Цель разработанного модуля: обеспечение безопасного доступа к ресурсам приложения с применением role/privilege-based security и row-level security.

Для обеспечения role-based security была использована библиотека spring-security c кастомизацией под нашу БД и требования. В отличии от базового функционала spring-security у нас добавились привилегии (privileges). Привилении могут назначаться пользователю через роли. Привилегия включает в себя два атрибута: тип объекта и действие, выполняемое над объектом. Т.к. spring-security позволяет ограничивать доступ к методам с помощью аннотаций и выражений, то для работы с привилегиями были добавлены выражения hasPrivilege/hasAnyPrivilege.

Модуль допускает наследование ролей. При наследовании ролей наследуются все привилегии от родитеской роли. Родительские роли с привилегиями не грузятся одновременно с загрузкой данных пользователя. Иерархия всех ролей загружается при необходимости и кэшируется. Далее при проверке доступа, в случае если ролей, явно назначенных пользователю, не достаточно, подтягиваются роли из кэша. Тут используется API spring-security (RoleHierarchy).

Для того, чтобы внедрить свои выражения для аннотаций spring-security, были реализованы интерфейсы обработчиков выражений spring-security. В качестве реализации взят код из библиотеки и доработан по нужды модуля. Выражение hasAuthority при этом было отключено. Т.к. оно основано на сомнительном механизме определения типа authority по префиксу имени (например ROLE\_USER роли USER).

В нашем модуле все authority имеют свои реализации от GrantedAuthority и не требуют префиксов.

Примеры аннотаций для методов:

@PreAuthorize("hasRole(USER)")

@PreAuthorize("hasPrivilege('Customer', 'READ') ")

Для выполнения аутентификации использован DaoAuthenticationProvider, из состава spring-security. Этот провайдер позволяет кэшировать данные пользователей (ключая роли и группы). Кэширование поключено.

Row-level security реализован в виде ACL (Access Control List). Общий подход был подсмотрен в spring-security-acl. Но сама spring-security-acl не использовалась из-за множества расхождений с нашими требованиями. Так же в коде spring-security-acl было обнаружено много “костылей”, на которые не хотелось опираться. Выдача прав на объект может выполняться непосредственно пользователю или через группу. Структура таблиц для хранения ACL будет описана позже.

Защищаемые объекты здесь представлены в виде AclObjectIdentity. Identity содержит ссылку на бизнес-объект и тип бизнес-объекта. Также у identity есть ссылка на пользователя (с FK). Identity допускает указание родительского identity и возможность наследовать ограничения доступа от родительского identity.

Для каждого AclObjectIdentity можно указать множество AclEntry. AclEntry определяет доступ к бизнес-объекту, описанному в AclObjectIdentity. AclEntry содержит ссылку на AclObjectIdentity, ссылку на principal (пользователя или группу), бинарную маску, содержащую список прав (permission) на бизнес-объект.

Определение возможности доступа к объекту выполняется по следующему алгоритму:

1. производится поиск AclObjectIdentity, соответствующего бизнес-объекту
2. если установлен флажек наследования, то к первому AclObjectIdentity добавляются его родительские AclObjectIdentity
3. если AclObjectIdentity найдены, то выполняется определение списка всех principal, соответствующих пользователю (это сам пользователь плюс группы)
4. в списке AclObjectIdentity ищутся AclEntry, содержащие ссылку на principal
5. если найден хоть один AclEntry, который запрещает доступ к исходному объекту (либо его родителям), то доступ пользователя к объекту запрещается
6. если не найдено ни одного AclEntry, т.е. доступ к объекту не определен ни для одного из principal пользователя, то возможность доступа определяется по ролям/привилегиям пользователя
7. если пользователь имеет привилегию на выполнение необходимого действия на объектами типа исходного, то доступ разрешается (тут необходимо сделать привязку действий над объектами (action) и правами на объекты(permission))

Правила ACL можно применить к ресурсу приложения несколькими способами:

1. C помощью аннотаций spring-security и выражения hasPermission. Таким образом например можно отфильтровать список объектов, принимаемые методом в виде параметра или возвращаемый методом. Фильтрация происходит только до или после выполнения метода. Но не во время выборки данных из источника данных.

Пример использования аннотаций:

@PreAuthorize("hasPermission(#objectId, 'ObjectType', 'READ')")

@PostFilter("hasPermission(filterObject, 'WRITE')")

1. С помощью аннотации из нашего модуля: AclObjectFilter. Эта аннотация позволяет применить ограничения доступа на уровне DAO. Достигается это путем внедрения фильтрующего условия в SQL запрос.

Пример использования аннотаций:

@AclObjectFilter

@AclObjectFilter(@AclRule(type = Customer.class, permissions = {Permission.WRITE}, inherit = true))

Ограничение доступа (фильтрация) на уровне DAO пока что реализована только для Hibernate ORM. В SessionFactory на этапе инициализации внедряются фильтры для сущностей, отмеченных интерфейсом SecuredObject. Далее на DAO вешается аспект, выполняющий анализ аннотации AclObjectFilter и включающий (и выключащий после выборки данных) фильтры выполняющие фильтрацию на основании ACL.

**Описание таблиц**

1. ACL\_ENTRY – описание прав доступа к объектам для принципалов

id – идентификатор

object\_identity\_id – ссылка на описание объекта acl\_object\_identity

principal\_id – ссылка на principal

permission\_mask - бинарная маска c парвами (READ – 1, WRITE – 2, CREATE – 4, DELETE – 8, ADMIN - 16)

1. ACL\_OBJECT\_IDENTITY – описание бизнес-объектов в ACL

id – идентификатор

object\_type\_id – ссылка на тип объекта

object\_id – идентификатор объекта

parent\_id – идентификатор родительского acl\_object\_identity

owner\_id – идентификатор пользователя-владельца объекта (владелец может все)

inheriting – признак наследование прав от родителя

1. USER – пользователи

id – идентификатор

user\_name – имя пользователя

password – хэш пароля (BCrypt)

active – признак активного пользователя

1. ROLE – роли

id – идентификатор

name – название роли

description – описание роли

parent\_id – ссылка на родительскую роль

1. USER\_ROLE – связь многие-ко-многим для пользователей и ролей

user\_id - идентификатор пользователя

role\_id – идентификатор роли

1. GROUP

id – идентификатор

name – наименование группы

description – описание группы

1. USER\_GROUP - связь многие-ко-многим для пользователей и групп

user\_id – идентификатор пользователя

group\_id – идентификатор группы

1. PRINCIPAL

Id – идентификатор. Общий ключ для пользователей и ролей. На него FK из ACL\_ENTRY

1. OBJECT\_TYPE

id - идентификатор

name – наименование типа объекта

description – описание типа объекта

1. ACTION – действия, допустимые над объектами

id - идентификатор

name – наименование действия

description – описание действия

1. PRIVILEGE – привилегии

id - идентификатор

object\_type\_id – идентификатор типа объекта

action\_id – идентификатор действия

1. ROLE\_PRIVILEGE - связь многие-ко-многим для ролей и привелегий

id - идентификатор

role\_id – идентификатор роли

privilege\_id – идентификатор привилегии