8.6 Security Context

Intent

Provide a container for security attributes and data relating to a particular execution context, process, operation, or action.

Also Known As

None known.

Motivation

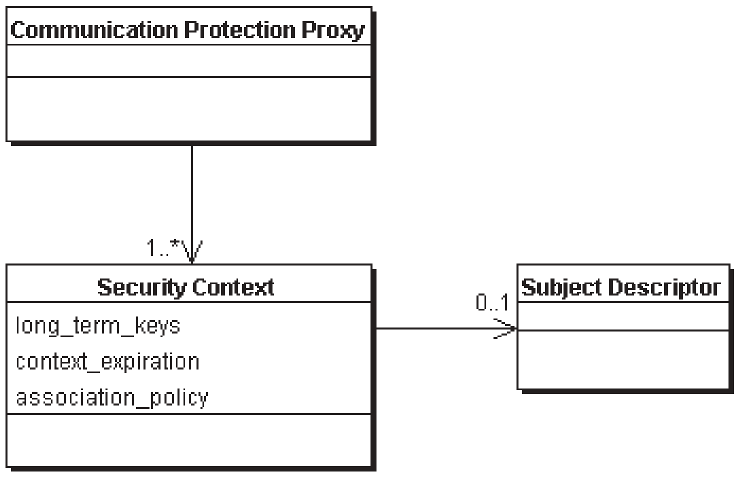
When a single execution context, program, or process needs to act on behalf of multiple subjects, the subjects need to be differentiated from one another, and information about each subject needs to be made available for use. When an execution context, program, or process needs to act on behalf of a single subject on multiple occasions over a period of time, it needs to be able to have access to information about the subject whenever it needs to take an action. The Security Context pattern provides access to subject information in these cases.

Applicability

Use the Security Context pattern when:

- A process or execution context acts on behalf of a single subject over time but needs to establish secure communications with a variety of different partners on behalf of this single subject.
- A process or execution context is able to act on behalf of different subjects and needs to manage which subject is currently active.

Structure



Participants

- Communication Protection Proxy

  Responsible for establishing Security Associations; used by Secure Communication to apply protection described in Security Association to messages.

- Security Context

  Stores information about a single subject, including secret attributes such as long-term keys to be used to establish Security Associations. A Communication Protection Proxy may create and retain several security contexts simultaneously, but it must always know which Security Context is active (that is, will be used to establish Security Associations).

- Subject Descriptor

  Stores the identity-related attributes of a subject.

Collaborations

Whenever a process becomes active in an execution context, the execution context's Communication Protection Proxy creates an instance of Security Context and populates it with the necessary information about the process. The execution context may perform some authentication challenge to verify the identity of the subject before creating a Security Context; the execution context may also set an expiration time for the Security Context to ensure that it is not re-used by a party other than the subject it refers to.

Consequences

Use of the Security Context pattern:

- Encapsulates security attributes relating to a process and user.

  Use of Security Context allows a user's security attributes, cryptographic keys, and process security attributes to be handled as a single object.

- Provides a point of access control.

  The Security Context will include attributes or accessors allowing callers to retrieve extremely sensitive information (such as long-term cryptographic keys belonging to the subject). This information must be protected against disclosure or misuse.

Implementation

As noted above, the Security Context implementation will need to protect the sensitive information contained within it.

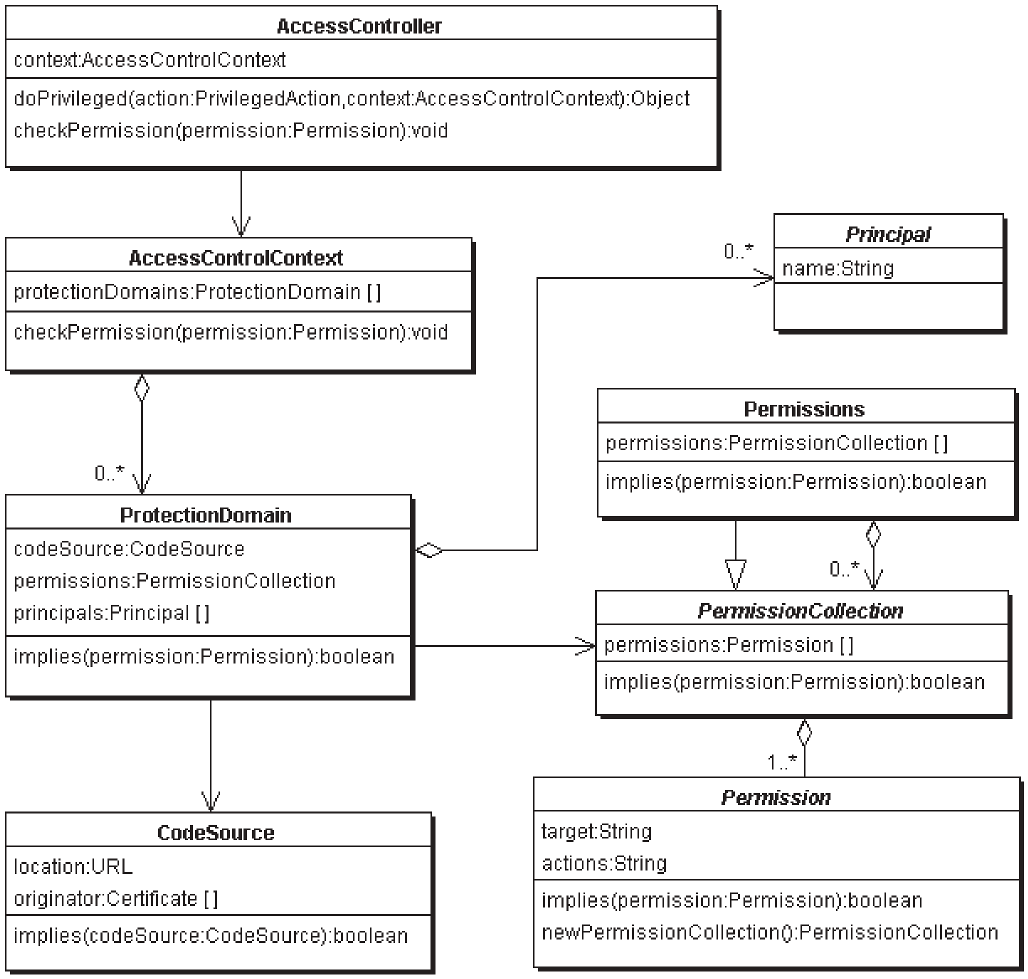
Access control can be implicit, if the system is architected such that only authorized callers can obtain a reference to a Security Context. If it is possible for unauthorized callers to discover references to Security Contexts, the implementation will need to provide accessors which check the authorization of the caller before returning sensitive information (see the Guard class in the Protected System pattern).

Known Uses

1. UNIX — Per-process User Information (‘‘u area’’)

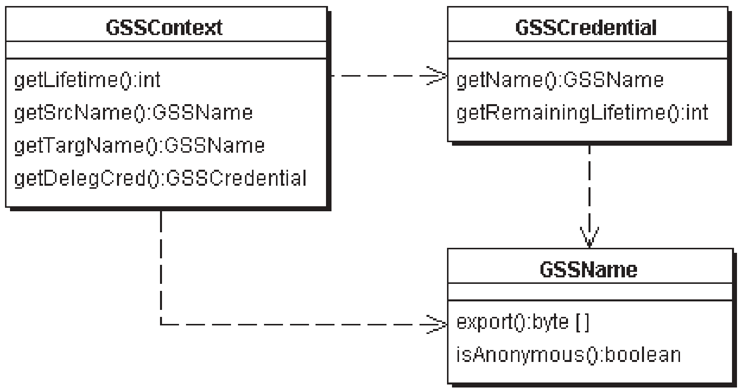
   The UNIX process table includes a ‘‘u area’’ which stores the identity of the logged-on user as well as the identity of an ‘‘effective user’’; the real user and the effective user are the same unless the user identity has been modified by executing a *setuid* operation. Retention of the real user ID allows switching back to the user’s original account after performing operations under the effective (*setuid*) identity.

2. Java 2 Standard Edition — java.security.AccessControlContext


AccessController
context:AccessControlContext
doPrivileged(action:PrivilegedAction,context:AccessControlContext):Object
checkPermission(permission:Permission):void
AccessControlContext
protectionDomains:ProtectionDomain []
checkPermission(permission:Permission):void
Principal
name:String
0..*
Permissions
permissions:PermissionCollection []
implies(permission:Permission):boolean
0..*
ProtectionDomain
codeSource:CodeSource
permissions:PermissionCollection
principals:Principal []
implies(permission:Permission):boolean
0..*
PermissionCollection
permissions:Permission []
implies(permission:Permission):boolean
1..*
CodeSource
location:URL
originator:Certificate []
implies(codeSource:CodeSource):boolean
Permission
target:String
actions:String
implies(permission:Permission):boolean
newPermissionCollection():PermissionCollection

The Java2 Access Control Context records the identity of the source of the executing code, together with the identity of the active user. The code source is recorded in a *ProtectionDomain* object, while the user identity is stored in a *Principal* object.

3. GSS-API — org.ietf.jgss.GSSContext



What GSS-API calls a ‘‘Security Context’’ is an instance of our Security Association pattern. The GSS-API structure which instantiates the Security Context pattern is the GSS Credential, which records the name and cryptographic key of the subject, together with an indication of whether the GSS Credential can be used to initiate outgoing GSS Security Contexts, or only to accept incoming GSS Security Contexts.

4. CORBA — SecurityLevel2::Current

CORBASecurity’s Current object (which represents an execution context) creates and stores three CORBA Credential objects; these objects are instances of Security Context; each Credential object contains information about a subject; the *InvocationCredential* object always refers to the active subject, and it is used by the Communications Protection Proxy (called a Security Interceptor) of the CORBA ORB (which is an instance of our Secure Communication pattern) to create CORBASecurity Context objects (which are instances of our Security Association pattern).

Related Patterns

Security Context uses Subject Descriptor [TG_SDP] to store identity-related information about subjects.

- Secure Association [TG_SDP] uses Security Context to store information about subjects and processes.