ИПМ им.М.В.Келдыша РАН

Абрау-2016 • Труды конференции

С.А. Афонин

Система логического разграничения доступа для облачных информационных систем

Рекомендуемая форма библиографической ссылки

Афонин С.А. Система логического разграничения доступа для облачных информационных систем // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19-24 сентября 2016 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2016. — С. 51-57. — URL: http://keldysh.ru/abrau/2016/17.pdf

Размещена также презентация к докладу

Система логического разграничения доступа для облачных информационных систем

С.А. Афонин

НИИ механики МГУ им. М.В. Ломоносова

Аннотация. Разграничение прав доступа к информации является основой для обеспечения безопасности информационных систем. Классическая ролевая модель ограничения доступа, предусматривающая присвоение привилегий группам пользователей, не позволяет выражать многие правила доступа, возникающие в современных приложениях. В данной работе предлагается модель определения прав доступа, разработанная в предположении, что информационные объекты хранятся в реляционной базе данных и проверка наличия прав производится путём вычисления последовательности SQL запросов. Атрибуты информационных объектов определяются в терминах модели «сущность-связь» и для задания правил, определяющих условия доступа, предлагается использовать высокоуровневую концептуальную модель предметной области. Допускается возможность переопределения некоторых прав доступа в структурных подразделениях организации или клиентами облачного сервиса.

Ключевые слова: информационная безопасность, логическое разграничение доступа, модель «сущность-связь», онтология, SQL

1. Введение

Проблемы, связанные с разграничением доступа в компьютерных системах, изучаются с 60-х годов. За это время было разработано множество подходов, учитывающих различные требования приложений. Значительное распространение получила ролевая модель управления доступом RBAC [1] которая предусматривает, что *права доступа* (или привилегии) выдаются группе пользователей, которые называются *ролями*. Эта модель хорошо себя зарекомендовала при реализации «офисных» автоматизированных систем. Пользователю назначается одна или несколько ролей, например, секретарь. С каждой ролью связан набор прав доступа, которые необходимы для выполнения служебных обязанностей. При выполнении конкретного действия, например, удаления записи из базы данных, приложение проверяет наличие у пользователя соответствующего права. Если набор атомарных прав считается фиксированным, то для изменения прав доступа конкретного пользователя, например, при переходе на новую должность, достаточно добавить запись в отношение «пользователь-роль», что не требует модификации самого

приложения. Таким образом, использование ролевой модели позволят разделить логику основного приложения и управление правами доступа.

Современные информационные системы, особенно предоставляющие возможность удалённого доступа через Интернет широкой группе пользователей, требуют реализации более сложных правил доступа к информации. Многие простые правила доступа, например, правило «каждый пользователь имеет право редактировать свои собственные данные», не могут быть естественным образом выражены в классической ролевой модели. В общем случае решение о предоставлении доступа зависит от *субъекта* (пользователя), *объекта* и *контекста* операции (например, времени или месте выполнения операции). Примером правила является условие «сотрудник технической поддержки моложе 25 лет может удалять файлы, созданные профессорами, только с 9 до 17». *Политикой* информационной безопасности будем называть набор таких правил. Подобные правила доступа могут быть реализованы на языке программирования высокого уровня, но такое решение обладает двумя существенными недостатками. Во-первых, любое изменение правил доступа требует модификации кода системы. Во-вторых, отсутствует возможность статического анализа правил доступа поскольку они реализованы на алгоритмически полном языке.

Актуальным направлением развития систем управления доступом являются модели доступа, учитывающие атрибуты объектов (ABAC) [2]. В таких моделях условие предоставления доступа определяется логическим выражением относительно значений атрибутов субъекта, объекта и контекста. Процедура вычисления значений атрибутов, как правило, не включается в модель и на практике сводится к вызову соответствующих функций. Значительным недостатком такого подхода является принципиальная невозможность проведения статического анализа политики безопасности на предмет внутренних противоречий.

В случае облака, когда информационная система предоставляется в качестве сервиса для сторонних организаций, возникает необходимость в поддержке многоуровневой политики безопасности. Некоторые правила могут быть определены на уровне все системы, другие — на уровне одной организации или её структурного подразделения. Целесообразна поддержка «переопределяемых» правил, когда организация может изменить некоторые правила, принятые в системе по умолчанию, и запрещающих правил, то есть правил, которые не допускают выполнения операции.

В данной работе предлагается модель многоуровневого логического распределения доступа и соответствующая система проверки наличия прав, которая может быть отделена от кода целевой системы, позволяет описывать достаточно широкий класс политик информационной безопасности и допускает возможность статического анализа политики. Модель, которая является развитием [3], разработана в предположении, что данные хранятся в реляционной базе данных. Атрибуты объектов определяются в терминах

отношений базы данных, а проверка прав доступа осуществляется путём выполнения SQL запросов. Для исключения дублирования и повышении наглядности описания политики информационной безопасности в модели используются элементы онтологического моделирования, что позволяет описывать правила доступа в терминах модели предметной области, а не в терминах структуры реляционной базы данных.

2. Концептно-ориентированная модель ограничения доступа

В ролевой модели RBAC права доступа носят безусловный характер. Если пользователь обладает правом «удалить файл», то он, как правило, может удалит файл независимо от его свойств. Многие варианты атрибутной политики ABAC по сути добавляют в RBAC логические выражения, которые должны выполняться для предоставления доступа. Приведённый во введении пример может быть выражен разрешающим правилом, которое связано с операцией удаления файла, ролью «сотрудник технической поддержки» и включает условие вида «subject.age() <= 25 and object.creator_post() = 'professor' and context.time_hour() >= 9 and context.time_hour() < 17». Набор функций типа age, creator_post составляется в соответствии с нуждами данной политики. Добавление нового правила доступа (или изменение условий доступа) может потребовать реализации новых функций.

Существенным, на наш взгляд, улучшением атрибутной модели является модель EBAC [3]. Предполагая, что данные хранятся в реляционной базе данных, авторы предлагают определять атрибуты в виде путевых выражений. Например, вместо функции creator_post() может использоваться выражение object.creator.post.name. Здесь предполагается, что таблица с описанием файлов (напомним, что в нашем примере объектом является файл, созданный профессором) имеется атрибут creator, который является ссылочным ключом на таблицу пользователей, у которой есть ссылочный ключ post. Использование путевых выражений, безусловно, менее универсальный метод по сравнению с вызовом функций, но, в отличие от последнего, не требует программирования при изменении политики. В EBAC допускается «движение» по ссылочным ключам в обоих направлениях (например, от пользователя можно перейти к множеству файлов, которые он создал) и вводится ряд соглашений на работу с множествами значений.

Очевидно, что логические выражения вида object.creator.post.name = 'professor' могут быть автоматически преобразованы в SQL запрос, который не выбирает ни одной записи тогда и только тогда, когда условие не выполняется. Точки в путевом выражении преобразуются в соединение таблиц (JOIN или LEFT JOIN), собственно сравнение представление условием в части WHERE, а целевой объект object задаётся первичным ключом. В рамках системы, которая предлагается в данной работе, язык путевых выражений расширяется дополнительными условиями фильтрации. Например, допускаются выражения вида object[size>5000].creator.post[part_time=0].name, которое означает, что

файл должен быть больше 5000 единиц и место работы должно быть основным. Такие расширенные выражения могут быть аналогичным образом сведены к вычислению SQL запросов.

В целях иллюстрации принципов концптно-ориентированной модели ограничения доступа рассмотрим представленный на рис. 1 фрагмент модели «Сущность-Связь» информационной системы, содержащей данные о исследовательских проектах. В данном примере данные о проектах и их участниках представлены в четырёх таблицах базы данных: проектах, источниках финансирования, участниках и сотрудниках. Следует отметить, что выбор таблиц и их атрибутов может зависеть от персональных предпочтений проектировщика базы данных и в другой системе аналогичная информация может храниться в другой схеме данных.

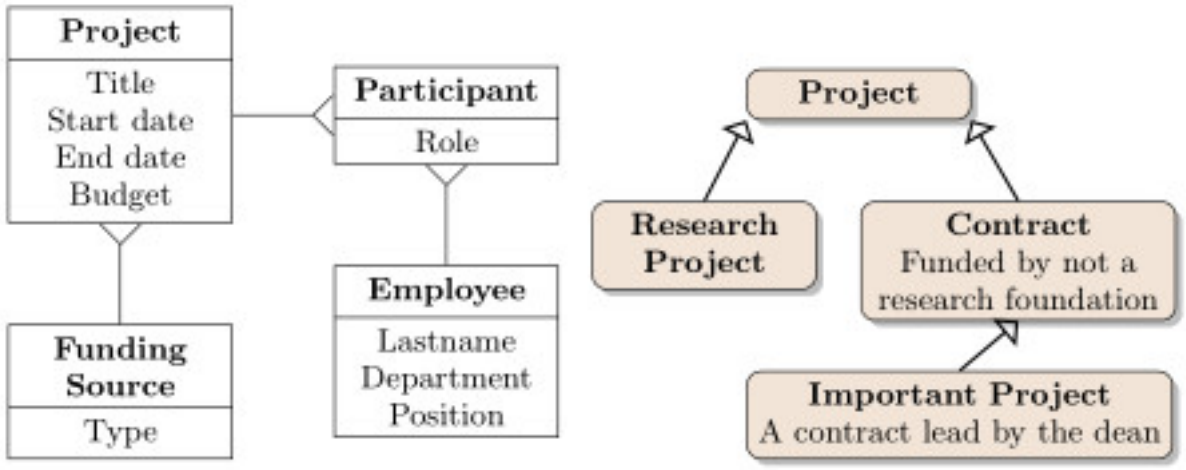

Рис. 1. Фрагмент модели «Сущность-связь» (слева) и иерархии понятий.

Политика безопасности может быть сформулирована в терминах «Грант», «Контракт с коммерческой организацией», «Важный контракт», где каждое понятие определяется значениями атрибутов. Например, грантом называется любой проект, источник финансирования которого имеет тип «фонд». Или «Важным считается проект, который выполняется по заказу коммерческой организации, сумма финансирования которого превышает 10 единиц и руководителем которого является декан факультета». Сама политика безопасности может включать правила типа «Важные проекты удалять нельзя».

Концептно-ориентированная модель предполагает, что описание политики включает:

- определение модели данных (структура таблиц и отношений базы данных);
- иерархию понятий;
- набор правил доступа.

Без ограничения общности можно считать, что понятия разделяются на сущности (то есть понятия, которые отображаются в таблицы базы данных) и

собственно понятия. Сущности определяют набор атрибутов и всю необходимую для генерации SQL запросов информацию (имена таблиц, ссылки на другие таблицы, атрибуты первичного ключа и т. п.). Собственно понятия наследуются от других понятий (или сущностей) и определяют условия, при выполнении которых объект базы данных может считаться экземпляром этого понятия. Например, понятие Грант может быть производным от понятия-сущности Проект и содержать условие object.funding.type='Research foundation'.

Каждое правило доступа определяется

- типом правил (разрешающее или запрещающее);
- понятием, к экземплярам которого это правило применимо;
- названием операций;
- списком ролей или пользователей, для которых это правило определено;
- условиями применимости правила.

Например,

```
(rule delete-important-project        ;; название правила
  (object ImportantProject alias proj) ;; зададим псевдоним для object
  (grantee
     (role “registered-user”))        ;; кто может выполнять операцию
  (operation “delete”)
  (constraint                         ;; дополнительные условия
    ;; “маленький” проект, пользователь является руководителем
    proj.budget < 100 and proj.leader.id = user.id))
```

Строго говоря понятия являются «синтаксическим сахаром». Условия, которые указываются при определении понятия, могут быть перенесены на уровень правила. Приведённое выше правило можно связать с сущностью «Проект», если в условие добавить ещё один конъюнкт означающий, что проект является важным. При наличии нескольких правил потребуется копирование части условий между правилами, что, очевидно является причиной потенциального появления ошибок. **Основные преимущества понятий** состоят в следующем.

- Понятия позволяю избежать дублирования условий, так как общая часть условий может быть перенесена на уровень понятия.
- Понятия могут сделать политику безопасности более короткой и ясной. Человеку, который будет редактировать политику безопасности не нужно будет каждый раз пытаться восстановить содержательный смысл логических выражений.
- Понятия позволяют описывать политику безопасности на высоком уровне абстракции. Модель «Сущность-Связь» отражает особенности конкретной схемы базы данных, в то время как понятия являются элементами концептуальной модели предметной области.

Политика безопасности определяется для административной единицы организации. Фактически можно считать, что каждое правило относится к

некоторому элементу иерархической структуры организации (или леса организаций, если речь идёт об облачной системе). Для каждого запроса на доступ, то есть набора (таблица, первичный ключ, пользователь, операция, контекст), необходимо составить список применимых разрешающих и запрещающих правил, учитывая иерархию подразделений. Проверка применимости конкретного правила доступа сводится к выполнению следующих действий:

- определить список понятий, экземплярами которых является объект;
- проверить выполнение условия, определённого для данного правила.

Доступ предоставляется, если существует по крайней мере одно применимое разрешающее правило и не существует применимых запрещающих правил.

3. Реализация

Программная реализация сервера проверки прав доступа, доступная по адресу https://github.com/sergadin/SecurityServer, позволяет работать с базами данных SQLite, mySQL, PostgreSQL и Oracle. Поддерживается загрузка политики безопасности из конфигурационных файлов, передача запросов на проверку прав доступа производится по протоколу HTTP. Каждый запрос включает название таблицы и значения первичного ключа (эта пара определяет запрашиваемый объект), название операции и идентификатор пользователя.

Язык описания ограничений допускает использование соединений таблиц (оператор точка) и фильтрацию промежуточных записей произвольными логическими выражениями относительно атрибутов промежуточных таблиц. Выражения фильтрации могут использовать оператор точка с вложенными условиями фильтрации. Поддерживаются кванторы существования и всеобщности для путевых выражений, выбирающих множество значений. Проверки доступа осуществляются путём выполнения последовательности SQL запросов.

Тестирование проводилось на данных и фрагменте политики безопасности информационной системы «ИСТИНА» [4]. Политика безопасности содержала 12 понятий, максимальная глубина вложенности структурных подразделений составляет 5 уровней. При обработке файла, содержащего последовательность из 20 000 запросов реальной системы, производительность сервера проверки прав доступа составила приблизительно 15 запросов в секунду.

4. Заключение

В данной работе описывается система проверки прав доступа, которая позволяет отделить методы описания и проверки политики безопасности от кода целевой информационной системы. Предварительные результаты тестирования, полученные на реальных данных информационной системы

«ИСТИНА», показывают, что предлагаемый метод может применяться при решении практических задач. Перспективы дальнейших исследований в рамках данного направления связаны с разработкой алгоритмов статического анализа политик безопасности и разработкой методов оптимизации последовательности выполняемых SQL запросов.

Литература

1. Ferraiolo, D. F., Barkley, J. F., and Kuhn, D. R. A role-based access control model and reference implementation within a corporate intranet. // ACM Trans. Inf. Syst. Secur. 2, 1 (Feb. 1999), 34–64.
2. Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., and Scarfone, K. Guide to attribute based access control (ABAC) definition and considerations. // Tech. rep., National Institute of Standards and Technology (NIST), 2014.
3. Bogaerts, J., Decat, M., Lagaisse, B., and Joosen, W. Entity-based access control: supporting more expressive access control policies. // Proceedings of the 31st Annual Computer Security Applications Conference, ACM, 291–300, 2015.
4. Интеллектуальная Система Тематического Исследования НАучно--технической информации (ИСТИНА) / В. А. Васенин, С. А. Афонин, Д. Д. Голомазов, А. С. Козицын // Информационное общество. — 2013. — № 1-2. — С. 21–36.