A FRAMEWORK FOR AUTHENTICATION AND AUTHORIZATION IN PLUG-IN-BASED CONTROL SYSTEM SOFTWARE

M. Clausen, J. Hatje, H. Rickens, DESY, Hamburg, Germany
J. Rathlev, University of Hamburg, Germany

Abstract

Preventing unauthorized use is a concern for many software systems, including control system software. The authorization mechanism used by a system should be pluggable, so that the software is not tied to a specific infrastructure. For the Control System Studio (CSS), we have developed a generic authorization framework which can be used by applications built on top of CSS to authorize user actions. For example, the framework provides support for the creation of menu items or graphical display elements that are automatically enabled and disabled based on the user's permissions. The framework is implemented in plug-ins which can be exchanged to interact with different infrastructures. Currently available implementations use standard Java authentication and authorization techniques to integrate with Kerberos and LDAP systems.

INTRODUCTION

Many applications require the ability to allow or deny access to certain functionality based on the permissions of the user. This is known as authentication and authorization. Authentication is the act of verifying the identity of the user, for example, based on the combination of the user name and a password known only to the user. Authorization includes granting permissions to a user, and checking whether the user has the required permissions when he uses some restricted functionality of an application.

To support authentication and authorization in Control System Studio (CSS) applications, we have developed an authorization framework which can be used by application plug-ins to implement permission-based access restrictions. The framework was originally developed as part of a diploma thesis at the University of Hamburg [1] and has since been integrated into the CSS platform.

In the following sections, we describe the architecture and functionality of the framework, how to use it, and the currently available implementations. In the section "Security Considerations", we describe how the framework fits into an overall security strategy.

AUTHORIZATION SUPPORT IN CONTROL SYSTEM STUDIO

The basic architecture of the framework is shown in Figure 1. Applications in the Control System Studio are implemented as plug-ins. These applications are built on top of the CSS Platform, which provides a common basis for their implementation. One of the features offered by

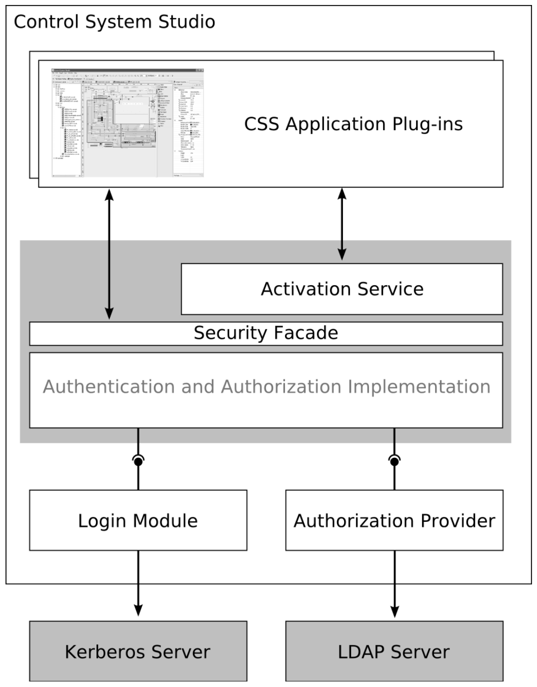

Figure 1: Architecture of the CSS authorization framework.

the CSS Platform is the authentication and authorization framework.

In its implementation, the framework relies on a login module and an authorization provider. The login module is responsible for performing the actual authentication of the user. The authorization provider is responsible for retrieving the list of permissions that are granted to the user and the permissions that are required for an action. Both the login module and the authorization provider are contributed to the framework as extensions via the Eclipse extension point mechanism. Thus, the login module and the authorization provider can be contributed by plug-ins, which can be exchanged in order to support integrating the CSS with different authorization infrastructures.

In their implementation, the login module and the authorization provider can connect to existing authentication and authorization services, for example, a Kerberos domain and an LDAP directory.

One of the main design goals of the CSS authorization framework was to support not only the activation and deactivation of certain actions based on the user's permissions, but also to integrate the authorization mechanism with the user interface. To achieve a high usability, users should be able to see which actions are available to them. For this purpose, the CSS Platform includes an Activation Service which can for example be used to automatically activate and deactivate UI widgets based on the current permissions of the user.

To identify actions that require permission, the framework uses simple strings called permission IDs. The advantage of this design is that it makes almost no assumptions about the organization of permissions, so it is very flexible. The downside is that because no hierarchical or other relationship of permission IDs is supported, the management of permissions may become complex if a large number of different permission settings have to be managed.

USING THE FRAMEWORK

There are two main entry points through which developers can use the framework, the Security Façade and the Activation Service.

The Security Façade is a simple, low-level API which can be used to directly check whether a specific permission is granted to the current user. The Security Façade also provides methods to register listeners which will be notified when the user's permissions change.

The Activation Service automatically activates and deactivates arbitrary objects based on the current user's permissions. It uses adapters to activate and deactivate the objects, so it can manage all types of objects for which an adapter is available. The adapter determines how activation and deactivation are implemented, so different adapters can be used to support different behaviours. For example, an adapter for menu actions could either disable or hide a menu item if the user does not have the required permission to use it.

The main purpose of the Activation Service is to automatically enable and disable user interface widgets. The CSS Platform provides adapters for Eclipse actions and SWT widgets, and the Synoptic Display Studio (SDS) provides an adapter for SDS widgets. Developers can implement additional adapters for their own objects.

In addition to the Security Façade and the Activation Service, the framework also includes some convenience classes that simplify its usage. For details, please see the CSS Platform's API documentation.

IMPLEMENTATIONS

DESY provides a login module implementation based on the Java Authentication and Authorization Service (JAAS) and an authorization provider based on LDAP.

The JAAS login module authenticates the user based on a standard JAAS configuration. At DESY, we use this module to authenticate users against a Kerberos server. The same module could also be used with other authentication services that support the JAAS standard.

To store the permission information, we use an LDAP directory, which is read by an LDAP Authorization Provider. We have designed a custom LDAP schema which is based on user groups and roles. Each user can belong to one or more groups with one or more roles, and each action can be permitted for a number of role-group combinations.

SECURITY CONSIDERATIONS

The CSS authorization framework is a client-side framework, that is, it performs authentication and authorization only within a single CSS application instance. It can be used to authenticate local users (and it can use a remote authentication service to do so), but it does not currently support the authentication of remote users.

Because the identity and permissions of remote users cannot be checked, the framework cannot be used on a server to prevent unauthorized access to the resources or services offered by the server. Such services must therefore be secured by other means. For example, access to an EPICS IOC can be restricted by using the Channel Access Security support. When such mechanisms are used, integration with the user interface on the client side must also be achieved by other means. For example, the Synoptic Display Studio integrates Channel Access Security information into the user interface by changing the mouse cursor depending on the user's Channel Access permissions.

Another important consideration is that because the login module and authorization provider are connected as extensions, a malicious user could circumvent the authorization by providing his own plug-ins, which simply claim that he is authorized to do everything. To prevent such attacks, additional security measures are required. For example, a system administrator could prevent users from installing their own plug-ins, or allow them to install only trusted, digitally signed plug-ins [1].

CONCLUSION

The CSS Platform provides an authorization framework which can be used by client applications that require authentication and authorization of users. The framework is based on plug-ins, which can be exchanged in order to integrate with different authentication and authorization infrastructures.

REFERENCES

[1] K. Meyer and T. Witte, „Benutzerautorisierung mit Anbindung an die Benutzungsoberfläche von Rich-Clients auf Basis der Eclipse RCP" (diploma thesis), University of Hamburg, Department of Informatics, 2007.