CincoSecurity:
Automating the Security of Java EE Applications with Fine-Grained Roles and Security Profiles

María Consuelo Franky
Department of Systems Engineering
Pontificia Universidad Javeriana
Bogotá, Colombia
lfranky@javeriana.edu.co

Victor Manuel Toro C.
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá, Colombia
vm.toro815@uniandes.edu.co

***Abstract*— Almost every software system must include a security module to authenticate users and to authorize what elements of the system can be accessed by each user. This paper describes a security model called "CincoSecurity" that follows the Role Based Access Control model (RBAC), but implementing fine-grained roles that can be grouped into "security profiles". This leads to a great flexibility to configure the security of an application by selecting the operations allowed to each security profile, and later, by registering the users in one or several of these profiles. We describe also a security software module (that implements the CincoSecurity model) that we propose to be the initial code baseline for the development of any Use Cases oriented Java EE system, offering from the beginning a flexible, extensible and administrable access control to the elements of the application that is to be developed. Moreover, CincoSecurity allows automating the generation of the additional code required to protect the use cases and its elements of the Java EE application being developed, with tools that add the required security restriction code accordingly with the proposed security model.**

Keywords- Security; Access control; RBAC; Framework; Java EE; Seam; Security automation.

I. INTRODUCTION

This paper summarizes the experience of the authors designing and developing a reusable security module, called CincoSecurity, that has been used for several years to control access to the elements of web applications written in Java Enterprise Edition (J2EE initially [9] and later Java EE 5 [10]). Currently, the module CincoSecurity is available [18] under the GPL license, and is used by some important software houses in Colombia.

The security model underlying CincoSecurity implements a RBAC (Role-Based Access Control) [7], providing high flexibility to control access to the various elements of a Web application, such as the invocation of an operation of a business component, the access to a web page, or the access to elements within that page. The innovation of CincoSecurity is the use of very fine-grained roles, each role having a single permission associated with the invocation of an operation (method) of a business component. From these fine roles —whose fulfillment the Application Server can directly control at run-time— CincoSecurity allows to define "security profiles" as sets of fine-grained roles. This facility of security profiles gives a great flexibility for configuring the security of an application by selecting the operations allowed to each profile (i.e., selecting a set of fine-grained roles for each profile), and later, by registering the users in one or several of these profiles.

A Java EE web application that is to be constructed with the Seam framework [12] gets several benefits by integrating the CincoSecurity module. When a user authentication is performed, the Application Server is informed about the fine roles derived from the security profiles the user belongs to, and a personalized menu is dynamically built containing only the entries leading to the use cases allowed for the user. Additionally, CincoSecurity contributes to the application being constructed with several use cases to administer the security profiles, to manage user registration in these security profiles and to administer passwords. Additionally, CincoSecurity comes with use cases to register new modules, new use cases and new services, as they become available during the development project, for their security to be administrable.

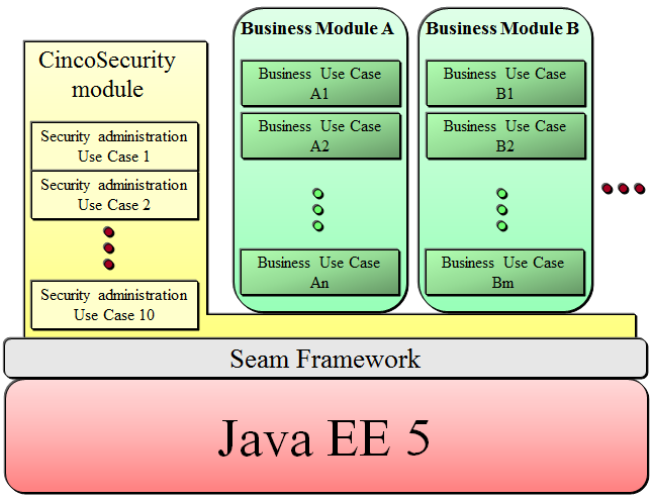

Figure 1: What is the CincoSecurity module?

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Figure 1 illustrates the CincoSecurity module as a platform to embed and support security (and its administration) into a new Java EE application. This means that, after generating the very initial codebase of the new application with the Seam framework [12], CincoSecurity shall be the first module to be coupled into the new application, in order to be able to include and administer security of the forthcoming business modules.

Once embedded into the new application that begins to be developed, the CincoSecurity module facilitates to automate the incorporation of security into the new modules as they are built, by the means of tools that add security restrictions to each new use case and its elements, and with administrative use cases (coming with CincoSecurity) that configure the security of the whole application. With respect to previous work of the authors [1], this paper is an extension that explains in detail how to automate the incorporation of security into a new Java EE application by applying the CincoSecurity model.

In the following section, this paper presents the RBAC security model on which CincoSecurity is based, and more specifically, the RBAC model applied in the context of Java Application Servers. Then, additional concepts provided by the CincoSecurity module are introduced, as well as its entities model. Later, there is a description of the use cases coming with the CincoSecurity module (e.g., create a new user, create/edit a security profile, add/delete users from a security profile, etc.). Then, the paper provides a short summary and references to the detailed guidelines [19] for integrating the CincoSecurity module to a Java EE application built with the Seam framework [12]. At the end, the security automation of an application that uses CincoSecurity is explained. Finally, there is a comparison with other works, followed by the conclusion and a short description of our future work.

II. EVOLUTION OF THE RBAC SECURITY MODEL

The RBAC model introduced the concept of "role" to control the access to computing resources. The RBAC term was first proposed by Ferraiolo and Kuhn [3], based on previous works of Baldwin [2]. The initial proposal of this model creates a role for each type of job within an organization (cashier, customer service person, office director, ...). Then, each role is assigned with the set of access permissions that are required for this type of job. Finally, each user is enrolled into one or more roles (rather than to specific permissions). This model simplifies security management because the roles (with their associated permissions) tend to be stable, and users can be added or retired easily from roles. The RBAC model allows reinforcing the "least privilege" principle by giving each user the minimum set of permissions required to perform his work, by enrolling him only in the appropriate roles [7].

From the initial RBAC model (called Core RBAC) the work of Sandhu and colleagues [4] defined extended models, such as the hierarchical RBAC (to include role hierarchy with inheritance of permissions), and constrained RBAC (to prevent, for example, to assign a user to two conflicting roles, or to restrict the time interval in which a user can use the permissions of one of its roles).

The main applications of the RBAC model have been in Data Base management Systems, Enterprise Security Management Systems, and Web applications that run on Application Server s [6] [7] [8].

The wide spread of RBAC models, implemented in numerous products from many providers, led to define an ANSI standard [5] in 2004, aiming to standardize terminology, promote its adoption and improve productivity. However, the current RBAC ANSI standard (consisting of a reference model and a functional specification) has some limitations and gaps as indicated in the work of Bertino and colleagues [8].

III. THE RBAC MODEL APPLIED TO JAVA EE APPLICATION SERVERS

Since the late 90's, the emergence of Application Servers brought a new way to build web applications (both in the enterprise Java platform and in Microsoft .NET), with business components managed by containers that provide added services for security, transaction management, parallelism, pool of connections, logging, etc. [9].

Regarding security, Java EE Application Servers [10] implement the Core RBAC model [7] to control the access to resources based on the roles the user belongs to. In order to take advantage of these security services (and not to write additional code in the application to internally control the access to resources), it is necessary to specify the roles of the application, the association of resources to roles, and the association of users to roles.

A. Enrolling users in roles

In a Java EE application that uses a database to store the authentication and authorization information, the following entities EJB3 (Enterprise Java Beans - version 3) are required [11]:

- An entity "User" shall be implemented (with its corresponding support table in the database), to store users and passwords.
- Entities shall be implemented (with its support tables in the database) to specify the association of each user with one or more roles.
- A "User management" use case shall be implemented to enroll a user in one or more roles.

These facilities are included in CincoSecurity. Similarly, it is also possible to store users, passwords and roles in a LDAP (Lightweight Directory Access Protocol) server.

B. Controlling access to resources

Seam is a framework to develop Java EE applications, that is being developed by JBoss since 2005, whose principal author is Gavin King [12] [14]. Seam allows to directly expose and use in the Web layer the entities and business components of the application. This simplifies enormously the development by eliminating the intermediaries and conversions between the layers of the application. Seam has been widely accepted and has been incorporated in the recent

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Java EE 6 standard, under the name of "CDI" (Contexts Dependency Injection).

To control access to resources in a Java EE application that uses the Seam framework, the following strategies are required [13]:

- An annotation is used to protect each method of the session EJB3. This annotation indicates what roles are authorized to invoke the method.
- The url of each JSF (JavaServer Faces) web page [10] can be protected in the navigation flow descriptor (pages.xml) so that it can be accessed only by users belonging to one of the specified roles.
- Each button or element of a JSF web page can be protected so that it is rendered only to users belonging to one of the specified roles.

Notice that annotations must be scattered along the code —in the declaration of methods, in the section of a page in the navigation flow descriptor, in the buttons tags and elements of JSF pages— to indicate what roles can access these elements.

C. User authentication

In a Java EE application that uses Seam, the authentication service must be specified in the descriptor components.xml. This service shall be a method of a class of the application, and must implement a query in JPQL (Java Persistence Query Language) [11] to verify the user's password (alternatively this process can also be performed with a LDAP server).

Additionally, the authentication service must also obtain the roles of the authenticated user. With the Seam component called "Identity" these roles can be added to the session and informed to the Application Server.

D. Controlling access to a JSF page

When an http access request is received, the Application Server verifies if the user belongs to a role allowed to access the requested JSF page, and if so, the requested page is displayed.

For example, in the following piece of the navigation flow descriptor it is specified that the access to myPage.xhtml is granted only to users having the 'tourist' role:

```
<page view-id="/myPage.xhtml" login-required="true">
   <restrict> #{s:hasRole('tourist')} </restrict>
</page>
```

Similarly, inside the page only the elements that the user is authorized to see are shown (elements such as buttons and text boxes can specify, with the attribute "rendered", what roles can see them). For example, in the following piece of page it is specified that the button "View hotel" is visible only to users with the 'tourist' role:

```
<s:button id="viewHotel" value="View hotel"
   action="#{hotelBooking.viewHotel(hot)}"
   styleClass="buttonSmall"
   rendered="#{s:hasRole('tourist')}"
/>
```

E. Authorizing an action from a JSF page

In a JSF page a button's action is typically associated with the invocation of a method of a session EJB3. The server verifies that the user roles allow him to invoke the associated method, assuming that the method is protected by an annotation indicating the roles that can invoke it.

For example, in the following piece of a session EJB, the method viewHotel is allowed only to users with the 'tourist' role:

```
@Restrict ("#{s:hasRole('tourist')}")
public void viewHotel(Hotel hot) {...}
```

IV. ADDITIONAL CONCEPTS IMPLEMENTED BY THE CINCOSECURITY MODULE

In addition to the security concepts for a Java EE application that uses the Seam framework [12] [13] explained above, the CincoSecurity module implements additional concepts to provide greater flexibility to define the permissions for users.

A. Use case and services

Definition: A use case is a system's capacity to deliver a useful and indivisible functionality to the user.

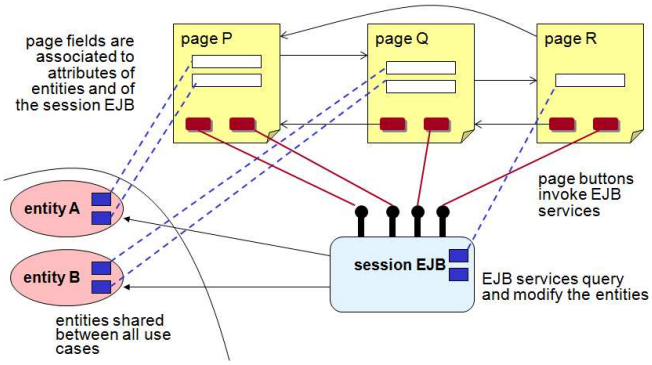

Figure 2: Elements of a Use Case implemented in Java EE with Seam

Definition of a use case in terms of its implementation in Java EE with Seam (see Figure 2): a use case consists of one (or more) business entities and a group of services that act upon them. These services are implemented as methods of a session EJB3 (see "use case controller" pattern). One or more JSF pages display attributes of the entities involved in the use case, and attributes of the session EJB3 controlling it. In those JSF pages there are actions that invoke the services of the session EJB3. These EJB3 services (methods) are programmed in terms of queries and modifications to the persistent business entities. The navigation flow descriptor contains rules to decide the next page to display.

B. Module

Definition: A module is a set of related use cases. The CincoSecurity module comes with the following use cases, that will be explained below: security profiles management, users management, change of password, basic security reports, registration of menu entries, and registration of modules, use cases and services.

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

C. Fine grained roles

The CincoSecurity module works with fine-grained security roles:

- A role for entering to each use case. The name assigned to this role is the same name of the use case (which is also the Seam name of the session EJB3 that supports the use case).
- A role to invoke each service within a use case (i.e., each of the methods of the session EJB3 that supports the use case). The name assigned to this role is "use case name"_ "method name".

D. Protection of resources

- The session EJB3 that supports a use case is protected with an annotation indicating the role for entering to the use case. For example, the profileGestion use case is supported by the session EJB3 ProfileGestionAction.java (which implements the interface ProfileGestion.java); this EJB3 has the Seam name "profileGestion". Consequently, the role for entering to this use case is called "**profileGestion**" and the EJB3 class will have the following annotation:

```
@Restrict("#{s:hasRole('profileGestion')}")
```

- Each service (method) of the session EJB3 is protected by an annotation indicating the role associated with the service. The name of this role is the concatenation of the use case name with the name of the service (with "_" between). For example, the **update** method of the session EJB3 that supports the use case **profileGestion** will have the following annotation:

```
@Restrict("#{s:hasRole('profileGestion_update')}")
```

- Methods get and set do not require any annotation: they are protected with the role of entering to the use case.
- Access to each JSF page of a use case is protected in the navigation flow descriptor by the role for entering to the use case. For example, the page **profiles.xhtml** of the use case **profileGestion** has a navigation flow descriptor called **profiles.page.xml** that contains the following restriction:

```
<page view-id="/profileManagementInit.xhtml"
        login-required="true">
   <restrict>
      #{s:hasRole('profileManagement')}
   </restrict>
</page>
```

- Each button in a JSF page should be displayed only to users having the role associated to invoke the action of the button, which corresponds to an EJB3 service (method). For example, the **profiles.xhtml** page of **profileGestion** use case contains a button whose associated action is to invoke the **update** method of the session EJB3 that supports the use case. Consequently the button tag indicates that it is showed only to the role **profileGestion_update**:

```
<h:commandButton id="update"
   value="Update"  styleClass="button"
   action="#{profileGestion.update}"
   rendered="#{s:hasRole('profileGestion_update')}"
/>
```

E. Security profile

A security profile is a set of fine roles, each fine role expressing the right to invoke a service belonging to a use case. Unlike the role, the concept of security profile is not supported directly by Application Servers and must be implemented with additional entities.

The use cases of the CincoSecurity module allow the association of users to roles via security profiles:

- A user can be enrolled in one or more security profiles, so he/she will have the set of fine roles allowed by the union of these profiles.
- There is a *many-to-many* relationship between users and security profiles.
- There is a *many-to-many* relationship between security profiles and fine roles.

F. Actions after a user authentication

After a user is authenticated, CincoSecurity calculates all the fine grained roles from the security profiles the user belongs to, and informs them to the Application Server (by assigning these roles to a Seam component called "Identity"). Additionally, the EJB3 Login performs the following actions:

- The session timeout is set, according to the parameters stored in the database.
- The user's menu is built, containing only the entries leading to use cases allowed to the user.
- The security information of the user is added to the session context, should the application logic needs it.

It is important to remark that the access to use cases not authorized to a user by any profile is prevented in two ways. From one side, not authorized use cases do not appear in the user's menu. From the other side –even if the user types in the url of a not authorized use case– the Application Server throws a security exception. This happens because the fine role for entering to this use case was not included in the list of fine roles that was informed to the Application Server.

Figure 3: User menu allowing access to all use cases of CincoSecurity

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The screen snapshot of Figure 3 shows the menu of a user that is enrolled in security profiles allowing access to all the use cases of the CincoSecurity module.

Figure 4: User menu allowing access to fewer use cases of CincoSecurity

The screen snapshot of Figure 4 shows the menu of another user that is enrolled in security profiles allowing access to just a few use cases of the CincoSecurity module.

V. ENTITIES MODEL OF THE CINCOSECURITY MODULE

The entities model shown in Figure 5 illustrates the relationship one-to-many from Module to Usecase, and from Usecase to Service.

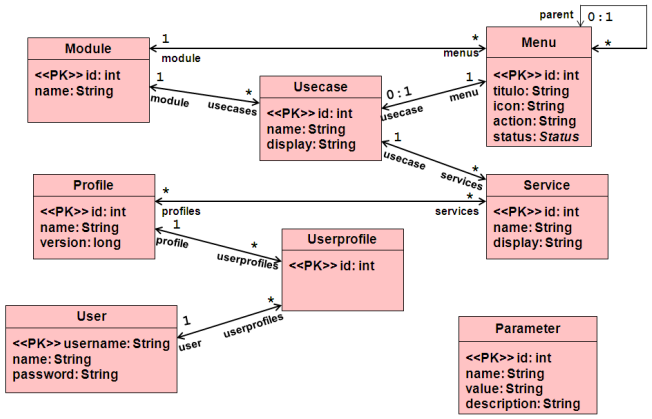

Figure 5: Model of entities of the CincoSecurity module

Figure 5 also illustrates the relationship many-to-many between Profile (security profile) and Service, as well as between Profile and User (via the intermediate entity Userprofile). Each menu entry may have submenus (only terminal menu entries have an action for going to the entry page of a use case).

The system parameters are arbitrary. They can be used, for example, to record the session timeout, the path of the directory to store reports, the address of the printer, etc.

In addition to this entity model, the CincoSecurity module also contains a view that directly associates a user with fine roles. The fine roles of a user are the union of the roles associated with the profiles the user belongs to.

VI. USE CASES OFFERED BY THE CINCOSECURITY MODULE

The following are the use cases offered by the CincoSecurity module:

A. *CRUD Use cases*

The CincoSecurity module offers:

- A use case to list/add/edit and remove **parameters** of the application.
- A use case to list/add/edit and remove **modules** of the application:

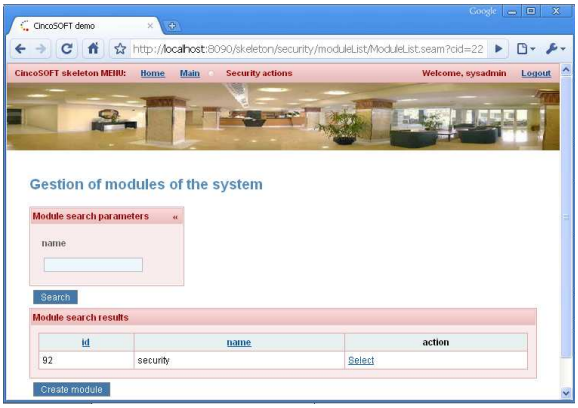

Figure 6: Use case to list/add/edit or remove modules.

- A use case to list/add/edit and remove the **use cases** of a module:

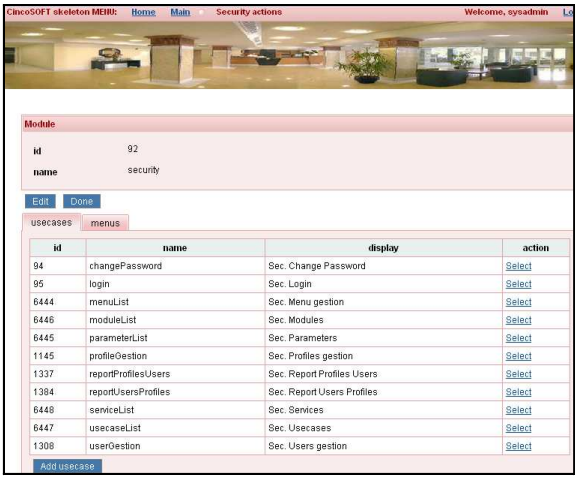

Figure 7: Use case to list/add/edit or remove use case.

- A use case to list/add/edit and remove the **services** of an application's use case.

- A use case to list/add/edit and remove **menu entries**:

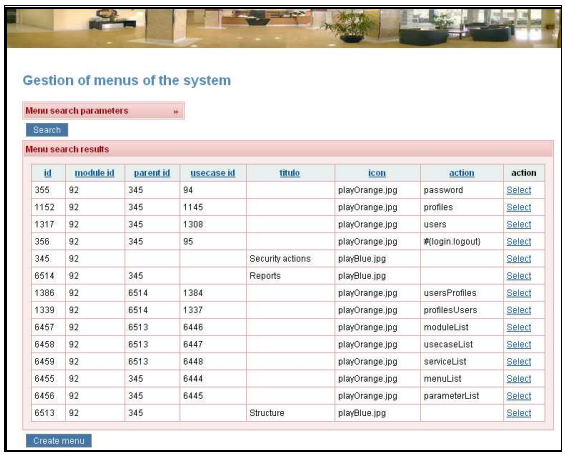

Figure 8: Use case to list/add/edit or remove menu entries.

It can be easily specified what menu entries have a submenu, as well as the use case associated with a terminal menu entry (see Figure 8).

B. *Management of security profiles*

This use case allows to add/edit/remove security profiles. Initially, the existing security profiles are listed. When a security profile is selected, the modules, use cases and allowed services are shown, so that the user can check or uncheck services (see Figure 9).

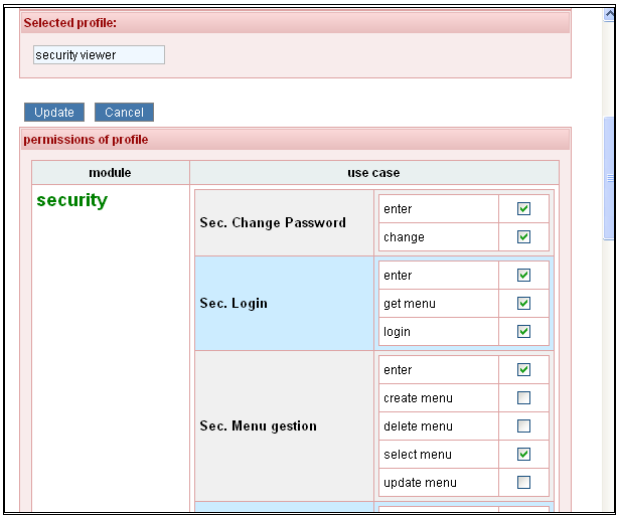

Figure 9: Use case to manage security profiles.

Similarly, the user can create a new security profile. In this case, the system displays all modules, and within it, the use cases and services, for the user to select those allowed by the new profile.

C. *Management of users*

This use case allows adding users of the application, indicating its name, login and password. It also allows enrolling the new user in one or more security profiles.

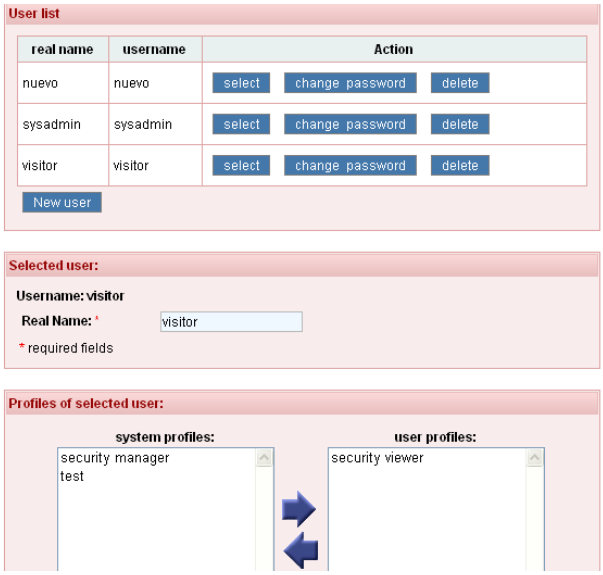

Figure 10. Use case to manage users.

D. *Password change*

This use case allows a user to change his password. Passwords are stored encrypted. See figure 11:

Figure 11. Use case to change password.

E. *Report of security profiles vs users*

This use case shows, for each security profile, what users are enrolled.

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

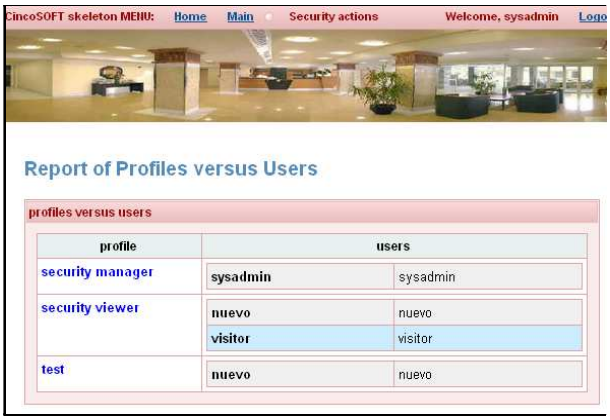

Figure 12. Use case to show a report of profiles vs. users.

F. *Report of users vs security profiles*

This use case reports, for each user, in what security profiles he/she is enrolled.

VII. How to integrate the CincoSecurity module to a Java EE Seam application

The CincoSecurity module is open source with GPL License [17]. It can be downloaded from SourceForge [18] in the form of an Eclipse project [16]. It comes ready to be deployed on the Application Server JBoss [15], but can be installed in any other Java EE Application Server by following the guidelines provided in the Seam manual [14].

The documentation accompanying the CincoSecurity module explains in detail how to deploy and execute the module, and how to integrate it with a Java EE application built with Seam. In particular, detailed explanations are included for registering application's use cases and services in the security module. An earlier publication about the CincoSecurity module, oriented to programmers, focused on these technical details [19].

It is important to emphasize that to incorporate and manage the security of an application, the modules of the application, the use cases contained in such modules, and the services offered by these use cases must be registered into the CincoSecurity module (by using CincoSecurity's use cases provided for this). This way, the fine roles associated with these services can be included in the security profiles, and the access to these use cases will appears in the menu of authorized users.

VIII. How to automate the protection of an application that uses the CincoSecurity module

The fine roles and the naming discipline proposed above to protect the services of the session EJBs and the web elements that invoke these services, allow to think in automating the protection of resources of an application that uses the CincoSecurity module. Indeed, from the names of the services to be protected, a tool could insert in the code the annotations (in the java sources) and the tag elements (in the JSF pages) required to achieve such protection.

With traditional coarse roles it is not possible to make such automation of the protection, because each time a role needs to be added or changed, it is necessary to review the whole code to decide what EJB services and web elements must be protected with the new role, thus requiring to manually write the appropriate annotations and tag elements.

The following section describes the main ideas of a generation framework based on Regular Expressions techniques [24], currently being developed by the authors, that automates the protection of a Java EE application that uses the CincoSecurity module.

A. *Techniques of Regular Expressions to build code generators*

The tool required to process Regular Expressions must be capable of detecting in a text file the strings –either in a single line or multi-line– that match a given tagged regular expression, and then, transforming the detected strings using the tags. These features are the basis for building a framework for code generation.

Example of tools that process tagged regular expressions are the java.util.regex library [24] [25] for Java programs and the replaceregexp command provided by the tool **ant** [26]. Below is an example of using this command to add the prefix **new_** to the name of each property in a set of files that contain properties (for example, a line with **aa = some string** becomes **new_aa = some string**):

```
<replaceregexp
  match="([^\s]+)=([^\s]+)"
  replace="new_\1=\2"
  byline="true"
>
    <fileset dir=".">
        <include name="*.properties"/>
    </fileset>
</replaceregexp>
```

In the previous example the regular expressions **[^\s]+** represents a word (*one or more non-space character*). The **match** attribute contains a regular expression that describes a property, enclosing in a first parenthesis the property name (tag \1) and enclosing in a second parenthesis the value of the property (tag \2). These tags are used in the **replace** attribute, that indicates that the **new_** string must be added before the name of the property. The command also indicates that the transformation must be done for each file of the form *.properties in the current directory, by analyzing each line separately.

The following example adds an annotation before the declaration of each Java class, which specifies the restriction that the user must be authenticated:

```
<replaceregexp
  match="(public class)"
  replace="@Restrict("#{identity.loggedIn}")  \1"
  byline="true"
>
    <fileset dir=".">
        <include name="*.java"/>
    </fileset>
</replaceregexp>
```

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

With techniques of Regular Expressions is also possible to extend source files. For example, the commands **replaceregex** and **loadfile,** provided by the tool **ant**, allow to assign to a property the text that matches a Regular Expression in a first source file, and then, inserting such text into a second source file, in the place matching a second Expression Regular.

B. *Framework of code generation based on on techniques of Regular Expressions*

The JBoss Seam generator [13] generates the initial skeleton of a Java EE 5 application with the following elements:

- Several control elements, based on the infrastructure frameworks JSF [10], Seam [13] and EJB3 [11].
- Several descriptor files correctly configured.
- An **ant** script [26] with tasks for compiling, packaging and deployment.
- Inclusion of many libraries providing infrastructure frameworks.

However, the code generated by Seam is not enough to be the basis of a serious business application. It does not have important features, such as: the organization of the source code into separated modules (each one with a set of related use cases) to facilitate maintenance; the inclusion of a module with use cases for managing the security; the inclusion of security constraints by fine roles in order to protect both the web pages and the business components.

Consequently, once the skeleton of a new application is generated with Seam, it is necessary to reorganize this initial skeleton and to couple it with the security module. An appropriate tool could incorporate the source code of the CincoSecurity module into the application. We have developed such tool based on techniques of Regular Expressions. The tool takes a copy of code pieces of CincoSecurity module, and then it adds or transforms the text, and incorporates the result into the software project under construction.

Other tools, also based on techniques of Regular Expressions, can add to the project use cases skeletons that facilitate to use JMS message queues [27], or to use report generator capabilities, or to produce pdf files, etc..

The set of all these tools is what we call a Code Generation Framework based on techniques of Regular Expressions. Each tool works from proven source code, which is copied, transformed and incorporated into the project that is being constructed.

C. *Generator to couple the CincoSecurity module*

The current version of the CincoSecurity module can be incorporated only to a Java EE 5 application that was initially created with JBoss Seam generator [13].

The CincoSecurity module can be directly taken as the generation model for the new application. The generator tool will look for certain strings in the source files of the module and will replace these strings with the appropriate strings for the new application. Examples of the strings that must be replaced are: the name of the project; the package name of the application; the name of the database, and so on. The descriptors of the new application must also be extended to incorporate the CincoSecurity module.

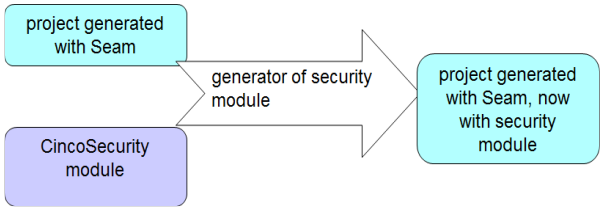

Figure 13. Generator of the security module, based on techniques of Regular expressions.

Figure 13 shows the process followed by the generator of the security module for an application previously generated with Seam.

Summarizing, the security module generator is composed by the tasks of text replacement, the tasks of copying the resulting files into the new application, and the tasks of extending the descriptors. These tasks must be expressed following the conventions of the tool used to process regular expressions.

D. *Automatic protection of new use cases by using the elements of CincoSecurity*

After incorporating the CincoSecurity module to an application, the Seam generator of CRUDs [14] can be used to generate the skeleton of new use cases. To facilitate the maintenance of the resulting application, this generation must be complement with a refactoring tool that reorganizes the generated code into subdirectories for modules, containing subdirectories for its use cases. This refactoring can be done with (Regular Expression) tasks that move the files to the appropriate directory, and (Regular Expression) tasks that fix the references in the web pages.

To take advantage of the security facilities provided by the CincoSecurity module (i.e., fine-grained roles and security profiles), the new use cases must be registered by the means of the use cases provided by CincoSecurity. After that, both the services of business components (EJBs) as the pages elements can be easily protected. A task, also based on Regular Expressions, provides:

- Registration of the new use case and each one of its services, associating each one with a fine-grained role.
- Registration of the menu item associated with the new use case.
- For each method of the EJB business component that controls the use case, an annotation with a security constraint is added, associating it with a fine-grained role.
- In the descriptors of the application, a security restriction for accessing the corresponding web pages is added, with the role of entering into the use case.

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

- A condition is added to each button on each web page of the use case, to make them visible only for users with the fine-grained role associated with the invoked service.

Thanks to the concept of fine-grained role and to the discipline of names proposed by CincoSecurity, the protection of the resources of the new application (use cases, services and page elements) can be added automatically. Subsequently, the administrator user can update the security profiles by selecting the services of old and new use cases (by the means of the use case of Management of security profiles, provided by CincoSecurity).

IX. COMPARISON WITH OTHER WORKS

There are other security modules proposed for the Java EE technology. Currently, in the SourceForge portal there is a dozen of software projects related with security for Java EE 5 applications, but most of them are proposals without implementation (i.e., they are in planning status). Two relevant projects with implementation and good acceptance from users are the following:

- *JPA Security* [20] is an Access Control Solution for the Java Persistence API (JPA) [11] with support for role-based access control, access control lists (ACLs) and domain-driven access control. Compared with CincoSecurity, this project does not offer access control to web pages as CincoSecurity does. JPA Security uses access rules in terms of database operations, which provides less flexibility than CincoSecurity, where security profiles are defined in terms of the services offered by the use cases of the application.
- *[fleXive]* [21] is a Java EE 5 open-source framework for the development of complex and evolving web applications. It offers an administration module that manages users and security. Some fleXive characteristics are a reliable data store backed by a relational database, native JavaEE 5+ support, complex data structures, fine-grained security, WebDAV and CMIS interfaces, and fully open source. It implements an access control list based approach, combined with roles; user accounts can be assigned to any number of user groups. Access control lists –which are assigned to user groups– define a list (Read, Edit, Create, etc.) of permissions attached to an arbitrary object. Compared with CincoSecurity, this project uses 10 coarse roles with predefined permissions related to the administration module, while CincoSecurity lets to define any number of security profiles, each one as a set of fine roles related to the services of the application (not only to the security services). We believe it is more intuitive to associate the users to these security profiles and not to the [fleXive] Access control lists (ACL), that define lists of permissions attached to arbitrary objects. [fleXive] does not offer access control to elements of web pages, as CincoSecurity does.

With respect to recent proposals for extending the RBAC model, some research works as [22] [23] try to statically validate the correctness of roles usage in an application, for solving what they call the fragility of traditional dynamic checks. In [23], Fisher et al. argument that traditional RBAC does not easily express application-level security requirements. For instance, in a medical records system it is difficult to express that doctors should only update the records of their own patients. Further, traditional RBAC frameworks rely solely on dynamic checks, which makes application code fragile and difficult to ensure correct. They introduce ORBAC, a generalized RBAC model allowing roles to be parameterized by properties of the business objects being manipulated, with static validation of a program's conformance to an ORBAC policy. Centoze et al. [22] present a theoretical foundation for correlating an operation-based RBAC policy with a data-based RBAC policy. They have built a static analysis tool for Java EE that analyzes bytecode to determine if the associated RBAC policy is location consistent, and reports potential security problems.

CincoSecurity does not implement static checks, but its strategy of fine grained roles enables to automate the correct incorporation of security in a web application. In effect, by following the names discipline explained in this paper, it is possible to automatically add annotations to each method of the session EJB3 controlling a use case, in order to permit its access only to users having the associated fine role; it is also possible to automatically modify button tags of JSF pages for rendering them only to users having the corresponding fine role. This automation has been explained in section VIII.

On the other hand, it is important to note that Seam offers a complete security module [14], that is based on (coarse) roles, permissions and rules, that achieves a very flexible control of resources. The CincoSecurity module takes advantage of the Seam security by using some of its facilities related with authentication, restriction annotations for roles, and tags of JSF pages for rendering only for the appropriate role. However, we believe that for an administrator it is more difficult to write rules for granting fine permissions to roles (as is done in the Seam module), than to configure security profiles by checking the services of the application to be granted (as is done in the CincoSecurity module). Also, given that CincoSecurity does not use permissions nor rules (only fine grained roles), the incorporation of security to a web application can be automated, as it was explained above; with the Seam module it seems more difficult to automate the incorporation of security.

X. CONCLUSION AND FUTURE WORK

Needless to say that developing a secure Java EE application is a difficult job, where dozens of subtle details must be handled coherently. The CincoSecurity module provides a complete code baseline to develop a Java EE application with the Seam framework, incorporating, from the beginning of the development process, a full and flexible access control to the use cases and services of the application being developed. The CincoSecurity module also provides the use cases required to administer the users and their access

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

permissions to the use cases and services of the application being developed.

With respect to the Core RBAC model [7], an access permission is materialized in CincoSecurity as the right to invoke a method (service) of a business component. Fine grained roles are defined and implemented, each one having just one permission to invoke a single method (service) of a business component (session EJB3). There is also a fine grained role to allow entrance to each use case, as well as a fine grained role to grant access to each one of the services provided by the use case. The concept of "security profile" is defined and implemented as a set of fine grained roles.

The CincoSecurity module takes advantage of the low level access control principle implemented by any Application Server, by feeding the Application Server with the fine grained roles included in the security profiles the authenticated user belongs to. Additionally, the CincoSecurity module dynamically builds a customized menu containing only the entries leading to the application's use cases authorized for the user.

The CincoSecurity module is a Java EE 5 application built using the Seam framework [12] [13]. It is distributed under the GPL license and can be freely downloaded from [18]. CincoSecurity is used by several software houses in Colombia.

CincoSecurity module may be modified or extended to incorporate more complex security policies (e.g., elaborated policies for password handling), permissions to access the different attributes of a persistent entity, or to implement extended RBAC models (hierarchical roles, constraints).

As future work, CincoSecurity will be extended to further automate and simplify the incorporation and administration of the security of a web application, as well as to include other capabilities of the Seam security module, like Identity management, in a compatible way with our approach.

REFERENCES

[1] M. C. Franky, V. M. Toro, "CincoSecurity: A Reusable Security Module Based on Fine Grained Roles and Security Profiles for Java EE Applications", ICIW 2011: The Sixth International Conference on Internet and Web Applications and Services, St. Maarten-The Netherlands Antilles, March 2011, pp. 118-123, ISBN: 978-1-61208-004-8

[2] R. L. Baldwin, "Naming and Grouping Privileges to Simplify Security Management in Large Databases", Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy (Oakland, CA), IEEE Computer Society Press, pp. 116-132, 1990.

[3] D. F. Ferraiolo and D. R. Kuhn, "Role-Based Access Control". Proc. 15th Nat'l Information Systems Security Conf., Diane Publishing Company, pp. 554–563, 1992.

[4] R. Sandhu, C. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models". IEEE Computer Magazine, pp. 38-47, 1996.

[5] American National Standard for Information Technology – Role Based Access Control, ANSI INCITS 359-2004, 2004.

[6] B. Messaoud, "Access Control Systems: Security, Identity Management and Trust Models", Springer Science+Business Media, Inc., 2006.

[7] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, "Role Based Access Control", Artech House 2003, 2nd Edition 2007.

[8] N. Li, J. W. Byun, and E. Bertino, "A Critique of the ANSI Standard on Role-Based Access Control", IEEE Security and Privacy, Volume 5, Issue 6, pp. 41-49, 2007.

[9] D. Alur, J. Crupi, and D. Malks, "Core J2EE Patterns: best practices and Design Strategies", Sun Microsystems - Prentice Hall, 2001.

[10] Oracle, "The Java EE 5 Tutorial", http://docs.oracle.com/javaee/5/tutorial/doc/ 01.23.2012

[11] M. Keith and M. Schincariol, "Pro EJB 3: Java Persistence API", Apress, 2006.

[12] M. Yuan and T. Heute, "JBoss Seam: Simplicity and Power Beyond Java EE", Prentice Hall, 2007.

[13] D. Allen, "Seam in Action", Manning Publications Co., 2009.

[14] JBoss Seam Group, "Reference manuals of JBoss Seam", http://seamframework.org 01.23.2012

[15] JBoss Community, "JBoss Application Server", http://www.jboss.org/jbossas 01.23.2012

[16] Eclipse Open source development platform comprised of extensible frameworks, tools and runtimes, http://www.eclipse.org 01.23.2012

[17] General Public License, http://www.gnu.org/licenses/gpl.html 01.23.2012

[18] M. C. Franky, V. M. Toro, and R. López, "CincoSecurity Module", http://sourceforge.net/projects/cincosecurity 01.23.2012

[19] M. C. Franky, V. M. Toro, and R. López, "CincoModule: Módulo de seguridad basado en roles finos y en perfiles de seguridad para aplicaciones Java EE 5". Quinto Congreso Colombiano de Computación (5CCC), Cartagena-Colombia, Abril 2010. ISBN: 978-958-8387-40-6.

[20] "JPA Security", http://jpasecurity.sourceforge.net 01.18.2011

[21] D. Lichtenberger, M. Plesser, G. Glos, J. Wernig-Pichler, H. Bacher, A. Zrzavy, and C. Blasnik, "[fleXive]™ 3.1 Reference Documentation", Copyright © 1999-2010 UCS - unique computing solutions gmbh, http://www.flexive.org/docs/3.1/xhtml/index.xhtml 23.01.2012

[22] P. Centonze, G. Naumovich, S. J. Fink, and Marco Pistoia, "Role-Based access control consistency validation", Proceedings of the 2006 international symposium on Software testing and analysis (ISSTA'06). ACM, New York, NY, USA, pp. 121-132, 2006

[23] J. Fischer, D. Marino, R. Majumdar, and T. Millstein, "Fine-Grained Access Control with Object-Sensitive Roles", ECOOP 2009 – Object-Oriented Programming, Lecture Notes in Computer Science, Volume 5653/2009, pp. 173-194, 2009.

[24] J. Friedl, "Mastering Regular Expressions". O'Reilly, 2002.

[25] M. Habibi, "Java Regular Expressions: Taming the java.util.regex Engine". Apress Publishing, 2004.

[26] The Apache Software Foundation, "Apache Ant™ 1.8.2 Manual", http://ant.apache.org/manual 01.23.2012

[27] M. Richards, R. Monson-Haefel, and D. A. Chappell, "Java Message Service", O'Reilly, 2009.

2011, © Copyright by authors, Published under agreement with IARIA - www.iaria.org