8.4 Reference Monitor

In a computational environment in which users or processes make requests for data or resources, this pattern enforces declared access restrictions when an active entity requests resources. It describes how to define an abstract process that intercepts all requests for resources and checks them for compliance with authorizations.

Also Known As

Policy Enforcement Point.

Example

In the hospital example described in ROLE-BASED ACCESS CONTROL (249) we declared the accesses allowed to doctors and other personnel. However, we expected voluntary compliance with the rules. It has not worked, busy personnel bypass the rules and there is no way of enforcing them.

Context

A computational environment in which users or processes make requests for data or resources.

Problem

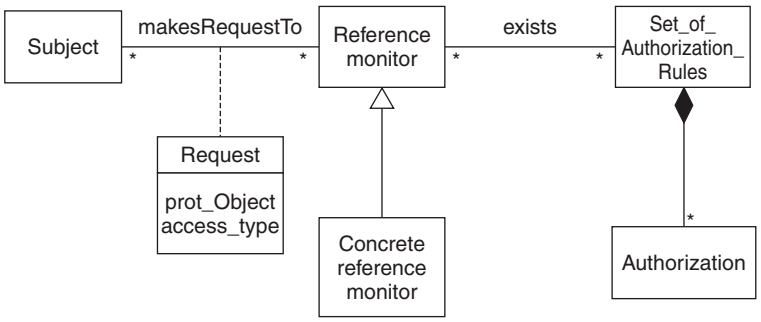
If we don't enforce the defined authorizations it is the same as not having them, users and processes can perform all type of illegal actions. Any user could read any file, for example.

The solution to this problem must resolve the following forces:

- Defining authorization rules is not enough, they must be enforced whenever a user or process makes a request for a resource.
- There are many possible ways of enforcement, depending on the specific architectural unit or level involved. We need an abstract model of enforcement that applies to every level of the system.

Solution

Define an abstract process that intercepts all requests for resources and checks them for compliance with authorizations.



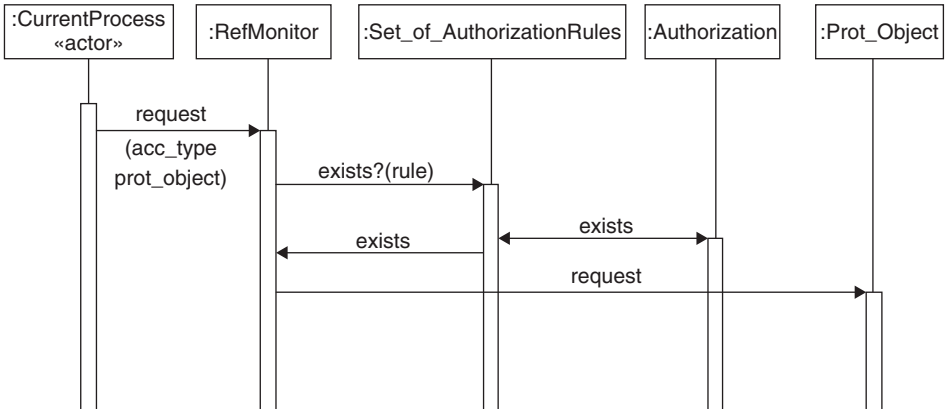
Class diagram for REFERENCE MONITOR (256)

Structure

The figure above shows a class diagram that describes a reified REFERENCE MONITOR (256). In this figure `Authorization Rules` denotes a collection of authorization rules organized as ACLs or in some other way.

Dynamics

The next figure is a sequence diagram showing how a request from a process is checked. The REFERENCE MONITOR (256) looks for the existence of a rule that authorizes the request. If one exists, the request is allowed to proceed.



Sequence diagram for enforcing security of requests

Implementation

A concrete reference monitor is required at each section of the system that has resources that can be requested. Examples include a memory manager (to control access to main memory), and a file manager (to control use of files).

Example Resolved

The hospital bought a database system to store patient data. Now, when a user attempts to access patient data, their authorization is checked before giving them access to it. Actions such as read or write are also controlled, for example, only doctors and nurses are allowed to modify patient records.

Known Uses

Most modern operating systems implement this concept, including Solaris 9, Windows 2000, AIX, and others. The Java Security Manager is another example. Database management systems also have an authorization system that controls access to data requested by queries.

Consequences

The following benefits may be expected from applying this pattern:

- If all requests are intercepted, we can make sure that they comply with the rules.
- Implementation has not been constrained by using this abstract process.

The following potential liabilities may arise from applying this pattern:

- Specific implementations (concrete REFERENCE MONITOR (256)s) are needed for each type of resource. For example, a file manager is needed to control requests for files.
- Checking each request may result in intolerable performance loss. We may need to perform some checks at compile-time, for example, and not repeat them at execution time.

See Also

This pattern is a special case of CHECK POINT (287). INTERCEPTOR [POSA2] can act as a REFERENCE MONITOR (256) in some situations. Concrete versions of REFERENCE MONITOR (256) include file control systems (Chapter 10) and firewalls (Chapter 12).