr. Lect., Maxim Baklanovsky

Reviewer:
First Deputy Director General, Deputy Designer General of
JSC NPO Mashinostroyenia on IT,
Martynov Vyacheslav

Saint-Petersburg
2014

Оглавление

Введение

Безопасность – один из наиболее важных и сложных аспектов построения и функционирования современных информационных систем: от отдельно взятого приложения до сверхпроизводительного компьютерного кластера. Тема безопасности пронизывает весь жизненный цикл программного продукта: от зарождения идеи до вывода системы из эксплуатации. В IT-сфере под безопасностью (или информационной безопасностью) обычно понимают защищенность информации и ее окружения от целенаправленных или случайных действий, в результате которых может быть нанесен неприемлемый ущерб самой информационной системе или ее пользователям. [41]

Информационную безопасность разделяют на три ветви: обеспечение целостности, доступности и конфиденциальности данных и поддерживающей их инфраструктуры. Целостность можно определить, как достоверность и непротиворечивость информации. Типичный пример – корректное проведение транзакций в банке: переводимая сумма, списанная с одного счета, либо обязательно переходит на другой счет, либо вовсе не списывается с первого.

Доступность – обеспечение доступа к информации по мере необходимости. Примером нарушения доступности информации может послужить физическое повреждение каналов связи с источником запрашиваемых данных.

Конфиденциальность – защита от несанкционированного доступа к информации. Например, в результате взлома пароля электронной почты личные данные пользователя попали в руки злоумышленника.

По данным Ponemon Institute [14], средняя стоимость потерянных данных может составлять до трети стоимости компании, а средняя стоимость одной потерянной записи, например, в США – около $277. Таким образом, потеря важной информации может нанести организациям ощутимый ущерб, а чем больше потенциальный ущерб, тем серьезнее принимаемые защитные меры. [49]

Конфиденциальная информация делится на «классы» (уровни секретности) в зависимости от ее ценности и серьезности возможных последствий в случае утраты. Эти уровни определяются международными стандартами или внутренними законами и прочими нормативными документами конкретной страны. Например, в Российской Федерации информация может быть несекретной, конфиденциальной или иметь статус государственной тайны. Гос. тайна в свою очередь обладает грифами секретности: особой важности, совершенно секретно, секретно. В зависимости от класса и/или грифа секретности к данным применяются определенные меры, направленные на их сохранность и защиту от рассекречивания, например разграничение доступа [46], [45], [50], [43], [37], [44], [38].

Политики контроля доступа (политики разграничения доступа) – это набор правил, определяющих и контролирующих действия каждого пользователя в системе. Политики основаны на моделях контроля доступа – математически формализованных системах, включающих в себя субъекты (пользователи/ процессы), объекты (данные/ документы/ файлы) и правила доступа, по которым вычисляется возможность выполнения субъектом

набора операций над объектом. Среди моделей контроля доступа можно выделить две наиболее распространенных: MAC (Mandatory Access Control / Мандатный контроль доступа) и DAC (Discretionary access control / Дискреционный контроль доступа) [49], [4], [5], [20], [30], [41]. Система, работающая с данными повышенного уровня секретности, по закону обязана реализовывать одну из выше упомянутых моделей в зависимости от класса обрабатываемой информации [45].

Во многих организациях сотрудники вынуждены работать с документами *различных* уровней секретности. В этом кроется серьезная проблема, связанная с обеспечением комплексного контроля за сохранностью информации, в том числе путем реализации нескольких концептуально разных политик разграничения доступа. В случае одновременной работы с файлами различных уровней конфиденциальности нужно соблюдать повышенную осторожность: как «дыры» в самой системе, так и любое непродуманное действие пользователя могут привести к утечке (например, случайный переход по вредоносной ссылке в браузере). Сейчас в серьезных организациях обработка подобных данных осуществляется на физически отдельных машинах с повышенными требованиями к безопасности. Такой подход, очевидно, очень неудобен.

На кафедре системного программирования был дан старт проекту, призванному решить эту проблему. Система Multi-Cloud Desktop может обеспечить одновременную безопасную обработку документов различных уровней секретности в рамках одной клиентской машины с помощью современных технологий виртуализации и удаленной доставки приложений [39]. Ясно, что при реализации подобной системы в первую очередь встает вопрос о безопасности. С учетом функциональных особенностей в MCD необходимо построить комплексную систему безопасности, поддерживающую сразу несколько ключевых моделей разграничения доступа. Возникшая задача «сочетания» политик контроля доступа в рамках одной системы порождает и другие проблемы, среди которых: выбор «базиса», на котором будут основаны используемые модели; сложность управления такой комплексной системой безопасности; отслеживание и определение потенциально небезопасных состояний системы и т.д.

В данной дипломной работе проведено исследование различных моделей контроля доступа; показана возможность построения комплексной системы контроля доступа [СКД], в рамках которой могут быть реализованы различные модели и, соответственно, политики. Описано построение прототипов такой СКД на основе ролевой модели и инструмента ее администрирования.

1. Описание проекта MCD

Как уже упоминалось выше, проект MCD направлен на решение задачи одновременной безопасной работы с документами разных уровней секретности на одной физической машине. Опишем немного подробнее проблематику и идею, положенную в основу системы Multi-Cloud Desktop (Более детальное описание архитектуры и проекта в целом можно найти в [39][40]).

Предположим, пользователь работает с некоторой конфиденциальной информацией на локальной машине. Если он использует при этом ненадежное ПО, существует риск утечки этой информации. Не исключена и вероятность того, что пользователь из-за невнимательности рассекретит данные, перейдя, например, по вредоносной ссылке в Интернет-браузере. Существует множество подобных векторов атак.

В современных компаниях, серьезно заботящихся о сохранности своих данных, реализуется простой, но неудобный и затратный подход к решению этой проблемы: использование физически разделенных машин для работы с документами разных уровней секретности.

В системе MCD, напротив, предлагается работать с любыми конфиденциальными данными в рамках одного физического компьютера. Основная идея, позволяющая воплотить желаемый подход в жизнь, заключается в максимальной «взаимной» изоляции клиентской машины, секретной информации и приложений [40].

1.1. Архитектура MCD

Пользователь запускает приложения на локальном компьютере с предустановленным тонким клиентом. Программы выполняются в привычной манере, взаимодействие с ними также происходит с помощью обычных устройств ввода [40]. Однако пользователю предоставляется лишь «иллюзия» локальной работы: приложения запускаются в удаленном облаке на специальных серверах. Клиент в свою очередь получает видео-поток, отражающий состояние конкретных программ. Стоит заметить, что с помощью современных технологий виртуализации и удаленной доставки приложений можно добиться одновременной работы с приложениями для различных операционных систем на одном компьютере. Пользовательская машина изолирована от сети Интернет и «общается» только с упомянутыми выше доверенными серверами, на которых в виртуальных машинах запускаются приложения. Серверы и хранилища данных имеют несколько уровней доверия в зависимости от запускаемых программ и хранимой информации соответственно. Наиболее защищенным должен быть сервер для запуска

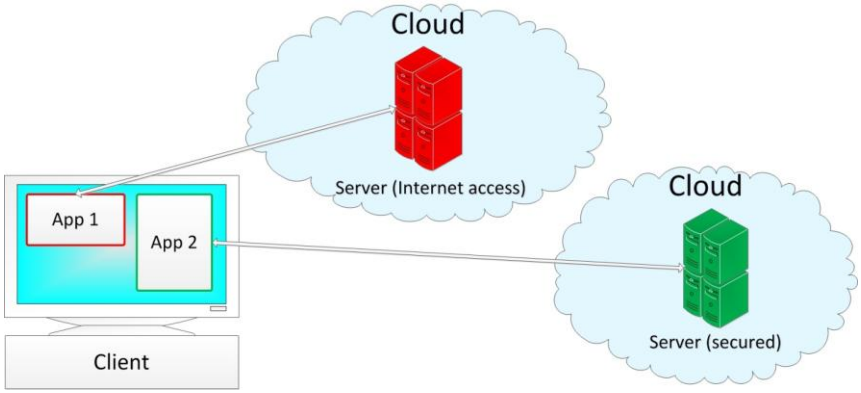


Рис.1. Общая архитектура MCD

приложений, обрабатывающих наиболее важные данные: желательно запретить ему доступ в Интернет и ограничить к нему физический доступ сотрудников организации [48].

1.2.Подсистема контроля доступа в MCD

Не будем заострять внимание на архитектурных особенностях (подробнее см. [39]), а сосредоточимся конкретно на обеспечении безопасности при работе с данными.

Все запросы от клиента к серверам должны проходить через специальный узел, который будет определять, можно ли начать работу с текущим пользователем и разрешить ли ему выполнение определенных операций в системе. Такие решения выносятся на основе различных политик контроля доступа в зависимости от степеней секретности данных в системе, как упоминалось ранее.

Итак, основной компонентой безопасности является подсистема контроля доступа, определяющая права пользователей при работе с информацией. В ней должен быть заложен универсальный единый «фундамент», на котором впоследствии будут построены требуемые модели и, соответственно, политики разграничения доступа. Реализация нескольких отдельных подсистем для каждой модели повлекла бы за собой резкое усложнение контроля безопасности в целом (и, тем самым, увеличение числа потенциальных уязвимостей), поэтому важно удачно подобрать единый «базис».

Конечно, при реализации универсальной модели резко возрастает сложность системы безопасности, но эта проблема решается с помощью разработки специальных инструментов управления политиками разграничения доступа. Такие средства должны включать в себя набор шаблонов для создания моделируемых политик, т.к. без них человеку придется оперировать терминами универсальной модели, что представляется затруднительным даже для профессионала. Также нужно предусмотреть возможность простого и понятного представления политик, например, в виде таблиц, диаграмм, схем.

Важно уделить должное внимание отслеживанию состояния системы безопасности и контролю действий, связанных с ее модификациями. При внесении каких-либо изменений в политики контроля доступа администратор может не заметить потенциально уязвимое состояние системы.

Таким образом, создание подсистемы безопасности в MCD обладает своими особенностями и трудностями, о которых пойдет речь ниже в данной работе.

2. Постановка задачи

Целью данной дипломной работы является доказательство возможности построения системы контроля доступа на базе универсальной модели, средствами которой могут быть выражены основные политики безопасности.

Для достижения сформулированной выше цели были поставлены следующие задачи.

- Обзор нормативных документов, касающихся требований к безопасности информационных систем, работающих с данными различных уровней секретности.
- Исследование существующих моделей контроля доступа, в том числе взаимосвязей между ними и возможности выражения одной через другую.
- Разработка прототипа инструмента управления политиками контроля доступа на базе универсальной модели.

3. Обзор нормативных документов

Ниже будет представлен обзор основных документов, имеющих отношение к теме данной работы, а именно к моделям контроля доступа в рамках информационных систем и к различным уровням секретности данных.

Руководящий документ «Концепция защиты средств вычислительной техники и автоматизированных систем от несанкционированного доступа к информации» [46]

В данном документе рассматриваются общие положения касаемо защиты информационных систем и данных. Вводятся определение несанкционированного доступа (НСД), основные принципы защиты от НСД. Описывается модель нарушителя в автоматизированной системе (АС), и выделяются 4 соответствующих уровня возможностей нарушителя. Все введенные понятия могут быть использованы при проектировании подсистемы безопасности автоматизированных информационных систем.

Закон о государственной тайне

Данный закон регулирует отношения, возникающие в связи с отнесением сведений к государственной тайне, их засекречиванием или рассекречиванием и защитой в интересах обеспечения безопасности Российской Федерации [37]. Для более полного изложения данной дипломной работы стоит выделить основные определения из этого закона.

Государственная тайна – защищаемые государством сведения в области его военной, внешнеполитической, экономической, разведывательной, контрразведывательной и оперативно-розыскной деятельности, распространение которых может нанести ущерб безопасности Российской Федерации.

Гриф секретности – реквизиты, свидетельствующие о степени секретности сведений, содержащихся в их носителе, проставляемые на самом носителе и (или) в сопроводительной документации на него.

Устанавливаются три степени секретности сведений, составляющих государственную тайну, и соответствующие этим степеням грифы секретности для носителей указанных сведений: "особой важности", "совершенно секретно" и "секретно".[1] В принципе, информация может быть отнесена не только к государственной тайне, но и быть конфиденциальной, а также попросту открытой. В этот класс попадают сведения, которые не угрожают безопасности целого государства, а могут повредить лишь интересам компании-владельца или ее клиентам (например, личная информация пользователей поставщика услуг мобильной связи).

[1] Важно заметить, что информация, не отнесенная к государственной тайне, не может быть помечена грифами секретности. Перечень сведений, отнесенных к государственной тайне РФ, постоянно обновляется и публикуется во многих источниках, к примеру, (Российская Газета, 2006)

Руководящий документ «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации». [45]

Документ устанавливает классификацию автоматизированных систем, подлежащих защите от несанкционированного доступа к информации, и требования по защите информации в АС различных классов. Этот документ может быть использован в качестве нормативного материала при разработке требований к защите. [45] В данном РД определяются 9 классов защищенности АС, отнесенных к трем группам в зависимости от характера обработки информации в системе. Требования безопасности ужесточаются от класса к классу, таким образом образуя иерархию. В каждой АС выделяется 4 основные подсистемы обеспечения защиты: управление доступом, регистрация и учет, криптография, обеспечение целостности. Три объемлющих группы АС различают следующим образом:

Группа	Кол-во пользователей	Доступная информация	Классы
1	Много. Разные права.	Не вся. Разные уровни конфиденциальности.	1А-…1Д
2	Много. Одинаковые права.	Вся. Разные уровни конфиденциальности.	2А, 2Б
3	Один	Вся. Один уровень конфиденциальности.	3А, 3Б

Табл.1. Группы АС

Стоит подчеркнуть, что проект MCD можно отнести к первой группе, из-за чего придется реализовать наиболее сложные механизмы защиты.

В приложении 1 можно увидеть выдержку из РД в виде таблицы, в которой описаны конкретные требования защиты АС в зависимости от класса. Заметим, что рассматриваемые далее модели контроля доступа также отражены в требованиях: в классе 1Г впервые упоминается необходимость реализации дискреционного (DAC), а в классах 1В и 2А – мандатного разделения доступа (MAC).

Руководящий документ «Защита от несанкционированного доступа к информации. Часть I. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей»

Руководящий документ классифицирует программное обеспечение средств защиты информации по уровню контроля отсутствия в нем недекларированных возможностей[44].

Существует 4 уровня контроля: первый (самый высокий) необходим при работе с грифом секретности «Особой важности», второй – с грифом «Совершенно секретно», третий соответствует «Секретно», четвертый относится к конфиденциальной информации. Таким образом, для ПО, которое защищает информацию, отнесенную к государственной тайне, должен соблюдаться уровень не ниже 3-го.

В рассматриваемом РД приводится сводная таблица (см. приложение 2) и описания конкретных требований к уровням контроля [44].

The Orange Book

«Оранжевая книга» («The orange book»; TCSEC, Trusted Computer System Evaluation Criteria) – один из ключевых стандартов в области компьютерной безопасности, использовавшийся правительством США с 1983 года; одна из книг известной «Радужной серии» стандартов. Определяет основные критерии оценки систем защиты, являющихся частью компьютерной системы [15]. Ясно, что абсолютно безопасных систем не бывает, и поэтому в TCSEC рассматривается понятие *доверенной системы*, т.е. системы, обладающей ресурсами, достаточными для обеспечения корректной одновременной работы с данными разных уровней секретности.[41] Также определяется *доверенная вычислительная база* – набор защитных механизмов системы, реализующих выбранную политику безопасности[2].

Степень доверия оценивается по двум показателям: *политике безопасности* (свод законов, норм, правил поведения, касающихся порядка обработки важных данных в системе/организации в целом) и *уровню гарантированности* (мера доверия архитектуре и реализации информационной системы).

В «Оранжевой книге» определяются различные классы безопасности (C1,C2,B1,B2,B3,A1), разделенные по группам – уровням доверия. Существует 4 уровня (в порядке увеличения степени доверия): D, C, B и A. Уровень D присваивается системам, не удовлетворяющим критериям рассматриваемого стандарта. При переходе от уровня C к уровню A к системам предъявляются все более жесткие требования, касающиеся различных аспектов безопасности. В том числе, конкретизируются и используемые модели контроля доступа. Так, уже на уровне C появляется требование реализации дискреционного, а на уровне B – мандатного разграничения доступа (о них пойдет разговор в следующих главах).

Common Criteria

Common Criteria (он же стандарт ISO/IEC 15408) – современный стандарт оценки безопасности информационных технологий (Common Criteria for Information Technology Security Evaluation), сменивший в 2005 году TCSEC [10][11][12]. Common Criteria, в отличие от «Оранжевой книги», не содержит предопределенных классов безопасности. В нем описываются «инструменты» и подходы к оценке защищенности систем. В терминах, предлагаемых CC, любая организация может построить требуемый класс безопасности. В CC рассматриваются два вида требований: функциональные требования и требования доверия. Функциональные требования соответствуют активному аспекту защиты (конкретные механизмы, функции, алгоритмы), а требования доверия – пассивному (предъявляются к технологии и процессу разработки) [10][11][12][41]. Common Criteria представлена тремя основными документами: «Part 1, Introduction and general model»/ «Введение и общая модель», «Part 2, Security functional components» / «Функциональные требования безопасности», «Part 3, Security assurance components» / «Требования доверия». Основными понятиями в Common Criteria являются: *TOE* (Target of Evaluation), *PP*

[2] Компоненты вне доверенной базы не обязаны быть доверенными, однако считается, что их уязвимости не критичны для системы безопасности в целом.

(Protection Profile), *ST* (Security Target), *SFR/SAR* (Security Functional/Assurance Requirements), *EAL* (Evaluation Assurance Level).

TOE – объект оценки – продукт, подлежащий оценке (например, операционная система, антивирус, чип в смарт-карте, локальная сеть со всем соответствующим оборудованием и программным обеспечением…).

PP – профиль защиты – описание требований к типу TOE (например, антивирус, файервол, операционная система), выбранных из наборов SFR/SAR. Может быть выбрано несколько профилей, которым удовлетворяет рассматриваемый TOE.

ST – задание по безопасности – документ, в котором описываются требования к конкретному TOE. Может содержать в себе один или несколько PP, а также описание TOE.

SFR/SAR – конкретные требования (функциональные/доверия), выбранные для оценки продукта. Существуют наборы предопределенных критериев, разделенные по классам в зависимости от их назначения. Например, класс FDP (User Data Protection) содержит требования, направленные на защиту пользовательских данных. В том числе, в терминах элементов FDP можно описать уже упоминавшиеся мандатную и дискреционную модели контроля доступа.

EAL – «уровень оценки» – численное представление «глубины» / «тщательности» оценки продукта в зависимости от выбранных критериев. Существует 7 EAL: от самого «поверхностного» EAL1 до самого серьезного EAL7. Уровень выше EAL4 встречается редко, так оценка становится все сложнее и дороже. К примеру, на данный момент в классе операционных систем только IBM (PR/SM гипервизор + RACF) и BAE Systems (XTS-400/STOP OS 6.4 U4) сертифицированы на уровень EAL5. [9]

Итак, организация, заинтересованная в сертификации продукта, определяет ряд понятий (в том числе, упомянутые выше), составляет спецификации с описанием требований к безопасности (ST) и таким образом осуществляет контроль безопасности создаваемой системы, в привязке к ее жизненному циклу. [41]

На этом закончим знакомство со стандартом Common Criteria. Более подробную информацию можно найти в [10][11][12][41][47].

4. Модели контроля доступа

Ниже приведены наиболее распространенные модели контроля доступа, описаны назначение и основные детали каждой модели, рассмотрены их особенности, даны примеры реализаций. Данное исследование составляет основную теоретическую часть работы и способствовало принятию ключевых решений при реализации прототипа продукта.

4.1.Дискреционная модель

Дискреционная модель (DAC, избирательное управление доступом, произвольное управление доступом) – одна из наиболее известных и распространенных моделей в мире безопасности, появившаяся более 40 лет назад.

Основные понятия DAC – субъекты (пользователи, процессы…), объекты (файлы…)[3] и разрешения (операции, направленные на обработку/взаимодействие с данными[4]). Для каждой пары субъект-объект явно перечисляются доступные привилегии, определяющие санкционированные действия данного субъекта по отношению к данному объекту. Важно отметить, что у каждого объекта существует владелец, который обладает полным набором прав и может делиться ими с другими субъектами. В такой «бесконтрольности» раздачи прав заключается серьезный недостаток с точки зрения безопасности[5]. По ошибке или по злому умыслу владелец конфиденциальных данных может их легко рассекретить.

Обычно DAC реализуется в виде списков контроля доступа (ACL, access control list) или матриц контроля доступа (ACM, access control matrix). Для каждого файла в системе можно хранить список пар (субъект-привилегия). К примеру, дискреционный контроль доступа к файлам в Unix-системах реализован таким образом, что каждый файл обладает атрибутами, описывающими разрешенные действия для разных типов пользователей. (см. Рис.2) [4]:

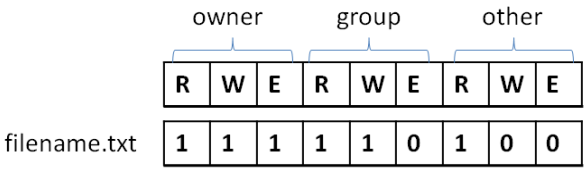


Рис.2. Атрибуты контроля доступа к файлам в Unix

Матрица контроля доступа – единая структура, где сосредоточены данные о многих парах субъект-объект. Столбцы соответствуют объектам системы, строки – субъектам, а в ячейке на пересечении конкретных столбцов и строк стоят соответствующие разрешенные привилегии.

	File 1	File 2	Process 1	Process 2
Process 1	Read, Write, Own	Read	Read, Write, Own, Execute	Write
Process 2	Append	Read, Write, Own	Read	Read, Write, Own, Execute

Табл.2. Пример матрицы контроля доступа [4]

[3] Зачастую множество объектов содержит в себе множество субъектов, что позволяет описывать также и межпроцессное взаимодействие.
[4] Обычно рассматривают три операции: чтение (Read), запись (Write) и исполнение (Execute).
[5] Соответствующую уязвимость называют «Троянский конь»

Формально матрицы контроля доступа можно описать, введя следующие обозначения [3]:

$$S - \text{множество субъектов} \quad O - \text{множество объектов} \quad A - \text{множество привилегий}$$

$$\text{Тогда } M - \text{матрица контроля доступа}, M = (M_{so})_{s \in S, o \in O} \text{ и } M_{so} \subset A$$

Основные операции, предусмотренные данной моделью: Grant (внести разрешение в нужную ячейку ACM), Revoke (удалить разрешение из ячейки), Check (проверить, есть ли требуемое разрешение в определенной ячейке ACM).

Что касается конкретных применений, то DAC реализована практически во всех системах, например, в операционных системах Unix и Windows. В Unix реализуется уже упомянутый механизм Unix file permissions, ставящий в соответствие файлам специальные атрибуты с разрешенными привилегиями. В Windows существует два списка ACL: Discretionary ACL, на основе которого контролируется доступ и операции с файлами, а также специальный System ACL, в котором описано, как система обрабатывает попытки доступа к файлам. [24]

Более подробное описание дискреционной модели контроля доступа можно найти в [3], [5], [4], [41], [32], [33].

4.2.Модели HRU и Graham-Denning

Развитием концепции дискреционного разграничения доступа стала модель Грэхэма-Деннинга (Graham-Denning access model). В ней более строго определены понятия, которыми можно оперировать в контексте рассматриваемой DAC. Все действия в модели представляют собой операции с матрицей контроля доступа. Основа DAC сохранилась: субъекты, объекты, права доступа – однако, чтобы следовать устоявшейся терминологии, немного переименуем эти множества:

$$R - \text{множество привилегий(прав доступа)}$$
$$\text{Тогда } A - \text{матрица контроля доступа}, A = (A_{so})_{s \in S, o \in O} \text{ и } A_{so} \subset R$$

В модели G-D определяется 8 операций и соответствующих пред- и постусловий для работы с матрицей контроля доступа (см.таблицу в приложении 3), а именно: создание/удаление объектов/субъектов; получение, передача, удаление и предоставление прав доступа. Подробности можно найти в [32]. Перейдем сразу к более комплексной модели, создатели которой добились немалых фундаментальных теоретических результатов при исследовании особенностей DAC.

Модель HRU (Harrison-Ruzzo-Ullman) была создана для анализа безопасности систем с дискреционным контролем доступа. Она оперирует все теми же понятиями:

$$S - \text{множество субъектов}$$
$$O - \text{множество объектов}$$
$$R - \text{множество привилегий(прав доступа)}$$
$$\text{Тогда } M - \text{матрица контроля доступа}, M = (M_{so})_{s \in S, o \in O} \text{ и } M_{so} \subset R$$

Однако в рамках HRU определено 6 примитивных операций, действующих на матрицу:

$\boldsymbol{enter}\ r\ into\ M_{so}$	$\boldsymbol{create\ subject}\ s$	$\boldsymbol{create\ object}\ o$
$\boldsymbol{delete}\ r\ from\ M_{so}$	$\boldsymbol{delete\ subject}\ s$	$\boldsymbol{delete\ object}$

Команды HRU имеют строгий формат:

$$\begin{aligned}&\boldsymbol{command}\ c(x_1, \dots, x_k)\\&\quad \boldsymbol{if}\ r_1\ \boldsymbol{in}\ M_{s1,o1}\ \boldsymbol{and}\\&\quad \dots\\&\quad \boldsymbol{if}\ r_m\ \boldsymbol{in}\ M_{sm,om}\\&\quad \boldsymbol{then}\ op_1\ , \dots, op_n\\&\boldsymbol{end}\end{aligned}$$

Т.е. любая команда составляется из предопределенных шести примитивных операций и условных операторов. Можно ввести понятие *моно-операционной команды* – команды, выполняющей только одну примитивную операцию. Аналогично, *моно-условная команда* имеет только один оператор условия, *би-условная команда* – с двумя операторами и т.д.

Допустим, мы хотим отвечать на вопрос: «Не может ли быть такого, что право, подаренное субъектом S1 субъекту S2, возможно, перейдет кому-то еще в результате дальнейшего применения некоторой цепочки команд?»

Информационная система переходит из состояния в состояние, которые описываются матрицей контроля доступа. Говорят, что *состояние* (т.е. матрица контроля доступа) *«теряет» право* **r**, если существует команда $\boldsymbol{c}$, которая добавляет $\boldsymbol{r}$ в ячейку матрицы, которая раньше не содержала этого права[6]. Более формально, существует **s** и **o**, т.ч. $\boldsymbol{r} \notin \boldsymbol{M_{so}}$, а в результате выполнения **c** – $\boldsymbol{r} \in \boldsymbol{M'_{so}}$. [5] Введем определение безопасной конфигурации (под конфигурацией понимается матрица M).

Определение. *Пусть дана система и право* **r**. *Говорится, что конфигурация* $\boldsymbol{Q_0}$ *небезопасна относительно r (или «теряет»* **r***), если существует конфигурация* **Q** *и команда* $\boldsymbol{\alpha}$*, т.ч.:*

- **Q** *достигается из* $\boldsymbol{Q_0}$
- **Q** *«теряет» r в результате действия* $\boldsymbol{\alpha}$

Тогда конфигурация безопасна относительно **r***, если она не является небезопасной.*

Основным теоретическим фактом, который был доказан в терминах модели HRU, является следующая теорема.

Теорема. *Для данной матрицы контроля доступа* **M** *и права* **r** *проверка безопасности* **M** *относительно* **r** *является неразрешимой задачей.*

Существует несколько следствий и уточнений этой теоремы, например:

[6] Заметим, что «потеря» правила не обязательно свидетельствует о бреши в защите: возможно, один субъект законно передал право другому субъекту.

- Проблема определения безопасности разрешима для моно-операционных систем
- Проблема определения безопасности неразрешима для би-условных монотонных[7] систем
- Проблема определения безопасности разрешима для монотонных моно-условных систем

Более подробное описание модели и связанных с ней теоретических результатов можно найти в [5],[8],[33],[32].

Заметим, чем сложнее система, тем труднее ее анализировать, а порой вопрос о безопасности конкретного состояния становится и попросту неразрешимым.

4.3.Модель Take-Grant

Рассмотрим еще одну модель, связанную с дискреционным разделением доступа. Модель Take-Grant была разработана непосредственно для проверки безопасности состояния системы, работающей по принципам DAC. Формально система представляется в виде направленного графа без петель, где узлы – объекты и субъекты, а маркированные дуги указывают права, которые имеет узел. Обычно множество меток состоит из 3-х элементов: r (read), w (write), c (call). В классической модели Take-Grant существуют 5 правил (take, grant, create, call, remove), с помощью которых изменяется граф и, соответственно, состояние системы (см. приложение 4). Применение правила переводит граф G в некоторый модифицированный граф G' ($G \vdash G'$). [2][8]

Вернемся к применению данной модели, а именно к исследованию безопасности системы. Главный вопрос, на который ищется ответ в терминах Take-Grant, звучит следующим образом: «Может ли p обладать правами α относительно q?» (для краткости будем писать pαq). Ответ формулируется в виде ключевой теоремы. [2]

Теорема. *Пусть p и q – различные вершины графа, α – метка. Тогда условия* ***1*** *и* ***2*** *являются необходимыми и достаточными, чтобы* ***pαq:***

Усл.1. ***p*** *и* ***q*** *соединены*[8] *в графе G*

Усл.2. Существует вершина **x** в G и дуга $\overrightarrow{xq}$ *с меткой* β*, т.ч.*

$$\text{либо } \alpha = r \Rightarrow \{r, c\} \cap \beta \neq \emptyset$$
$$\text{либо } \alpha = w \Rightarrow w \in \beta$$
$$\text{либо } \alpha = c \Rightarrow c \in \beta$$

Т.о. существует алгоритм проверки свойства системы " **pαq** ". Более того, он линеен, т.к. оба условия из теоремы проверяются линейно в зависимости от размеров графа.

На этом модель Take-Grant не исчерпывает себя. Существует ряд расширений модели. Например, в [17] описаны иерархические, древовидные, ациклические системы защиты. Для каждого типа определены «протоколы», представляющие собой алгоритмы

[7] Монотонность подразумевает, что права, однажды внесенные в матрицу, больше не удаляются из нее.
[8] Т.е. существует путь в графе G (не обязательно направленный) между p и q

модификации соответствующих графов. Работы Мэтта Бишопа [18],[35] также посвящены иерархическим системам безопасности. Модель Take-Grant расширена такими правилами Post, Pass, Spy, Find, отражающими возможные неявные пути получения доступа к данным в системе. Рассматривается и анализируется предикат can_know (x,y), принимающий значения true, например, если x может читать y, y может писать в x.

Итак, дискреционный контроль доступа – очень гибкий метод, позволяющий конкретизировать правила доступа с высокой гранулярностью. Также можно отметить интуитивность модели и ее эффективность для задач малых организаций. Среди проблем, как уже отмечалось, главной является уязвимость типа «Троянский конь». Плюс, стоит упомянуть о сложностях в поддержке больших DAC-систем и их верификации. [22]

На этом можно закончить знакомство с дискреционным контролем доступа и переходить к следующему типу моделей.

4.4.Мандатная модель

Вспомним, что в рамках дискреционной модели владелец данных может произвольно раздавать пользователям права доступа, что является значительной брешью в защите. Около 40 лет назад была разработана мандатная (принудительная; MAC, Mandatory Access Control[9]) модель контроля доступа, которая устраняет эту проблему. Основная идея MAC заключается в том, что конкретный пользователь (даже владелец) не может изменять права доступа к файлам. Система должна проверять, может ли определенный субъект работать с требуемым объектом. Цель MAC – сохранение целостности и конфиденциальности информации и предотвращение возможности изменения прав доступа пользователями [4]. Зачастую мандатная модель дополняет, например, дискреционную.

В основе MAC лежит концепция MLS (multilevel secure): каждый объект в системе имеет классификацию (метка безопасности), каждый субъект имеет «уровень видимости» (допуск, метка безопасности)[10], задача системы контроля – сравнивать метки и тем самым определять возможность доступа.

Классы доступа объектов и субъектов состоят в общем случае из двух компонент[11] [4].

- *Метка безопасности* – элемент линейно упорядоченного множества. Например, уровни секретности документов: Unclassified < Confidential < Secret < TopSecret[12].
- *Множество категорий* – список имен тематических областей, к которым принадлежит объект. Например, {Nato, Nuclear, Navy} или {Хирургия, Кардиология, Приемное отделение}.

[9] В англоязычной литературе иногда встречается под родственным названием MLS (multilevel secure).
[10] Метки объектов и субъектов называют также классами доступа
[11] Иногда в MAC ограничиваются только одной компонентой класса доступа – меткой безопасности.
[12] Представленные уровни секретности документов используются в сфере безопасности в США. Аналогичны уровням секретности в РФ.

Таким образом, некоторый объект O может иметь класс $(Confidential, \{Nato\})$; некоторый субъект может иметь класс $(Secret, \{Nato, Navy\})$.

На множестве классов вводится частичное отношение порядка – отношение доминирования: класс AC_1 доминирует над классом AC_2, если метка безопасности первого класса больше второго и второе множество категорий содержится в первом:

$$AC_1\ dom\ AC_2 \Leftrightarrow \begin{cases} AC_1.МеткаБезопасности \geq AC_2.МеткаБезопасности \\ AC_1.Категории \supseteq AC_2.Категории \end{cases}$$

Например, $AC_1 = (Secret, \{Nato, Navy\})$ доминирует над $AC_2 = (Confidential, \{Nato\})$, т.к. $Secret \geq Confidential$ и $\{Nato, Navy\} \supseteq \{Navy\}$. Существуют также несравнимые классы, например, $AC_1 = (Secret, \{Nato, Navy\})$ и $AC_2 = (Secret, \{Nato, Nuclear\})$, т.к. ни одно множество категорий не содержится в другом.

Таким образом, множество классов образует решетку с отношением доминирования. На Рис.3 приведен пример решетки для всех возможных классов, основанных на множестве меток $Secret \leq TopSecret$ и множестве категорий $\{Army, Nuclear\}$ [4]. Доминирование обозначено стрелками.

$AC_1 = (Secret, \{\emptyset\})$
$AC_2 = (Secret, \{Army\})$
$AC_3 = (Secret, \{Nuclear\})$
$AC_4 = (Secret, \{Army, Nuclear\})$
$AC_5 = (TopSecret, \{\emptyset\})$
$AC_6 = (TopSecret, \{Army\})$
$AC_7 = (TopSecret, \{Nuclear\})$
$AC_8 = (TopSecret, \{Army, Nuclear\})$

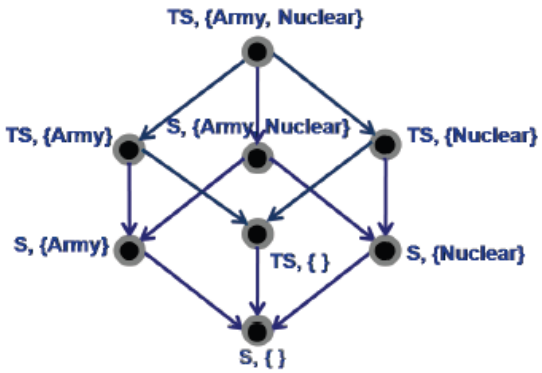


Рис.3. Пример решетки MAC

Мандатные системы контроля доступа являются основными в военных, правительственных и других серьезных организациях. В качестве примеров продуктов, в которых реализован MAC, можно привести разработки, сертифицированные на высокие уровни безопасности по различным стандартам: SELinux (используется в Red Hat Enterprise Linux, SUSE – Common Criteria EAL4+), Windows Mandatory Integrity Control (Windows Server 2008 R2, Windows 7 – CC EAL4+), Astra Linux Special Edition (сертификаты ФСТЭК, ФСБ, Минобороны. Класс 1), СУБД Линтер (ФСТЭК. Класс 2), Oracle Database (Oracle Label Security – CC EAL4+)…

4.4.1. Модель Белла-ЛаПадулы (Bell-LaPadula)

Данная модель (1973 г., Д. Белл и Леонард Ла Падула) основана на концепции MAC и реализует контроль конфиденциальности данных с помощью введения специальных правил проверки прав доступа [4],[21],[30],[33].

- **NO-READ-UP (Simple Security Property)**
  Субъекту разрешается читать объект O, только если класс субъекта доминирует над классом объекта[13]
- **NO-WRITE-DOWN (*-Property)**
  Субъекту разрешается запись в объект O, только если класс объекта доминирует над классом субъектом

Эти правила являются естественными для обеспечения конфиденциальности данных. NO-READ-UP запрещает чтение файлов, имеющих уровень секретности больше, чем уровень доступа пользователя. NO-WRITE-DOWN запрещает «рассекречивать» информацию, т.е. записывать информацию в файлы, имеющие меньший уровень секретности, чем текущий уровень пользователя.

В подобных мандатных системах существует ряд проблем. Например, представим ситуацию: полковник имеет класс доступа $(Secret, \{Nuclear, Army\})$, а майор – $(Secret, \{Army\})$. Полковник хочет послать сообщение майору, однако из-за правила NO-READ-DOWN не может написать документ с более низким уровнем секретности. Чтобы решать подобные задачи, можно ввести для пользователей *максимальный класс доступа* и *текущий класс доступа*. Субъект может изменять класс доступа, но максимальный класс всегда должен доминировать над текущим.

Основным теоретическим результатом является ***основная теорема безопасности***. Введем сначала несколько определений.

$L -$ *решетка классов доступа*
$F: S \cup O \to L -$ *функция, сопоставляющая класс объектам* (O) *и субъектам* (S)
$V = \{(F, M)\} -$ *множество состояний системы, где*
$M -$ *матрица контроля доступа, как в модели* HRU $(см. ранее)$

Система безопасности состоит из:

$v_0 -$ *начальное состояние системы*
$R -$ *набор запросов*
$T: (V \times R) \to V -$ *функция перевода системы из состояния в состояние*

Определение. *Состояние* (F, M) *называется безопасным относительно чтения (read-secure, simple security) тогда и только тогда, когда*

$$\forall s \in S \; и \; \forall o \in O: read \in M[s, o] \Rightarrow F(s) \geq F(o)$$

Определение. *Состояние* (F, M)*называется безопасным относительно записи (write-secure, *-property) тогда и только тогда, когда*

$$\forall s \in S \; и \; \forall o \in O: write \in M[s, o] \Rightarrow F(o) \geq F(s)$$

[13] В случае дополнительного дискреционного доступа добавляется соответствующее условие «субъект имеет дискреционное право чтения объекта O». Аналогично и в других правилах в дальнейшем.

Определение. *Состояние* (F, M)*называется безопасным тогда и только тогда, когда оно безопасно относительно чтения и относительно записи.*

Определение. *Система* (υ_0, R, T) *безопасна тогда и только тогда, когда* υ_0 *безопасное состояние и каждое состояние, достижимое из него в результате выполнения конечного количества запросов из* R*, также безопасно.*

Теперь можно сформулировать ***основную теорему безопасности (Basic Security Theorem)*** [33],[30]:

Теорема (BST). *Система* (υ_0, R, T) *безопасна тогда и только тогда, когда* ***(1)*** υ_0 *безопасно и* ***(2)*** T *такое, что для каждого* υ*, достижимого из* υ_0 *последовательным применением конечного числа запросов из* R*,* ***если*** $T(\upsilon, c) = \upsilon^*$*, где* $\upsilon = (F, M)$ *и* $\upsilon^* = (F^*, M^*)$*,* ***то*** *для каждого* $s \in S$ *и* $o \in O$*:*

- Если $read \in M^*[s, o]$ и $read \notin M[s, o]$, то $F^*(s) \geq F^*(o)$;
- Если $read \in M[s, o]$ и $F^*(s) \not\geq F^*(o)$, то $read \notin M^*[s, o]$;
- Если $write \in M^*[s, o]$ и $write \notin M[s, o]$, то $F^*(o) \geq F^*(s)$ и
- Если $write \in M[s, o]$ и $F^*(o) \not\geq F^*(s)$, то $write \notin M^*[s, o]$.

4.4.2. Модель Биба (Biba)

Данная модель основана на концепции MAC и реализует контроль целостности данных с помощью введения специальных правил проверки прав доступа [4], [20].

- **NO-WRITE-UP (Simple Security Property)**
  Субъекту разрешается читать объект O, только если класс субъекта доминирует над классом объекта[14]
- **NO-READ-DOWN (*-Property)**
  Субъекту разрешается запись в объект O, только если класс объекта доминирует над классом субъектом

Эти правила являются естественными для обеспечения целостности данных и «двойственными» к правилам модели Bell-LaPadula. NO-WRITE-UP запрещает запись данных в файлы, имеющих более высокий уровень целостности. NO-READ-DOWN запрещает читать информацию из менее целостных файлов.

Итак, мандатные системы контроля доступа широко используются в серьезных организациях, в том числе военных. Они дополняют дискреционный контроль доступа, позволяя избавиться от возможности атак типа «Троянский конь». Однако с учетом решеточной структуры модели становится затруднительным динамическое/частое изменение политик безопасности. Плюс ко всему, большая часть операционной системы должна быть «доверенной» и находиться вне модели (например, процессы и библиотеки, которые верифицируют MAC-модель и ее правила). Т.о. необходимо реализовать дополнительную систему безопасности, контролирующую доступ к этим отдельным ресурсам. Поэтому MAC системы дороги и сложны в реализации.

[14] В случае дополнительного дискреционного доступа добавляется соответствующее условие «субъект имеет дискреционное право чтения объекта O». Аналогично и в других правилах в дальнейшем.

4.5.Модель Chinese Wall

Модель Chinese Wall (Китайская стена) сочетает элементы DAC и MAC. Была разработана в 1989 году Д.Брюэром и М.Нэшем (David F.C. Brewer, Michael J.Nash) [34]. Лучше всего суть модели можно пояснить на примере из финансового мира, где она и получила наибольшее распространение. Рассмотрим аналитика, работающего на некоторое финансовое агентство, которое предоставляет корпоративные услуги для бизнеса [6]. Аналитик должен сохранять конфиденциальность данных клиентов (стратегии, планы, материальная составляющая…) Т.е. он не может давать всяческие советы и рекомендации компаниям-конкурентам. Однако аналитик может свободно рекомендовать что-то компаниям, не являющимся конкурентами.

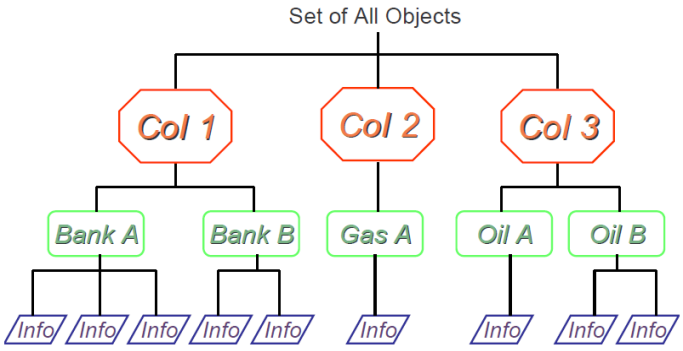


Рис.4. Модель Chinese Wall

Основная черта модели – «динамическая» установка прав доступа с учетом ранее полученных привилегий. Среди основных элементов модели можно выделить следующие сущности. Субъекты – активные сущности, взаимодействующие с защищенными объектами. Объекты - данные, разделенные на три класса: Информация (Information), Множество Данных (DataSet), Классы конфликтов интересов (Conflict-of-Interest classes, CoI). Правила доступа: правило чтения и правило записи. На Рис.4 изображен пример модели, в которой есть три области интересов: Банковская сфера, Газовые и Нефтяные компании; компании: Bank A, Bank B, Gas A, Oil A, Oil B; некоторая внутренняя информация компаний Info.

Правило чтения. Субъект S может читать объект O, если:

- O находится в том же DataSet, что и объект, к которому S уже доступился **ИЛИ**
- O принадлежит другому CoI, в котором S еще не доступался к информации.

Однако правило чтения не предотвращает косвенное рассекречивание информации. Пусть Джон имеет доступ к OilA и BankA,а Джейн – к OilB и BankA. Тогда, если Джону разрешено читать OilA и писать BankA, то Джейн может передать данные о OilA в OilB.

Правило записи. Субъект S может писать в объект O, если:

- S может читать O согласно правилу чтения **И**
- Ни один объект еще не был прочитан S из DataSet, отличного от того, куда планируется запись.

Таким образом, заметим, что поток информации с учетом правила записи ограничился DataSet’ом одной компании. Также из-за правила записи пользователь, прочитавший данные из более, чем одного DataSet’а, не сможет осуществить запись больше никуда. Метафорически, вокруг данных компании построена Китайская стена. Модель существенно ограничивает манипуляции с данными. Существуют методы обезличивания информации, однако они выходят за рамки данной дипломной работы.

4.6.Модель Clark-Wilson

Разработана в 1987 году Кларком и Уилсоном (David D. Clark, David R. Wilson). [1] Является неформальной моделью без строгой теоретической математической привязки. Цель модели C-W – сохранение целостности данных. Основные элементы модели:

- CDI/UDI – Constrained/Unconstrained data item – «доверенный/недоверенный» элемент данных. Т.е. данные делятся на те, которые удовлетворяют набору правил, заданных в системе, и наоборот.
- TP – transform procedures – процедуры преобразования данных. Обязательно выдают CDI-данные.

$$TP: CDI|UDI \rightarrow CDI$$

- IVP – integrity verification procedure – процедура, контролирующая работу системы, в том числе валидность CDI.

Система безопасности на основе C-W модели функционирует согласно набору правил, касающихся взаимоотношений данных и процедур преобразования. Список правил приведен в приложении 5. Детальное описание модели Clark-Wilson можно найти в [1].

4.7.Ролевая модель

Ролевая модель (Role-Based Access Control), разработанная в 1992 году [29], послужила новым витком эволюции моделей контроля доступа. Ее оформили в виде стандарта ANSI/INCITS[15] в 2004 году. Данная модель является естественным отражением должностных структур организаций и служебных обязанностей пользователей. RBAC основана на присвоении пользователям «ролей», представляющих собой множество разрешенных «функций», т.е. пар {действие, объект}. Можно сказать, что «роль» аналогична должности в компании. [29]

4.7.1. Описание модели

Классическая модель RBAC (известная как $RBAC_0$) основана на следующих понятиях [28], [29].

- Пользователь (User) – человек, машина, процесс…
- Роль (Role) – функция в контексте организации с соответствующей семантикой
- Привилегия (Permission) – способ доступа к объектам в системе
- Сессия (Session) – сеанс подключения пользователя (User) к системе, определяющий набор активных ролей

В [27], [28], [29] можно найти основные отношения и определения, необходимые для разговора в терминах ролевой модели (в том числе, отмеченные на Рис.5 RA, UA, PA RSA).

[15] http://csrc.nist.gov/groups/SNS/rbac/index.html

Определение. **RBAC$_0$** состоит из:

- U, R, P, S – (Users, Roles, Permission, Sessions)
- $PA \subseteq P \times R$ отношение многие ко многим (Permission-to-Role Assignment)
- $UA \subseteq U \times R$ отношение многие ко многим (User-to-Role Assignment)
- $user: S \rightarrow U$ функция, сопоставляющая сессии s определенного пользователя user(s). Не изменяется в течение сессии.
- $roles: S \rightarrow 2^R$ функция, сопоставляющая сессии s набор ролей $roles(s) \subseteq \{r|\ (user(s), r) \in UA\}$ (может меняться со временем).
  Сессия s имеет привилегии $\cup_{r \in roles(s)} \{p|(p, r) \in PA\}$

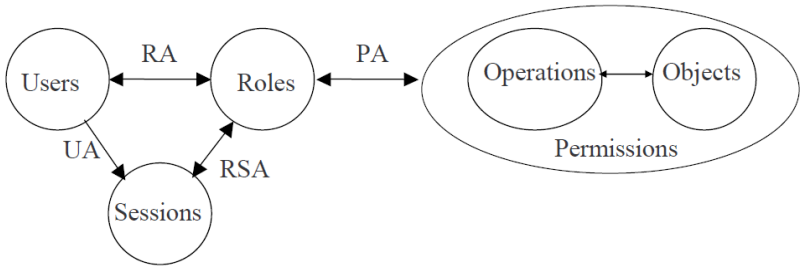


Рис.5. RBAC$_0$

Рисунок показывает модель RBAC$_0$ и соответствующие определенные понятия.

Дальнейшим расширением базовой модели является внедрение иерархий ролей (RBAC$_1$). Ролевые иерархии являются естественным способом моделирования должностных структур реальных организаций.

Определение. **RBAC$_1$** состоит из:

- U, R, P, S, PA, UA и user – как в RBAC$_0$
- $PH \subseteq R \times R$ частичный порядок на R ("≥"), называемый ролевой иерархией.
- $roles: S \rightarrow 2^R$ функция, измененная относительно RBAC$_0$.
  $roles(s) \subseteq \{ r|\ (\exists\ r' \geq r)\ [(user(s), r') \in UA] \}$ (может меняться со временем).
  Сессия s имеет привилегии $\cup_{r \in roles(s)} \{p|(\exists\ r'' \geq r)\ [(p, r'') \in PA]\}$

Следующая модель добавляет «ограничения» (RBAC$_2$). Ограничение может касаться любого элемента модели RBAC.

Определение. **RBAC$_2$** не отличается ничем от RBAC$_0$, кроме как введенными ограничениями, которые определяют, допустимы или нет определенные значения компонент RBAC$_0$.

К примеру, во многих организациях не разрешено быть членом двух ролей одновременно – такое ограничение носит название «разделение обязанностей» (Separation of Duties, mutually exclusive roles). Или существуют такие ограничения, как cardinality – количество членов конкретной роли (возможны роли, которые могут быть присвоены определенному количеству пользователей). Более детальная информация об ограничениях содержится в [23].

Модель, сочетающая базовую и оба расширения, называется RBAC$_3$.

Считается, что обычные привилегии могут касаться только данных. Однако можно ввести специальные административные права, способные модифицировать объекты самой модели. Организация управляющей модели ARBAC ничем не отличается от классической. Аналогично вводятся расширения административной модели $ARBAC_0$-$ARBAC_3$. В приложении 6 приведена схема наиболее общей системы, включающей в себя все возможные компоненты как базовой, так и управляющей моделей.

4.7.2. Администрирование RBAC. Ролевой граф

В [7] рассматриваются способы администрирования модели RBAC. Вводится понятие графа привилегий (Baldwin's privilege graph) – ациклический граф с тремя типами узлов: привилегия (конечные вершины), роль, пользователь (стартовые вершины). Здесь предусматривается поддержка ролевых иерархий, т.е. если есть направленное ребро $\langle v_i, v_j \rangle$, то множества привилегий $\Psi(v_i) \supseteq \Psi(v_j)$ (см. приложение 7). Можно искусственно создать специальные минимальную и максимальную роли:

$$\Psi(MinRole) = \begin{cases} минимальный\ набор\ привилегий, если\ определен \\ \emptyset, иначе \end{cases}$$

$$\Psi(MaxRole) = \bigcup_{r \in R} \Psi(r)$$

Определив несколько дополнительных отношений (is-junior '→', common-junior '⊙', common-senior '⊕'), можно рассмотреть модель Ролевого Графа (Role Graph). Формально это граф $RG = (R, \rightarrow)$. Т.е. ребра RG строятся исходя из отношения is-junior. На Рис.6 представлен пример, а в Табл.3 – соответствующие наборы привилегий.

Role	Direct (D)	Indirect (I)	Effective ($D \cup I$)
A	{1}	{}	{1}
B	{2}	{}	{2}
C	{3}	{}	{3}
D	{4}	{}	{4}
E	{5}	{1,2}	{1,2,5}
F	{6}	{3}	{3,6}
G	{7,8}	{4}	{4,7,8}
H	{9,10}	{1,2,5}	{1,2,5,9,10}
I	{11,12}	{1,2,3,4,5,6,7,8}	{1,2,3,4,5,6,7,8,11,12}

Табл.3. Привилегии RG

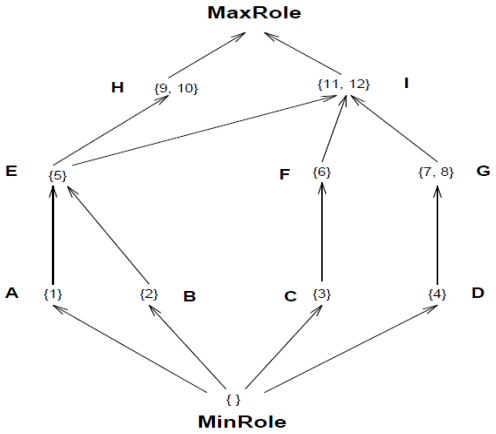


Рис.6. Role Graph

Если поддерживать RG в «хорошем» состоянии (в любой момент он должен оставаться транзитивно сокращенным и иметь в узлах только непосредственные привилегии) то можно легко определить алгоритмы управления моделью в терминах графа: добавление/удаление и вертикальное/горизонтальное разделение ролей. Очевидным образом RG трансформируется в дерево иерархий или в граф привилегий (и наоборот). [7]

Ролевая модель по праву считается более общей моделью, нежели DAC и MAC (подробнее об этом пойдет речь позже). Пока что заметим, что основное отличие от DAC состоит в том, что привилегии даются не каждому пользователю непосредственно, а

ролям, которые уже потом присваиваются определенным пользователям. Это позволяет избежать проблемы произвольной передачи привилегий. В некотором смысле RBAC – это гибкая («точечная») форма MAC, но без требований MLS [29]. Есть в ролевой модели и что-то от Clark-Wilson: права даются в виде выше упомянутых «процедур» (привилегий типа {действие-объект}) [22].

Что касается проблемы определения безопасного состояния, то в RBAC они формулируются и решаются намного сложнее, нежели в описанных ранее MAC и DAC [31].

Неоспоримые плюсы RBAC: возможность контроля доступа с высокой гранулярностью; в больших организациях авторизация роли сразу для нескольких пользователей облегчает задачу управления политикой безопасности. Отметим также, что в терминах ролевой модели очень легко реализуются принцип «минимальной привилегии» (привилегии, которая доступна всем), принцип разделения обязанностей (например, запрет активации определенных ролей одним пользователем), централизованное администрирование модели. Проблемы в очень больших системах, увы, по-прежнему присутствуют: наличие ролевых иерархий и наследования существенно усложняют процесс управления политиками. Также существуют проблемы с контролем безопасного состояния системы.[22]

Среди современных систем, реализующих в том или ином виде ролевой подход, можно упомянуть: SELinux, MS Exchange Server 2013, MS Active Directory, Oracle Enterprise Repository… В большинстве случаев концепция RBAC применяется для выдачи пользователям привилегий по управлению своими ресурсами в системе (например, создать LUN в СХД, осуществить миграцию данных, создать списки рассылки и т.п.)

4.8.Выражение MAC через RBAC

В терминах RBAC можно выразить мандатную модель контроля доступа (не только классическую, но и в различных модификациях). Построение основано на модели $RBAC_3$. [13], [26] Напомним, что мандатная модель реализуется на решетках классов доступа с частичным отношением порядка. Модифицируем уже известное правило **-Property*, подразделив его на два.

Определение. Нестрогое *-правило (Liberal *-Property). Субъект s может писать в объект o, если $\lambda(s) \geq \lambda(o)$, где λ – функция, выдающая метку объекта/субъекта.

Определение. Строгое *-правило (Strict *-Property). Субъект s может писать в объект o, если $\lambda(s) = \lambda(o)$.

Моделирование решетки Bell-LaPadula.

Основная идея состоит в том, чтобы создать ролевую иерархию (РИ), соответствующую структуре решетки MAC. Нужно помнить, что чем выше роль в РИ, тем больше у нее привилегий (далее на рисунках доминирующие роли сверху). Т.о. придется использовать две

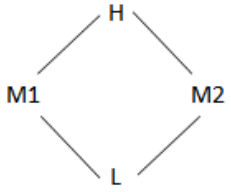


Рис.7. Решетка MAC

различных РИ, чтобы удовлетворить и правилу чтения, и правилу записи в МАС. Будем для простоты рассматривать решетку (на Рис.7) с 4 метками: High, Medium1 и 2, Low.

Тогда в случае моделирования МАС с нестрогим правилом записи получим следующую RBAC [13], [26]:

- $R = \{HR, M1R, M2R, LR, HW, M1W, M2W, LW\}$ – роли для чтения и записи
- RH – см. Рис.8.
- $P = \{(o, r), (o, w) | o - объект\ в\ системе\}$
- Ограничение на UA: каждому пользователю присваивается ровно 2 роли: xR и xW
- Ограничение на сессии: каждая сессия активирует ровно 2 роли: yR и yW
- Ограничение на PA:
  - (o,r) присваивается xR т.т.т.[16] (o,w) присваивается xW
  - (o,r) присваивается ровно одной роли xR

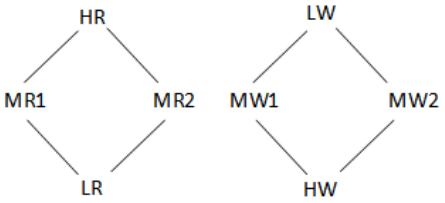


Рис.8. Иерархии RBAC, нестрогая запись

При моделировании МАС со строгим правилом записи получим ту же RBAC, но РИ для записи станет дискретной (см. Рис.9):

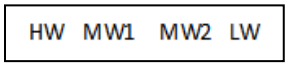


Рис.9. РИ при строгом *-правиле

С небольшими поправками реализуются возможные модификации МАС (см. [13], [26]):

- *МАС с доверенным интервалом записи (Trusted write range).* Появляется два типа меток субъектов: λr – для чтения, λw – для записи, причем $\lambda w \leq \lambda r$.
- *МАС с независимым интервалом записи (Independent Write Range).* На метки чтения и записи не накладывается ограничение доминирования.
- *МАС с «фиксированной» записью (Strict *-property with designated write).* Аналогично предыдущему варианту с правилом строгой записи.

Моделирование решеток Bell-LaPadula + Biba.

При добавлении контроля целостности возникает потребность в усложнении соответствующей структуры RBAC. Правила в модели Biba двойственны относительно правил в Bell-LaPadula, поэтому и ролевая модель будет похожа на уже рассмотренную нами. [13], [26]

Для простоты определим решетки из двух меток. (см. Рис.10):

Рис.10. Пример решеток

Так же, как и в предыдущем случае, определяются строгое/нестрогое правила для записи с учетом целостности. Тогда модель RBAC, моделирующая МАС в случае нестрогих правил конфиденциальной и целостной записи, выглядит следующим образом. [13], [26]

- $R = \{HSR, LSR, HSW, LSW, HIR, LIR, HIW, LIW\}$ – отдельные роли для чтения и записи для контроля целостности (I) и конфиденциальности (S).
- RH – см.Рис.11.

[16] Тогда и только тогда

- $P = \{(o, r), (o, w) | o - \textit{объект в системе}\}$
- Ограничение на UA: каждому пользователю выдаются 2 пары ролей: xSR, xSW и yIR, yIW.
- Ограничение на сессии: каждая сессия активирует ровно 2 пары ролей: xSR, xSW и yIR, yIW.
- Ограничение на PA:
  - (o,r) присваивается xSR т.т.т.
    (o,w) присваивается xSW
  - (o,r) присваивается ровно одной роли xSR
  - (o,r) присваивается yIR т.т.т.
  - (o,w) присваивается yIW
  - (o,r) присваивается ровно одной роли yIR

Рис.11. Иерархии RBAC

Аналогично могут быть представлены оставшиеся сочетания строгой/нестрогой конфиденциальности/целостности.

Комплексную MAC-модель конфиденциальности и целостности можно свести от двух решеток к одной. Тогда становится возможным модифицировать соответствующую RBAC, в которой останется одна «конфиденциально-целостная» ролевая иерархия для чтения, но все так же потребуются несколько РИ для контроля записи. [13], [26]

Стоит заметить, что через MAC тоже можно выразить RBAC, однако с существенными ограничениями на ролевые иерархии, которые должны иметь структуру дерева [25]. Т.о. класс моделей RBAC шире, чем класс MAC-моделей.

4.9. Выражение DAC через RBAC

В терминах RBAC также можно выразить и модель дискреционного контроля доступа и ее различные модификации. [13][19] Из-за того, что DAC-модель обладает высокой «гранулярностью» (с точностью до объекта), то размеры структуры RBAC существенно возрастают. Для моделирования потребуется использовать как $RBAC_3$, так и $ARBAC_3$. Рассмотренные вариации DAC:

- *Строгий DAC (Strict DAC)*. Владелец объекта не меняется. Только он «дарит» привилегии.
- *Нестрогий DAC (Liberal DAC)*. Владелец может передавать другим пользователям право «дарить» привилегии.
- DAC с изменяющимся владельцем (DAC with change of Ownership).
- Grant-Dependent/Grant-Independent Revocation. Привилегии могут быть отозваны только тем пользователем, кто их дал, или не обязательно.

Общая концепция моделирования DAC

В общем случае при создании объекта *O* необходимо выполнить следующие действия. Создать три административные роли в ARBAC: OWN_O, PARENT_O, PARENTwithGRANT_O; одну роль в RBAC: READ_O. Роль OWN_O должна иметь право добавлять и удалять пользователей роли PARENTwithGRANT_O, роль PARENTwithGRANT_O имеет право добавлять/удалять пользователей роли PARENT_O. Также необходимо создать еще 8 привилегий:

- canRead_O. Позволяет пользователю роли READ_O читать объект.
- destroyObject_O. Право OWN_O на удаление объекта.
- add/remReadUser_O. Права PARENT_O на добавление/удаление читателей.
- add/remParent_O. Права PARENTwithGRANT_O на добавление/удаление Parent_O.
- add/remParentWithGrant_O. Права OWN_O на добавление/удаление ParentWithGrant_O.

Для удаления объекта необходимо удалить роли OWN_O, PARENT_O, PARENTwithGRANT_O, READ_O вместе с привилегиями. Структура описанной ролевой модели представлена на Рис.12:

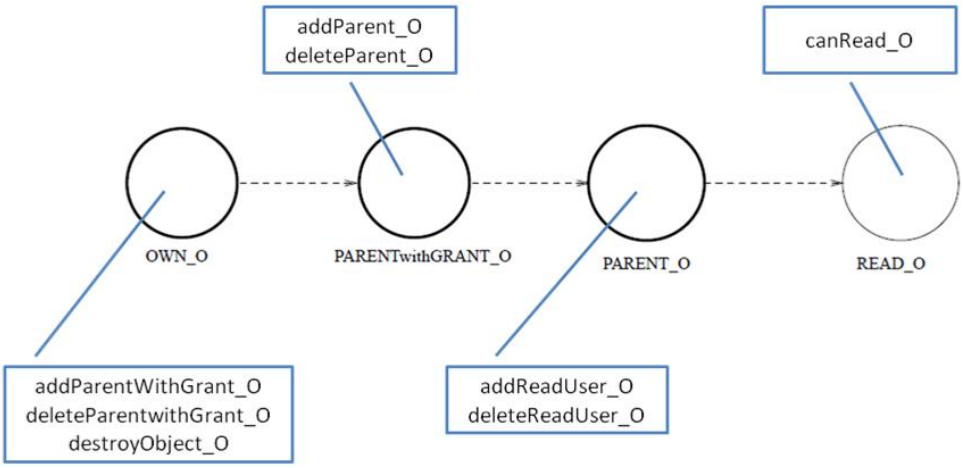


Рис.12. DAC в терминах RBAC

Различные варианты модели DAC моделируются путем введения дополнительных ролей, либо с помощью добавления ряда ограничений модели RBAC. [13], [19].

Т.о. класс RBAC шире, чем класс DAC моделей.

4.10. Выводы

Итак, доказано, что класс RBAC шире, нежели классы мандатной и дискреционной моделей. Конечно, как мы могли заметить, существуют довольно большие накладные расходы при моделировании, однако в среде, где требуется реализация сразу нескольких различных моделей, выбор единого «базиса» может оказаться существенно дешевле, проще, безопаснее. Особенно, когда этот базис – это ролевая модель, сочетающая в себе гибкость DAC, уровень безопасности MAC и относительную простоту управления.

Таким образом, в качестве универсальной основы для реализации системы контроля доступа в проекте MCD была выбрана ролевая модель.

5. Прототип инструмента управления политиками контроля доступа

Итак, с учетом сказанного выше было принято решение доказать возможность реализации системы безопасности, основанной на ролевой модели контроля доступа.

В этой главе рассматриваются основные аспекты построения прототипа инструмента управления политиками контроля доступа на основе RBAC модели. Под управлением понимается набор действий/возможностей, среди которых:

- конфигурация рассматриваемой RBAC, поддержка операций с элементами моделей в рамках построенной архитектуры, проектирование политик (создание, удаление, изменение элементов моделей);
- поддержка согласованности и непротиворечивости созданных политик и моделей и соответствующих структур данных;
- выражение других моделей контроля доступа в терминах RBAC;
- контроль выполнения политик и выдача прав пользователям, применение созданных политик в системе MCD;
- отслеживание проблем, связанных с определением безопасного состояния системы, при построении политик;
- визуализация политик (отображение данных в удобном и понятном виде)
- визуальное управление политиками (графическое создание элементов моделей и модификация существующих данных)
  и т.п.

Перед разработкой были отмечены некоторые специфические особенности, присущие подобному ПО. К примеру, важный аспект любой системы безопасности – «не надоедать» и «не перегружать» пользователя. Обычно все конфигурационные и визуализирующие возможности – это лишь вершина айсберга. На самом деле, контроль за безопасностью ведется везде: от системных недр до окошек ввода пароля. Т.о. важно понимать, что конфигуратор должен быть простым и удобным, т.к. под ним таится сложная архитектура, непонятная обычному пользователю. Системе MCD добавляет сложности единый базис RBAC, поэтому важно предусмотреть шаблоны моделируемых ПКД[17]: DAC и MAC (ACL/ACM или метки-субъекты-объекты должны переводиться в элементы ролевой модели, как описывалось ранее). Отметим также, что в организациях, по-настоящему заботящихся о безопасности своих ресурсов, стараются не допускать появления неразрешимостей касаемо безопасного состояния системы. Следовательно, возникает необходимость использования описанных выше идей проверки защищенности системы.

Конечно, в существующих системах уже реализованы собственные подсистемы контроля доступа и конфигурирующие инструменты. В качестве примеров можно привести MS Windows (Local Security Policy Editor, Active Directory Users and Computers tool и т.п.[18]); Linux (SELinux, AppArmor); различные СУБД, СХД и служебное ПО (продукты Oracle, EMC[2])… Зачастую подобные утилиты недостаточно хорошо визуализируют информацию:

[17] Политика Контроля Доступа
[18] http://technet.microsoft.com/en-us/library/dd277306.aspx

либо они консольные, либо отображают данные в огромных перегруженных таблицах. Плюс ко всему, упомянутые инструменты занимаются контролем действий в конкретной ОС или управлением специфичной архитектурой. Поэтому их заимствование для решения задач проекта MCD невозможно.

Конечно, реализация полноценного инструмента управления ПКД – огромная задача, особенно с учетом специфических требований проекта MCD. В данной работе поставлен вопрос о возможности построения подобной системы, на который найден положительный ответ. Перейдем непосредственно к реализации прототипа инструмента управления политиками контроля доступа на основе ролевой модели под названием PMTool – Policy Management Tool.

5.1. Обзор функциональности

Для доказательства возможности реализации PMTool была поставлена задача создания базовой «инфраструктуры» для работы с ролевой моделью:

- Реализация системы хранения для базовой модели RBAC.
  Разработка архитектуры и создание структуры хранения элементов ролевой модели в виде некоторой базы данных.
- Реализация основной функциональности по управлению ПКД на основе RBAC
  Здесь подразумевается управление моделью $RBAC_0$ с учетом поддержки согласованности и корректности данных и соблюдения необходимых ограничений/правил модели.
- Создание политики средствами GUI
  Имеется в виду разработка некоторого GUI, который будет скрывать сложность системы безопасности в целом и выдавать лишь высокоуровневое управление.
- Импорт/Экспорт данных (например, с помощью XML-файла)
  Необходимость экспорта/импорта проявляется, например, в случае переноса имеющихся политик из одной системы в другую такую же. Плюс, реализация такого внесения данных доказывает возможность импорта политик из других существующих систем путем модификации процедуры разбора для соответствующего конфигурационного файла.
- Визуализация некоторых отношений модели в виде диаграмм/схем
  Визуализация позволит упростить восприятие системы безопасности пользователем, даст наглядное понимание того, как сконфигурированы политики. Это является также первым шагом к реализации подхода визуального моделирования – в некотором смысле «двойственного» действия относительно визуализации.

5.2. Используемые технологии

В качестве языка программирования и среды разработки были выбраны C# и MS Visual Studio 2012 соответственно. C# является широко используемым высокоуровневым объектно-ориентированным языком. Он интуитивно понятен, имеет развитое «окружение» в виде огромного количества различных библиотек, на нем легко

программируются GUI приложения. Также стоит отметить наличие LINQtoSQL[19] – способа работы с базами данных в объектно-ориентированном стиле. Visual Studio[20] – современное удобное средство разработки программных продуктов любой сложности, помимо редактора кода поддерживающее взаимодействие с различными инструментами, например MS SQL Server.

Для хранения элементов «базисной» модели RBAC было решено использовать реляционную базу данных SQL и соответствующую СУРБД MS SQL Server 2012[21], т.к. ее относительно легко использовать вкупе с остальными выбранными технологиями. SQL-сервер был развернут на локальной машине и интегрирован в Visual Studio, как упоминалось выше.

Для визуализации данных выбран MS Visio 2013[22] – инструмент, обладающий впечатляющими возможностями построения схем и диаграмм с помощью разнообразных шаблонов. В данной работе используется Visio SDK и Visio API [36], предоставляемый для языка C#. Для визуализации некоторых отношений ролевой модели был выбран ряд диаграмм UML: Use Case, Class, Object (об этом пойдет речь позже).

Реализация пакетного внесения данных основана на формате XML. Он является одним из наиболее распространенных форматов (в том числе при экспорте/импорте данных), легко генерируется и анализируется средствами C#, имеет понятную человеку теговую структуру, отлично подходит для хранения данных ролевой и других моделей.

5.3. Архитектура

5.3.1. Архитектура приложения

Сначала опишем архитектуру разрабатываемого инструмента в целом (см. Рис.13).

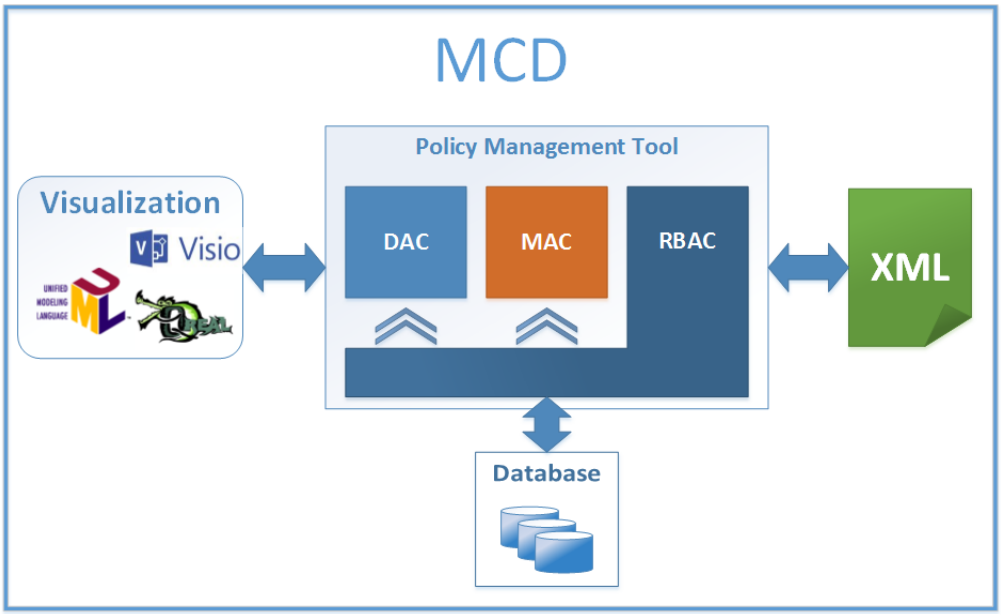


Рис.13. Архитектура PMTool

[19] http://msdn.microsoft.com/en-us/library/bb425822.aspx
[20] http://www.visualstudio.com/
[21] http://www.microsoft.com/ru-ru/server-cloud/products/sql-server/default.aspx#fbid=lXWjiX1F7LK
[22] http://office.microsoft.com/ru-ru/visio/

В основе PMTool лежит база данных MS SQL, которая содержит основные элементы модели $RBAC_3$ и другую необходимую информацию (ее архитектура будет рассмотрена ниже). С базой данных взаимодействует основной модуль приложения, реализующий функциональность ролевой модели. В результате различных управляющих воздействий содержимое БД изменяется соответствующим образом (добавляются и/или удаляются роли, пользователи, права…), однако данные должны остаться в консистентном состоянии. В противном случае под угрозу будет поставлена стабильность работы подсистемы контроля доступа и, как следствие, безопасность системы в целом. На основе RBAC-модуля строятся модули DAC и MAC, которые переводят элементы этих моделей в «ролевые» термины.

Среди дополнительных возможностей PMTool выделяются в первую очередь Визуализация и Экспорт/Импорт политик. Т.о. существует модуль, отвечающий за графическое представление политик в виде схем/диаграмм с использованием возможностей MS Visio. Не исключается возможность интеграции с другими средствами визуализации и инструментами визуального моделирования. За экспорт/импорт политик отвечает модуль XMLManager, способный сохранить RBAC-данные из базы в файл XML с определенной структурой (описывается далее), а также заполнить БД новой информацией.

5.3.2. Архитектура базы данных

Опишем архитектуру базы данных (см. приложение 8). Набор таблиц базы данных почти идентичен элементам из определения $RBAC_3$ (т.е. $RBAC_0$+$RBAC_1$+$RBAC_2$).

Сначала переведем $RBAC_0$ в БД. Так, ключевыми являются таблицы *User, Role, Permission, Session* (см. пункт 1 определения $RBAC_0$). Они обязательно содержат идентификационные поля *Id* и *Name*, а также некоторые специфические детали. Необходимо также ввести таблицы *Action* (все возможные действия в системе) и *Object* (все контролируемые объекты) для использования в привилегиях.

Привилегия – это, по сути, именованная пара {объект, набор действий}. Нельзя хранить списки значений в одной ячейке реляционной БД (1 Нормальная Форма БД), следовательно, существует таблица *PermissionPerObject*, по отдельности сопоставляющая действия объектам в виде пар {*Action_ID, Object_ID*}. Следует понимать, что одна привилегия может касаться нескольких объектов и действий с ними, поэтому *Permission.ID* служит третьим столбцом в сводной таблице *PermissionPerObject*, а остальные детали остаются в т. *Permission* (суть 2-ая Нормальная Форма).

Отношения многие-ко-многим Permission Assignment (PA) и User Assignment (UA) (см. пункты 2 и 3 определения $RBAC_0$) представлены сводными таблицами *RolePermission* и *AuthUserRole*, состоящими из двух полей: {*Role_ID,Permission_ID*} и {*Role_ID,User_ID*} соответственно.

Функция $user: S \to U$, возвращающая пользователя конкретной сессии, представляется дополнительным столбцом *User_ID* в таблице *Session*.

Функция $roles: S \to 2^R$, возвращающая набор активированных ролей в рамках конкретной сессии, определяется с помощью сводной таблицы *ActiveRole*, сопоставляющей *Role_ID Session_ID*.

$RBAC_1$ (иерархии ролей) легко воплощается в виде таблицы RoleHierarchy, хранящей пары {*SeniorRole_ID,JuniorRole_ID*}. Т.о. осуществляется хранение ролевого графа (ориентированного графа без петель) в виде пар вершин, определенных заданным частичным отношением порядка «is-junior» (см. п.4.7.2).

Реализация $RBAC_2$ может обладать рядом нетривиальных особенностей в зависимости от вида ограничений. Классическое разделение обязанностей (SoD) описывается таблицей, содержащей все пары ролей (как «прямые», так и «обратные»[23]). Свойство cardinality подразумевает добавление в таблицу *Role* соответствующего поля, хранящего целое число – количество допустимых пользователей в конкретной роли (значение «-1» – бесконечность). Моделирование более сложных ограничений (например, представляющих собой сложную функцию, зависящую от многих элементов RBAC) может не укладываться в БД и потребовать реализации дополнительных механизмов.

PMTool должен уметь управлять несколькими политиками, поэтому введена еще одна таблица *Policy* с их описанием. Чтобы в разных политиках можно было создавать одноименные элементы ролевой модели, в таблицы *User*, *Role*, *Permission* было добавлено поле *Policy_ID*. В текущей реализации считается, что элементы таблиц *Action* и *Object* являются общими для всей системы и доступными для всех политик, поэтому к ним не добавляется идентификатор *Policy_ID*.

В некоторых таблицах были искусственно введены индексы «ID» для увеличения производительности БД.

5.4. Особенности реализации

5.4.1. Реализация базовой функциональности RBAC и поддержка согласованности БД

Ключевой особенностью реализации управляющей функциональности для ролевой модели можно считать поддержку консистентного состояния БД. Существует набор правил, соблюдение которых гарантирует такую корректность. [16] Из них также вытекают пред- и постусловия операций модификации состояния системы безопасности. В простом случае этих правил 19, и они касаются естественных формализованных ограничений, накладываемых на элементы RBAC-модели. К примеру, ограничение cardinality может быть записано так:

$$\forall r \in Roles, \left|authorized_{users(r)}\right| \leq cardinality(r)$$

Для прототипа было принято решение реализовать основную функциональность для $RBAC_3$, включающую в себя полную поддержку элементов $RBAC_0$ и $RBAC_2$ (Static

[23] Если роли R1 и R2 являются взаимоисключающими, то в таблице SoD будут присутствовать две записи: {R1,R2} и {R2,R1}

Separation of Duties, Dynamic Separation of Duties, Cardinality), но без Ролевых Иерархий $RBAC_1$.

Основные действия: **add**xxx – добавить элемент xxx, **rm**xxx – удалить элемент xxx, **udpate**xxx – обновить данные элемента xxx (своего рода модификация **add**xxx).

При реализации действий по добавлению элементов зачастую не требуется проведение сложных проверок условий в целях сохранения консистентности, кроме предотвращения создания дублирующихся сущностей. Единственное, создание привилегии Permission влечет за собой проверку существования (или добавление) соответствующих пар {объект, действие} в таблице PermissionPerObject, т.к. одно лишь название привилегии не может существовать.

В свою очередь реализация корректных операций удаления требует бóльших затрат. В текущей версии прототипа реализованы каскадные удаления сущностей с соблюдением правил [16], обеспечивающих консистентность БД.

К примеру, при удалении пользователя (rmUser) необходимо сначала удалить активные роли (т.*ActiveRole*), потом все его сессии (т.*Session*), затем авторизованные роли (т.*AuthUserRole*) и, в конце концов, самого пользователя (т.*User*). Порядок операций в каскадном режиме чрезвычайно важен, т.к. иначе будут оставаться «висячие» ссылки на объекты, которые уже были удалены. На самом деле, если везде правильно прописаны внешние и первичные ключи таблиц, то СУБД не даст нам сделать подобное и выдаст ошибку, из-за чего работа приложения будет нарушена.

Удаление ролей происходит аналогично: деактивация роли у работающих пользователей (т.*ActiveRole*), удаление из списков авторизованных (т.*AuthUserRole*), удаление привилегий, записей в списках SoD (т.*RolePermission*, т.*SSOD/DSOD*) и самой роли (т.*Role*).

Удаление привилегий также требует каскадной ликвидации всех связанных сущностей: уничтожение требуемых пар {объект, действие} (т.*PermissionPerObject*), соответствующих сущностей {роль, привилегия} (т.*RolePermission*) и непосредственно самой привилегии (т. *Permission*).

Удаление политик – самая большая операция. Здесь «каскадно» удаляются абсолютно все записи таблиц с заданным *Policy_ID*.

Аналогично реализуются и остальные операции. Т.о. описана в некотором смысле функциональная спецификация PMTool.

В [16] рассматривается теорема о согласованности RBAC базы данных. Необходимым и достаточным условием для этого является выполнение 13 перечисленных правил, касающихся таких свойств ролевой модели, как списки SoD, ролевые иерархии, авторизация и активация ролей. Можно доказать, что реализованная на текущий момент функциональность PMTool сохраняет консистентность БД. Проведя аналогию с многомерным пространством, заметим, что введенное понятие Политики увеличило

размерность пространства RBAC на единицу. При конкретных значениях *Policy* наша система согласована, т.к. является в точности такой, о которой говорит теорема, и требуемые условия выполнены. Политики независимы между собой, а функциональность в силу реализации замкнута относительно конкретного *Policy_ID*.

Стоит отметить, что в случае наличия ролевых иерархий важно следить за всем операциями более пристально. При присвоении пользователю набора ролей рекомендуется следить за тем, чтобы в структурах RBAC были отражены только главные из них (учитывая заданную иерархию), т.к. остальные авторизуются автоматически в силу правил $RBAC_1$. В набор активных ролей вносится любое подмножество авторизованных (в том числе иерархически младших). [16] [27].

Рекомендуется оперировать с ролями, привилегиями и ролевыми иерархиями только тогда, когда эти модификации не коснутся ни одного активного пользователя. В противном случае может быть нарушена безопасность системы. К примеру, пользователь U активировал набор ролей, получил соответствующие привилегии с учетом текущего состояния иерархии. Администратор A вдруг решил пересмотреть политику и удалил/изменил разрешения, иерархии или роли, среди которых были и активные для U. Теперь «статическая» часть системы может все так же находиться в консистентном состоянии, однако пользователь U в течение своей сессии может иметь старые привилегии. Для предотвращения таких ситуаций необходимо наладить глобальные механизмы сообщений, либо вовсе не рисковать, а проводить подобные административные изменения во время пауз для сервисного обслуживания.

5.4.2. Работа с базой данных

В продолжение разговора о базе данных стоит сказать, что для взаимодействия с ней были созданы классы LINQtoSQL, позволяющие в ОО стиле создавать и обрабатывать разнообразные запросы.

Во время выполнения административных функций может сложиться так, что применение итоговых изменений не потребуется, либо в середине составной операции произошел сбой (пропало соединение с БД)… В таких случаях необходимо корректно откатить все изменения и восстановить состояние БД. Для этих целей реализован механизм транзакций. Отчасти его возможности доступны средствами класса `DataContext`: метод `SubmitChanges()`. Однако если транзакция состоит из нескольких блоков, которые по тем или иным причинам не удалось обработать вызовом этого метода, то можно использовать класс `TransactionScope`[24], который создает транзакцию и применяет ее при вызове метода `Complete()`.

Также при работе с базой данных можно пользоваться подкачкой данных для повышения производительности, если заведомо известно, какая информация вскоре потребуется.

[24] http://msdn.microsoft.com/ru-ru/library/system.transactions.transactionscope.aspx

Чтобы при выполнении запроса произошла подгрузка, необходимо заранее определить требуемые данные с помощью класса `DataLoadOptions`[25].

При работе с базой данных удобно представлять ее данные в виде таблиц. Таким образом, был выбран один из вариантов визуализации элементов модели в виде сетки `DataGridView`[26] (см. Рис.14). Во-первых, это отличный наглядный способ отладки заполнения базы данных, который позволит не обращаться постоянно к средствам и инструментам SQL Server, а отслеживать изменения прямо в приложении. Плюс ко всему, это один из наиболее распространенных способов визуализации отношений БД. При реализации были использованы привязки элементов `DataGridView` к данным из RBAC-базы с помощью `BindingSource`[27].

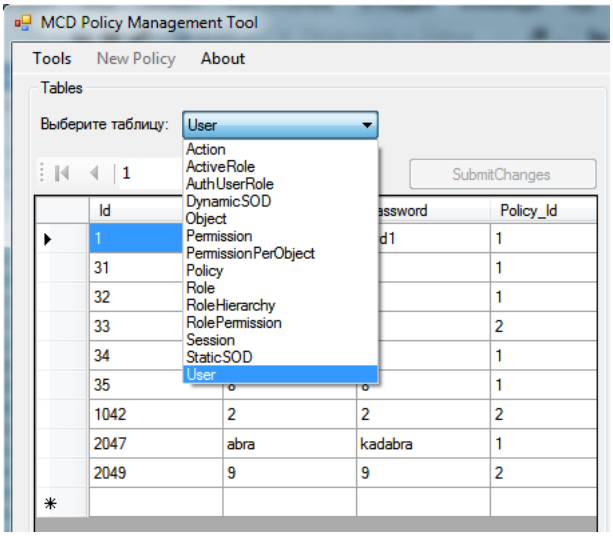


Рис.14. Табличное представление RBAC БД

5.4.3. Визуализация

Помимо табличного представления (см. Рис.14) для более полного восприятия данных, как уже упоминалось, было принято решение реализовать модуль визуализации на основе MS Visio. Его задача – отрисовка отношений RBAC БД в виде определенных диаграмм и схем, в том числе и транзитивных зависимостей, например, рассмотренные ниже отношения Users и User-Role-Permission.

Работа с Visio обладает рядом особенностей[28]. Для реализации встроенной в приложение функциональности необходимо сначала активировать соответствующую COM компоненту (MS Visio Drawing Control). После этого в списке доступных элементов Visual Studio можно обнаружить нужный нам Drawing Control – объект, представляющий, по сути, мини-Visio. При создании диаграмм разработчик оперирует несколькими основными классами: Visio.Application, Visio.Page, Visio.Document, Visio.Window, Visio.Shape – необходимыми для доступа к соответствующим объектам.

Существует несколько типов документов Visio:

- .vsdx (Visio drawing)
- .vsdm (Visio macro-enabled drawing)
- .vssx (Visio stencil)
- .vssm (Visio macro-enabled stencil)
- .vstx (Visio template)
- .vstm (Visio macro-enabled template)

[25] http://msdn.microsoft.com/ru-ru/library/system.data.linq.dataloadoptions(v=vs.110).aspx
[26] http://msdn.microsoft.com/ru-ru/library/system.windows.forms.datagridview.aspx
[27] http://msdn.microsoft.com/ru-ru/library/system.windows.forms.bindingsource.aspx
[28] Многие тонкости работы с Visio превосходно описаны в блоге http://www.mikeborozdin.com/

Нам понадобятся в простом случае файлы-трафареты (stencil .vssx). Эти файлы могут быть созданы вручную и состоят из конкретных «шаблонов» (Master) элементов Visio диаграмм. Существуют стандартные файлы-трафареты для базовых схем Visio. Во время работы с документом (.vsd) можно подключать несколько трафаретов одновременно. Для создания фигуры на странице (в обычном ручном режиме) необходимо перетащить соответствующий Master на рабочую поверхность, что повлечет за собой вызов метода dropShape и отрисовку этой фигуры. В автоматическом режиме достаточно вызывать метод dropShape с нужными параметрами.

В данной работе были созданы трафареты для UML Use Case и UML Class диаграмм с определенными параметрами: от цвета и формы элементов до шрифта[29]. При этом использовался режим разработчика в среде Visio,позволяющий увидеть все скрытые от обычного пользователя свойства фигур.

К примеру, для UML Class были определены Class, Member, Separator:

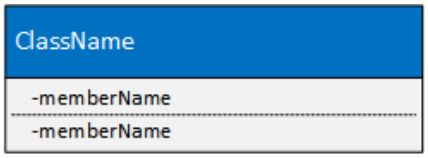


Рис.16. Class Master

Рис.15. Separator Master

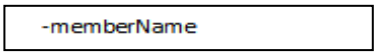


Рис.17. Member Master

Для UML Use Case определены Actor, Use Case, Subsystem Container, Association Masters.

Рис.20. Actor Master

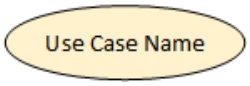


Рис.21. Use Case Master

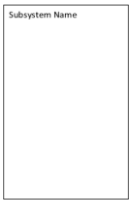


Рис.19. Subsystem Master

Рис.18. Association Master

Визуализация отношения Users

Отношение Users по запросу пользователя отображает сущности таблицы Users в формате диаграммы классов/объектов UML (аналогично табличному представлению DataGridView, описанному ранее). Соответствующая функция VisualizeUsers открывает трафарет UML_Class, создает элемент типа «Контейнер» из Class Master. Производится запрос к БД, а все полученные сущности обрабатываются и переводятся в элементы контейнера Member Master с текстом в формате `<Name|Password|Policy>`. Каждый элемент отделяется от следующего специальным Separator Master. В итоге получается (см. Рис.22):

[29] http://office.microsoft.com/en-us/visio-help/create-save-and-share-a-custom-stencil-HA101782586.aspx

Визуализация отношения User-Role-Permission

Было введено отношение URP (User-Role-Permission) – одно из самых информативных отношений модели RBAC, показывающее, какие роли авторизованы пользователю и какие привилегии он, таким образом, может получить от них. С учетом аналогии RBAC модели и должностной структуры некоторой организации представляется разумным визуализировать URP в виде диаграммы UML Use Case. Т.е. пользователь получает несколько «должностей» и, таким образом, располагает определенными правами.

User Table
name: name1|password: pwd1|policy: 1
name: 4|password: 4|policy: 1
name: 5|password: 5|policy: 1
name: 6|password: 6|policy: 2
name: 7|password: 7|policy: 1
name: 8|password: 8|policy: 1
name: 2|password: 2|policy: 2
name: abra|password: kadabra|policy: 1
name: 9|password: 9|policy: 2

Рис.22. Схема Users

Функция VisualizeURP открывает трафарет UML_Use_Case.vssx и создает контейнер Subsystem для последующей отрисовки доступных ролей. Также создается конкретный пользователь из мастера Actor. Роли, полученные в результате запроса к БД, являются конкретными UseCase элементами и отображаются внутри контейнера.

Затем активируется трафарет Basic_U.vssx, представляющий собой стандартный файл Visio с фигурами общего назначения (многоугольники, окружности, простейшие линии и т.п.). Из него выбираются прямоугольники для создания привилегий на схеме. Выполняется еще один запрос к БД, возвращающий наборы прав доступа для авторизованных ролей. Полученные данные заполняют текстовое поле внутри прямоугольников и рисуются на диаграмме. Размеры объектов вычисляются с учетом параметров текста внутри них. Созданные только что привилегии необходимо «привязать» к соответствующим ролям без пересечения их фигур. Связь производится с помощью динамического объекта «Assosiation», начало которого задается в элементе-роли, а конец прикрепляется к соответствующему блоку привилегий.

В результате получается диаграмма URP для конкретного пользователя (см. Рис.23):

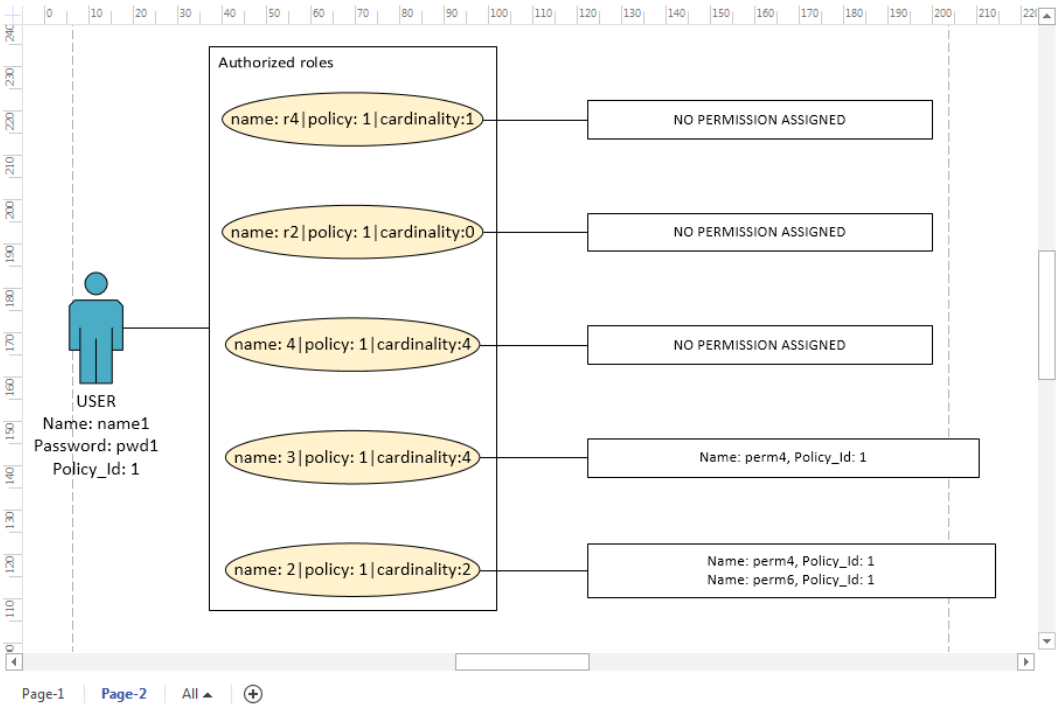


Рис.23. Схема URP

5.4.4. Экспорт/Импорт данных в XML

Возможность экспорта и импорта политик играет немаловажную роль при переносе настроек из одной системы в другую. К примеру, в системе MCD может быть очень востребована функциональность, позволяющая перенести данные моделей контроля доступа из старых систем (например, Windows или Linux машин) в новый формат, основанный на RBAC. Например, рассмотрим дискреционное разделение доступа и, соответственно, списки контроля доступа в современных ОС. В Windows можно получить результат системных утилит, таких как icacls[30], get-acl[31] и др., и на его основе импортировать смоделированную DAC политику в терминах ролевой модели. Аналогично для Linux с помощью команд getfacl[32], ls…

В данной работе разработан модуль, выполняющий экспорт/импорт политик RBAC-модели в формате XML. Первое, с чем предстояло определиться – удобная для автоматизированной обработки структура, которая при этом была бы понятна человеку. Была выбрана следующая схема.

Тэг самого верхнего уровня описывает файл в целом. Далее хранятся блоки `POLICIES`, `ACTIONS`, `OBJECTS` (напомним, что `Actions` и `Objects` являются общими для системы в целом). Блоки `ACTIONS` и `OBJECTS` содержат перечисление сущностей `Action` и `Object` соответственно, в атрибутах которых записаны все их данные из БД. На третьем уровне хранятся блоки конкретных политик `Policy`. В каждом из них есть непосредственные атрибуты политики, а также дочерние блоки `USERS`, `ROLES`, `PERMISSIONS`, `SSOD`, `DSOD`, `RoleHierarchy`. Для каждого пользователя в блоке `USERS` создается дочерний элемент с его атрибутами и списком авторизованных ролей `AuthRoles`. Аналогично, для ролей определяются элементы `Role` в `ROLES`, обладающие собственными атрибутами и списком привилегий `Permissions`. Блок `PERMISSIONS` состоит из сущностей типа `Permission`, каждая из которых имеет название и дочерний список `PermissionsPerObject` (пары *{`action`,`object`}*, как упоминалось ранее). Списки `SSOD`, `DSOD` «распадаются» на элементы `Role`, внутри которых есть набор `ExclusiveRole`. Т.о. для каждой роли в одном блоке хранятся все ее взаимоисключающие. Аналогичным образом построена структура блока `RoleHierarchy`: это пары *{`SeniorRole`,список `JuniorRole`}*.

Пример неполного конфигурационного файла приведен в приложении 10.

При реализации функций экспорта/импорта существует несколько возможностей для чтения и записи файлов XML. Для записи можно пользоваться, например, классами XmlWriter (более абстрактный), XmlTextWriter (имплементация предыдущего), XmlDocument (оперирует с файлами с точностью до сущностей, напоминает ОО подход). Аналогично и для чтения: XmlReader, XmlTextReader, XmlDocument. В силу предпочтения автора и удобства реализации выбор пал на XmlWriter (запись) и XmlDocument (чтение).

[30] icacls – http://technet.microsoft.com/ru-ru/library/cc753525(v=ws.10).aspx
[31] get-ACL – http://technet.microsoft.com/en-us/library/hh849802.aspx
[32] getfacl – https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/acls-retrieving.html

Отметим очевидную, но очень важную деталь: при импорте политик из XML файла необходимо внимательно следить за порядком введения в систему новых сущностей. Сначала в базу данных добавляются ни с чем не связанные объекты, после чего уже можно строить отношения, использующие их идентификаторы. Таким образом, применительно к описываемой задаче, последовательность разбора импортируемого файла такова:

1. Object, Actions
2. Policies
   2.1. Permissions
   2.2. Roles, RolePermissions
   2.3. Users, UserRoles
   2.4. SSOD, DSOD, RoleHierarchy

Заключение

В рамках данной дипломной работы была доказана возможность построения системы контроля доступа на базе ролевой модели (RBAC), средствами которой могут быть выражены основные политики безопасности (MAC и DAC).

1. Проведен обзор нормативных документов, касающихся информационной безопасности
2. Исследованы существующие модели контроля доступа, их взаимосвязи и возможности выражения одной через другую
3. Был реализован прототип средства администрирования политик безопасности, обладающий базовой функциональностью по управлению ролевой моделью и возможностью визуализации некоторых ее отношений.

Данная работа была представлена (отдельно и в рамках проекта MCD) на конференциях и семинарах:

- Семинар кафедры системного программирования Мат-Мех СПбГУ
- Конференция СПИСОК-2014 [49]
- Научно-техническая конференция студентов «СТУДЕНЧЕСКАЯ НАУЧНАЯ ВЕСНА» [48]
- 3-я международная науч.-техн. конференция «Аэрокосмические технологии», посвященная 100-летию со дня рождения ак. В.Н.Челомея, 20-21 мая 2014 г.[33]

Дальнейшее развитие

Среди ближайших планов можно выделить несколько основных направлений. В первую очередь необходимо реализовать расширенную функциональность, обеспечивающую полноценное управление моделью $RBAC_3$. Также важную роль играет исследование подходов к заданию разнообразных ограничений в модели $RBAC_2$. Помимо этого планируется расширение возможностей визуализации и визуальное моделирование (управление) политиками. Возможна работа не только с Visio, но и интеграция с другими технологиями, например, QReal. Еще одним важным пунктом является создание шаблонов для моделирования дискреционных и мандатных моделей в терминах RBAC. Пристальное внимание также требуется уделить уже упоминавшимся подходам к контролю состояния системы с точки зрения безопасности и проверке неразрешимостей. Итоговой целью, естественно, является интеграция в систему MCD.

[33] http://hoster.bmstu.ru/~mntkakt2021/index.html

Список литературы

[1] **A comparison of commercial and military computer security policies** / авт. David D. Clark David R. Wilson. - 1987 г..

[2] **A Linear Time Algorithm for Deciding Subject Security** [Журнал] / авт. Lipton R.J. Snyder L. // Journal of the Association for Computing Machinery. - 1977 г.. - 3 : Т. 24.

[3] **Access Control Basics** [В Интернете] / авт. Kantarcioglu Murat. - UT Dallas, 2009 г.. - http://www.utdallas.edu/~muratk/courses/dbsec09s_files/access1.pdf.

[4] **Access Control Fundamentals, part 3** [В Интернете] / авт. Sadeghi Cubaleska. - 2009 г.. - http://www.trust.rub.de/media/ei/lehrmaterialien/232/OSS_chap3.pdf.

[5] **Access Control Models. Part I** [В Интернете] / авт. Kantarcioglu Murat. - UT Dallas, 2009 г.. - http://www.utdallas.edu/~muratk/courses/dbsec09s_files/access2.pdf.

[6] **Access Control Models. Part II** [В Интернете] / авт. Kantarcioglu Murat. - UT Dallas, 2009 г.. - http://www.utdallas.edu/~muratk/courses/dbsec09s_files/access2.pdf.

[7] **Access rights administration in Role-Based Security Systems** [Журнал] / авт. Matunda Nyanchama Sylvia Osborn. - [б.м.] : The dep. of CS, The University of Western Ontario, 1996 г..

[8] **Advances in Computers** [Книга] / авт. Marshal Yovits C.. - Indianapolis : Academic Press, 1985. - Т. 24.

[9] **Certified Products** [В Интернете] // Common Criteria. - http://www.commoncriteriaportal.org/products/.

[10] **Common Criteria for Information Technology Security Evaluation. Part I: Introduction and general model** [В Интернете] // Common Criteria Portal. - 2012 г.. - http://www.commoncriteriaportal.org.

[11] **Common Criteria for Information Technology Security Evaluation. Part II: Security functional components** [В Интернете] // Common Criteria Portal. - 2012 г.. - http://www.commoncriteriaportal.org/cc/.

[12] **Common Criteria for Information Technology Security Evaluation. Part III: Security Assurance Components** [В Интернете] // Common Criteria Portal. - 2012 г.. - http://www.commoncriteriaportal.org/cc/.

[13] **Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies** [Журнал] / авт. SYLVIA OSBORN RAVI SANDHU, QAMAR MUNAWER. - 2000 г..

[14] **Cost of Data Breach Study** / авт. Ponemon Institute. - 2013 г..

[15] **DoD Standard 5200.28-STD "The orange book"** / авт. United States Department of Defense. - 1985 г..

[16] **Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management** [Журнал] / авт. Serban I. Gavrila John F. Barkley. - 1998 г..

[17] **Hierarchical protection systems** [Журнал] / авт. Margaret Wu S.. - Iowa City : [б.н.], 1981 г..

[18] **Hierarchical Take-Grant Protection Systems** [Журнал] / авт. Matt Bishop.

[19] **How to do Discretionary Access Control Using Roles** [Журнал] / авт. Ravi Sandhu Qamar Munawer. - 1998 г..

[20] **Integrity considerations for secure computer systems**. - Hanscom Air Force Base, Bedford, Massachusetts : DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS, ELECTRONIC SYSTEMS DIVISION, AIR FORCE SYSTEMS COMMAND, UNITED STATES AIR FORCE, 1977 г..

[21] **Mandatory Access Control** / авт. Hakan Lindqvist // Master's Thesis in Computer Science. - 2006 г..

[22] **Methods for Access Control: Advances and Limitations** / авт. Ausanka-Crues Ryan. - Claremont, California : [б.н.].

[23] **Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems** [Журнал] / авт. Kuhn Richard. - 1997 г..

[24] **Parts of the Access Control Model** [В Интернете] / авт. Microsoft. - http://msdn.microsoft.com/en-us/library/windows/desktop/aa374862(v=vs.85).aspx.

[25] **RBAC on MLS Systems without Kernel Changes** [Журнал] / авт. Kuhn D. Richard. - 1998 г..

[26] **Role Hierarchies and Constraints for Lattice-Based Access Controls** [Журнал] / авт. Sandhu Ravi. - [б.м.] : George Mason University & SETA Corporation, 1996 г..

[27] **Role-Based Access Control (RBAC): Features and Motivations** / авт. David F. Ferraiolo Janet A. Cugini, D. Richard Kuhn. - [б.м.] : National Institute of Standards and Technology, 1995 г..

[28] **Role-Based Access Control Models** [Журнал] / авт. Ravi S. Sandhu Edward J. Coyne, Hal L.Feinstein, Charles E. Youman. - 1995 г..

[29] **Role-Based Access Controls** / авт. David F. Ferraiolo D. Richard Kuhn. - [б.м.] : National Institute of Standards and Technology, 1992 г..

[30] **Secure Computer Systems: Mathematical Foundations** / авт. D.Elliott Bell Leonard J. LaPadula. - [б.м.] : MITRE, 1973 г..

[31] **Security Analysis in Role-Based Access Control** [Журнал] / авт. Li Ninghui Tripunitara V. Mahesh. - 2004 г..

[32] **Security in Computing** [Книга] / авт. Pfleeger Charles P.. - [б.м.] : Prentice Hall, 2006.

[33] **Security Models** [Журнал] / авт. McLean John. - [б.м.] : Wiley Press, 1994 г.. - Encyclopedia of Software Engineering.

[34] **The Chinese Wall Security Policy** [Журнал] / авт. David F.C. Brewer Michael J.Nash. - 1989 г..

[35] **Theft of information on the Take-Grant protection model** [Доклад] / авт. Matt Bishop.

[36] **Visio 2013 SDK** [В Интернете] / авт. Microsoft. - http://www.microsoft.com/en-us/download/details.aspx?id=36825.

[37] **Закон о государственной тайне**. - 1993 г..

[38] **Защита и обработка конфиденциальных документов** [Книга] / авт. Шрамкова И.Г. Крат Ю.Г.. - Хабаровск : Издательство ДВГУПС, 2008.

[39] **Курсовая работа "Безопасное рабочее пространство пользователя MCD. Конфигурация системы. Построение протокола взаимодействия компонентов"** [Доклад] / авт. С.А. Серко. - СПб : СПбГУ, 2013.

[40] **Курсовая работа "Безопасное рабочее пространство пользователя Multi-Cloud Desktop. Модуль внедрения автозапуска приложений в виртуальные машины"** [Доклад] / авт. Р.С. Одеров. - СПб : СПбГУ, 2013.

[41] **Основы информационной безопасности** [В Интернете] / авт. Владимир Галатенко // ИНТУИТ. - 2003 г.. - http://www.intuit.ru/studies/courses/10/10/info.

[42] **Перечень сведений, отнесенных к государственной тайне** [В Интернете] / авт. Российская Газета // rg.ru. - 2006 г.. - http://img.rg.ru/pril/9/90/00/gostajna.pdf.

[43] **Постановление №870 "Об утверждении Правил отнесения сведений, составляющих государственную тайну, к различным степеням секретности"**. - 1995 г..

[44] **Руководящий документ "Защита от несанкционированного доступа к информации. Часть 1. ПО средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей".**. - 1999 г.. - Т. Часть I.

[45] **Руководящий документ "Классификация автоматизированных систем и требования по защите информации"** / авт. ГТК РФ.

[46] **Руководящий документ "Концепция защиты средств вычислительной техники и автоматизированных систем от несанкционированного доступа к информации"** / авт. ГТК при Президенте РФ. - 1992 г..

[47] **Стандарты информационной безопасности** [В Интернете] / авт. Владимир Галатенко // ИНТУИТ. - 2003 г.. - http://www.intuit.ru/studies/courses/30/30/info.

[48] **Технология одновременной безопасной обработки данных разных уровней секретности** [Конференция] / авт. Малыгин А.О. Одеров Р.С., Серко С.А. // Научно-техническая конференция студентов «СТУДЕНЧЕСКАЯ НАУЧНАЯ ВЕСНА», МГТУ им. Н.Э. Баумана . - Реутов : [б.н.], 2014.

[49] **Управление политиками контроля доступа на основе ролевой модели** [Конференция] / авт. Р.С. Одеров // СПИСОК-2014. - Санкт-Петербург : СПбГУ, 2014.

[50] **Федеральный закон о безопасности**. - 2010 г..

Приложения

Приложение 1. Классификация АС по требованиям к защите информации

Подсистемы и требования	Классы								
	3 Б	**3 А**	**2 Б**	**2 А**	**1 Д**	**1 Г**	**1 В**	**1 Б**	**1 А**
1. **Подсистема управления доступом** 1.1. *Идентификация*, проверка подлинности и контроль доступа субъектов: в систему;	+	+	+	+	+	+	+	+	+
к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ;	-	-	-	+	-	+	+	+	+
к программам;	-	-	-	+	-	+	+	+	+
к томам, каталогам, файлам, записям, полям записей.	-	-	-	+	-	+	+	+	+
1.2. Управление потоками информации	-	-	-	+	-	-	+	+	+
2. Подсистема регистрации и учета 2.1. Регистрация и учет: входа/выхода *субъектов доступа* в/из системы (узла сети);	+	+	+	+	+	+	+	+	+
выдачи печатных (графических) выходных документов;	-	+	-	+	-	+	+	+	+
запуска/завершения программ и процессов (заданий, задач);	-	-	-	+	-	+	+	+	+
доступа программ *субъектов доступа* к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ, программам, томам, каталогам, файлам, записям, полям записей;	-	-	-	+	-	+	+	+	+
изменения полномочий *субъектов доступа*;	-	-	-	-	-	-	+	+	+
создаваемых защищаемых объектов доступа.	-	-	-	+	-	-	+	+	+
2.2. Учет носителей информации.	+	+	+	+	+	+	+	+	+
2.3. Очистка (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей.	-	+	-	+	-	+	+	+	+
2.4. Сигнализация попыток нарушения защиты.	-	-	-	-	-	-	+	+	+
3. Криптографическая подсистема 3.1. Шифрование конфиденциальной информации.	-	-	-	+	-	-	-	+	+
3.2. Шифрование информации, принадлежащей различным *субъектам доступа* (группам субъектов) на разных ключах.	-	-	-	-	-	-	-	-	+
3.3. Использование аттестованных (сертифицированных) *криптографических средств*.	-	-	-	+	-	-	-	+	+
4. Подсистема обеспечения целостности 4.1. Обеспечение целостности программных средств и обрабатываемой информации.	+	+	+	+	+	+	+	+	+
4.2. Физическая охрана средств вычислительной техники и носителей информации.	+	+	+	+	+	+	+	+	+
4.3. Наличие администратора (службы защиты) информации в АС.	-	-	-	+	-	-	+	+	+
4.4. Периодическое *тестирование* СЗИ НСД.	+	+	+	+	+	+	+	+	+
4.5. Наличие средств восстановления СЗИ НСД.	+	+	+	+	+	+	+	+	+
4.6. Использование сертифицированных средств защиты.	-	+	-	+	-	-	+	+	+

Приложение 2. Требования к уровню контроля отсутствия недекларированных возможностей

« = » - *то же, что на предыдущем уровне.*

« - » - *нет требований*

« + » - *новые или дополнительные требования*

№	Наименование требования	Уровень контроля			
	Требования к документации	4	3	2	1
1	**Контроль состава и содержания документации**				
1.1.	Спецификация (ГОСТ 19.202-78)	+	=	=	=
1.2.	Описание программы (ГОСТ 19.402-78)	+	=	=	=
1.3.	Описание применения (ГОСТ 19.502-78)	+	=	=	=
1.4.	Пояснительная записка (ГОСТ 19.404-79)	-	+	=	=
1.5.	Тексты программ, входящих в состав ПО (ГОСТ 19.401-78)	+	=	=	=
	Требования к содержанию испытаний				
2.	**Контроль исходного состояния ПО**	+	=	=	=
3.	**Статический анализ исходных текстов программ**	-	-	+	=
3.1.	Контроль полноты и отсутствия избыточности исходных текстов	+	+	+	=
3.2.	Контроль соответствия исходных текстов ПО его объектному (загрузочному) коду	+	=	=	+
3.3.	Контроль связей функциональных объектов по управлению	-	+	=	=
3.4.	Контроль связей функциональных объектов по информации	-	+	=	=
3.5.	Контроль информационных объектов	-	+	=	=
3.6.	Контроль наличия заданных конструкций в исходных текстах	-	-	+	+
3.7.	Формирование перечня маршрутов выполнения функциональных объектов	-	+	+	=
3.8.	Анализ критических маршрутов выполнения функциональных объектов	-	-	+	=
3.9.	Анализ алгоритма работы функциональных объектов на основе блок-схем, диаграмм и т. п., построенных по исходным текстам контролируемого ПО	-	-	+	=
4.	**Динамический анализ исходных текстов программ**				
4.1.	Контроль выполнения функциональных объектов	-	+	+	=
4.2.	Сопоставление фактических маршрутов выполнения функциональных объектов и маршрутов, построенных в процессе проведения статического анализа	-	+	+	=
5.	**Отчетность**	+	+	+	+

Приложение 3. Модель Graham-Denning, операции для работы с матрицей контроля доступа

Command	**Precondition**	**Effect**
Create object o	-	Add column for o in A; place owner in A[x,o]
Create subject s	-	Add row for s in A; place control in A[x,s]
Delete object o	Control in A[x,o]	Delete column o
Delete subject s	Control in A[x,s]	Delete row s
Read access right of s on o	Control in A[x,s] or owner in A[x,o]	Copy A[s,o] to x
Delete access right r of s on o	Control in A[x,s] or owner in A[x,o]	Remove r from A[s,o]
Grant access right r to s on o	Owner in A[x,o]	Add r to A[s,o]
Transfer access right r or r* to s on o	r* in A[x,o]	Add r or r* to A[s,o]

Здесь:

- ***r**** – *право доступа, которое может быть впоследствии передано другому субъекту. Соответственно, право без символа '*****' считается непередаваемым.*
- *Субъект, выполняющий каждую из команд, обозначается '**x**'*

Подробнее см. [32]

Приложение 4. Правила модели Take-Grant:

См. [2], [8]

1. Take

   Пусть x,y,z – вершины графа; дуга $\overrightarrow{xy}$[34] помечена ‘γ’, причем r ∈ γ;[35] дуга $\overrightarrow{yz}$ помечена α ⊆ {r, w, c}. Тогда разрешается добавить дугу с меткой α от x к z (см. Рис.24). Неформально, x берет себе права узла y по отношению к z.

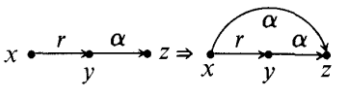


Рис.24. Take

2. Grant

   Пусть x,y,z – вершины графа; дуга $\overrightarrow{xy}$ помечена ‘γ’, причем w ∈ γ; дуга $\overrightarrow{xz}$ помечена α ⊆ {r, w, c}. Тогда разрешается добавить дугу с меткой α от y к z (см. Рис.25). Неформально, x дарит права на узел z узлу y.

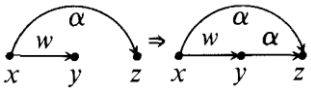


Рис.25. Grant

3. Create

   Пусть x – вершина графа. Тогда разрешается создать новую вершину и дугу, помеченную полным набором привилегий (см. Рис.26). Неформально, x создает новый узел и может выполнять надо ним любое действие.

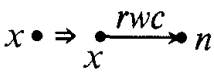


Рис.26. Create

4. Call

   Пусть x,y,z – вершины графа; дуга $\overrightarrow{xz}$ помечена ‘γ’, причем c ∈ γ; дуга $\overrightarrow{xy}$ помечена α ⊆ {r, w, c}. Тогда разрешается добавить новый узел n и две дуги с метками: α для $\overrightarrow{ny}$ и r для $\overrightarrow{nz}$. (см. Рис.27)
   Неформально, x вызывает программу z и передает параметр y. Создается процесс исполнения: n может читать программу z и выполнять действия α с параметрами.

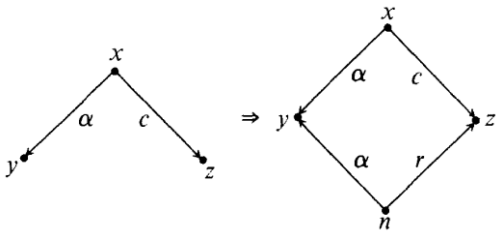


Рис.27. Call

5. Remove

   Пусть x,y – вершины графа; дуга $\overrightarrow{xy}$ помечена α. Тогда разрешается удалить эту дугу. (см. Рис.28) Неформально, x удаляет свои права по отношению к y.

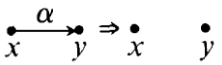


Рис.28. Remove

[34] Запись дуг соответствует направлению: дуга $\overrightarrow{xy}$ идет от вершины x к вершине y

[35] Для краткости не пишется множество меток γ для дуги, а отмечается только конкретная требуемая метка, которую содержит дуга.

Приложение 5. Модель Clark-Wilson, правила.

Здесь C=certification, E=enforcement.

Правило C1. Процедура контроля целостности (IVP) должна контролировать целостность любого элемента из множества CDI.

Правило C2. Процедуры преобразования (TP) не должны нарушать целостность обрабатываемых ими CDI. С каждой TP должен быть связан список элементов CDI, которые допустимо обрабатывать данной процедурой. Такая связь устанавливается администратором безопасности.

e.g. $(\mathrm{TP_i}, \{\mathrm{CDI_a}, \mathrm{CDI_b}, \mathrm{CDI_c}\})$

Правило E1. Система должна контролировать допустимость применения TP к элементам CDI в соответствии со списками, указанными в правиле C2.

Правило E2. Система должна поддерживать список разрешенных конкретным пользователям процедур преобразования с указанием допустимого для каждой TP и данного пользователя набора обрабатываемых элементов CDI.

e.g. $(\mathrm{UserID}, \mathrm{TP_i}, \{\mathrm{CDI_a}, \mathrm{CDI_b}, \mathrm{CDI_c}\})$

Правило C3. Список, определенный правилом C2, должен отвечать требованию разграничения функциональных обязанностей.

Правило E3. Система должна аутентифицировать всех пользователей, пытающихся выполнить какую-либо процедуру преобразования.

Правило C4. Каждая TP должна записывать в журнал регистрации информацию, достаточную для восстановления полной картины каждого применения этой TP.

Журнал регистрации — это специальный элемент CDI, предназначенный только для добавления в него информации.

Правило C5. Любая TP, обрабатывающая элемент UDI, должна выполнять только корректные преобразования этого элемента, в результате которых UDI превращается в CDI.

Правило E4. Только уполномоченное лицо может изменять списки из правил C2 и E2. Это лицо не имеет права выполнять какие-либо действия, если оно уполномочено изменять регламентирующие эти действия списки.

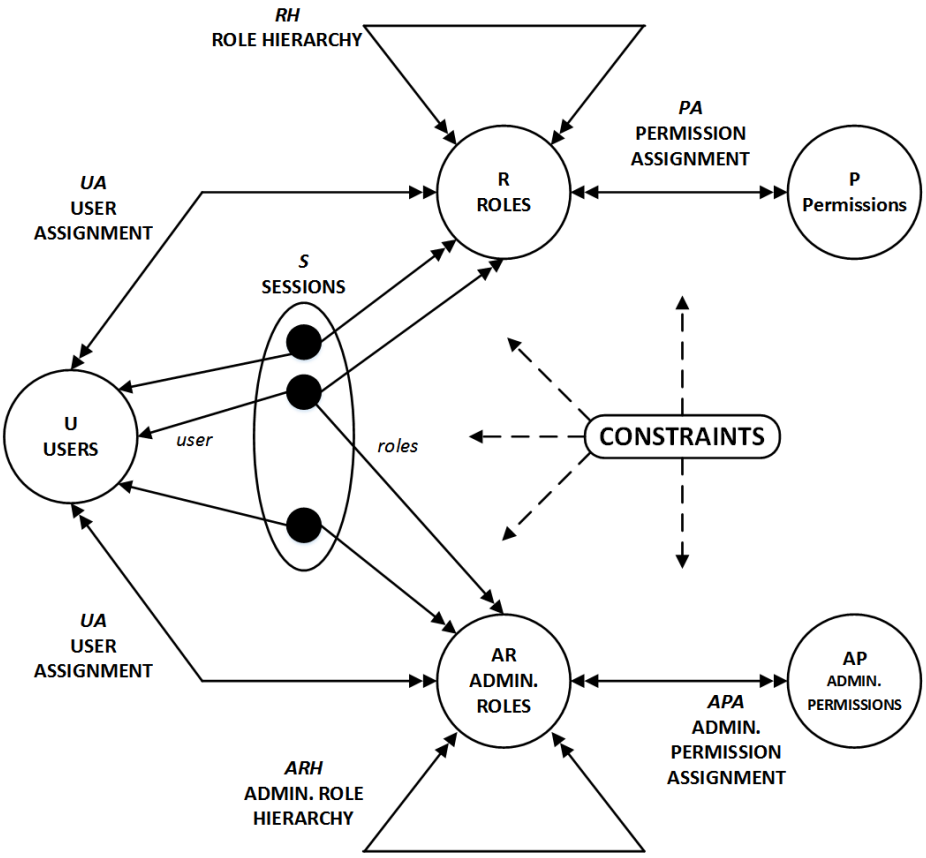
Подробнее см. [1] и

- http://nob.cs.ucdavis.edu/book/book-intro/slides/06.pdf
- http://dic.academic.ru/dic.nsf/ruwiki/1566259

Приложение 6. RBAC и ARBAC модель.

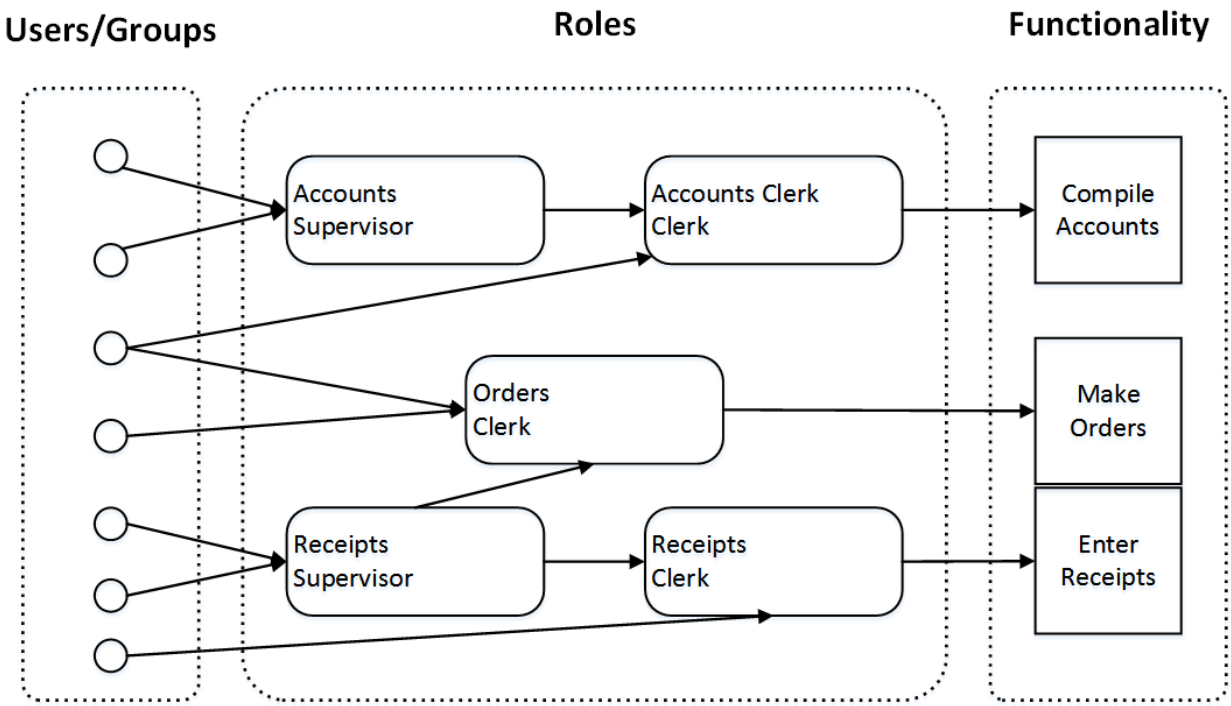
[28]

***Здесь**: двойная стрелка обозначает отношение многие-ко-многим.*

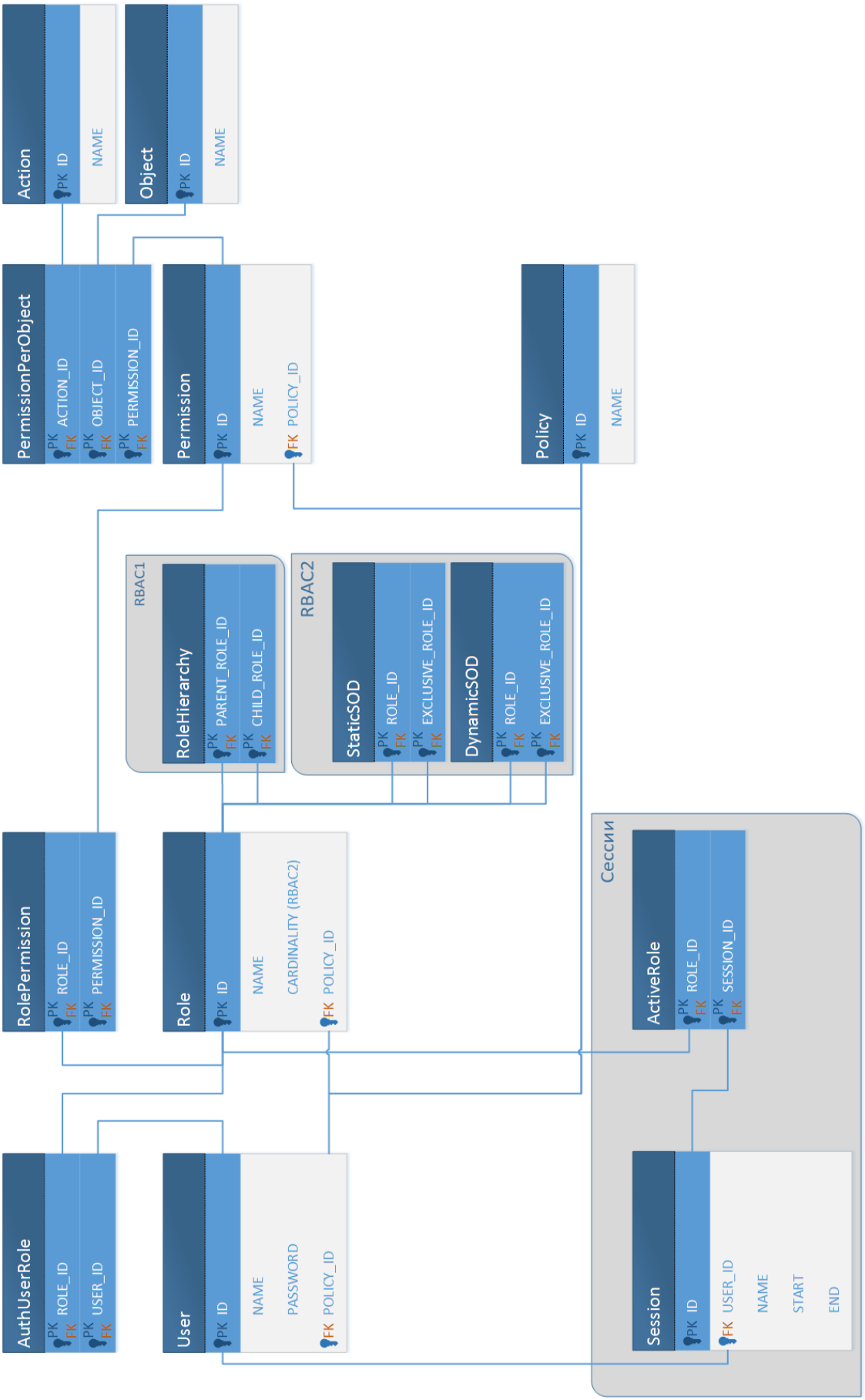


Приложение 7. Граф привилегий (Baldwin's privilege graph)

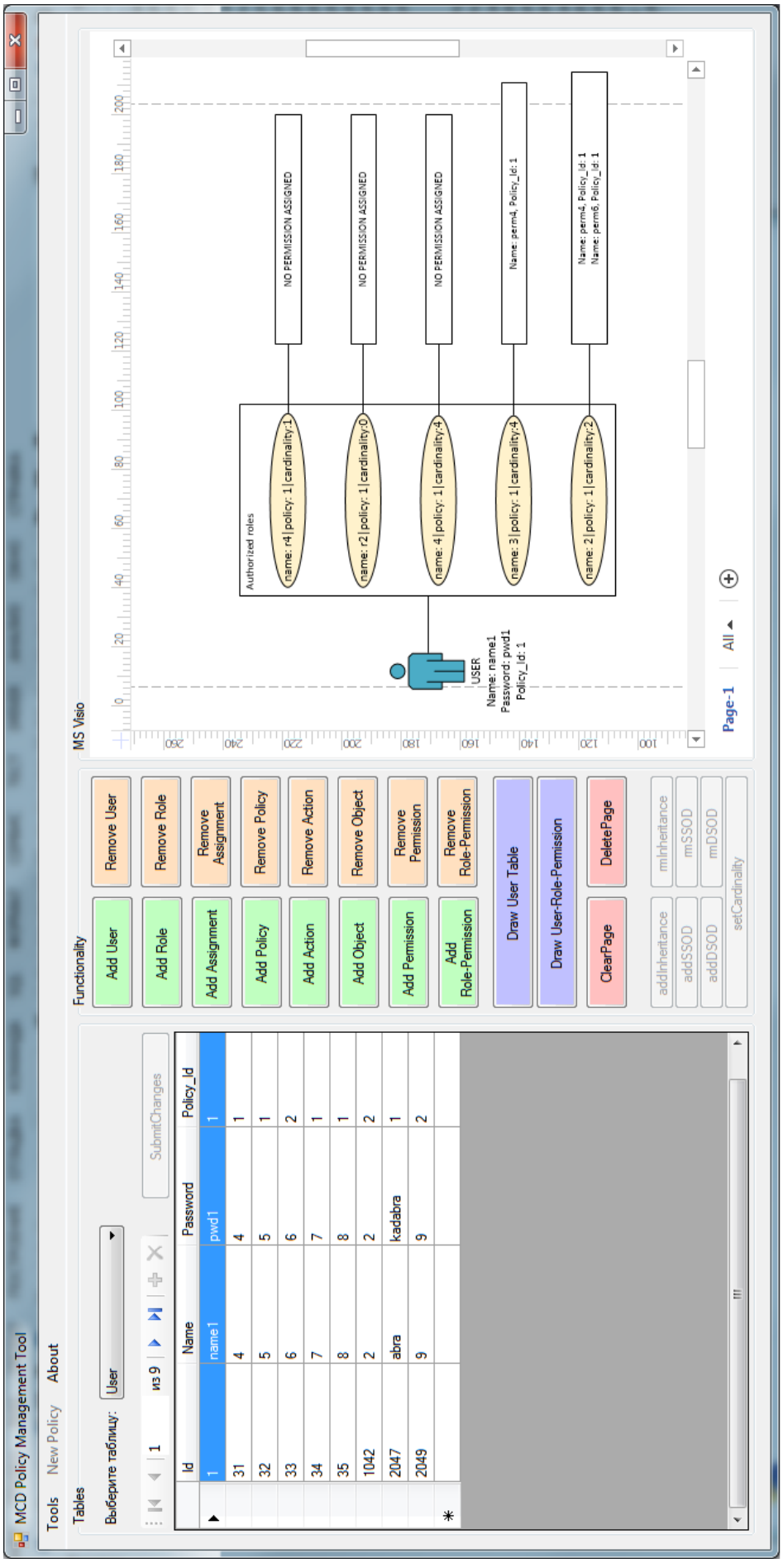
[7]



Приложение 8. Схема базы данных модели RBAC



Приложение 9. Скриншот приложения PMTool



Приложение 10. Структура конфигурационного XML файла

```
<?xml version="1.0" encoding="utf-8"?>
<MCD_PMTOOL_RBAC_DATABASE>
    <POLICIES>
        <Policy name="policy_1">
            <USERS>
                <User name="name1" password="pwd1">
                    <AuthRoles>
                        <Role name="2" />
                    </AuthRoles>
                </User>
            </USERS>
            <ROLES>
                <Role name="2" cardinality="2">
                    <Permissions>
                        <Permission name="perm4" />
                    </Permissions>
                </Role>
            </ROLES>
            <PERMISSIONS>
                <Permission name="perm4">
                    <PermissionsPerObject>
                        <PpO action="read" object="file1" />
                    </PermissionsPerObject>
                </Permission>
            </PERMISSIONS>
            <SSOD>
                <Role name="2">
                    <Exclusive_Role name="3" />
                </Role>
            </SSOD>
            <DSOD>
                <Role name="2">
                    <Exclusive_Role name="3" />
                </Role>
            </DSOD>
            <RoleHierarchy>
                <SeniorRole name="r2">
                    <JuniorRole name="r3" />
                </SeniorRole>
            </RoleHierarchy>
        </Policy>
    </POLICIES>
    <ACTIONS>
        <Action name="read" />
        <Action name="write" />
    </ACTIONS>
    <OBJECTS>
        <Object name="file1" />
    </OBJECTS>
</MCD_PMTOOL_RBAC_DATABASE>
```

Замечание: из приведенного здесь файла вырезаны некоторые блоки, чтобы не перегружать читателя, а лишь дать понимание структуры