10.4 Controlled Object Monitor

This pattern addresses how to control access by a process to an object. Use a reference monitor to intercept access requests from processes. The reference monitor checks whether the process has the requested type of access to the object.

Example

Our operating system does not check all user requests to access resources such as files or memory areas. A hacker discovered that some accesses are not checked, and was able to steal customer information from our files. He also left a program that randomly overwrites memory areas and produces serious disruption to the other users.

Context

An operating system that consists of many users, objects that may contain sensitive data, and where we need to have controlled access to resources.

Problem

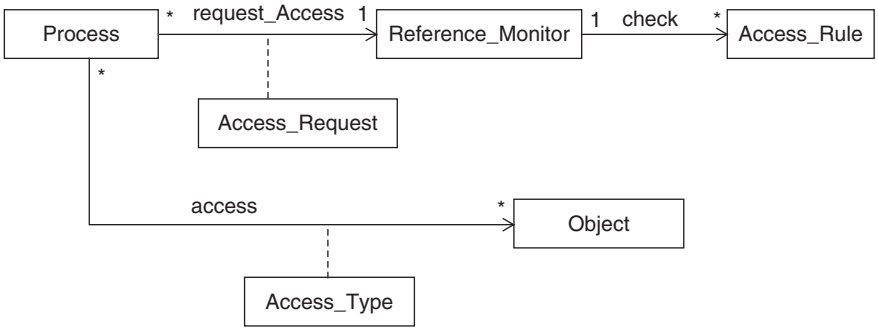
When objects are created we define the rights processes have to them. These authorization rules or policies must be enforced when a process attempts to access an object.

The solution to this problem must resolve the following forces:

- There may be many objects with different access restrictions defined by authorization rules: we need to enforce these restrictions when a process attempts to access an object
- We need to control different types of access, or the object may be misused

Solution

Use a REFERENCE MONITOR (256) to intercept access requests from processes. The REFERENCE MONITOR (256) checks whether the process has the requested type of access to the object according to some access rule.



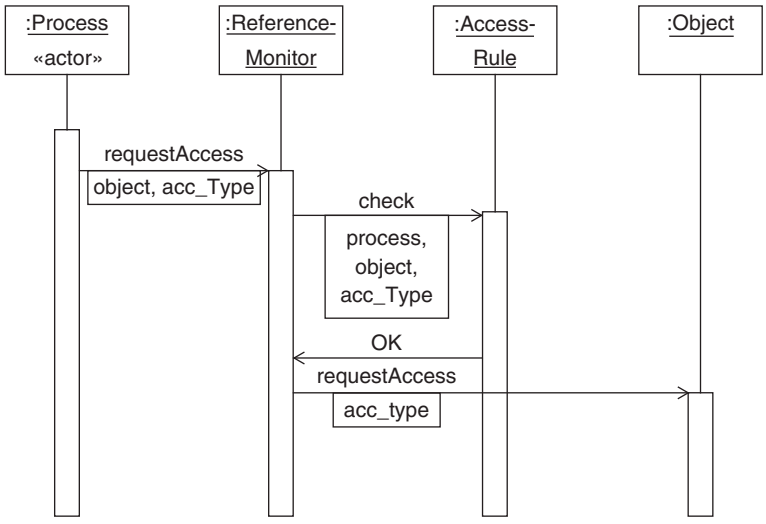
Class diagram for CONTROLLED OBJECT MONITOR

Structure

The figure above shows the class diagram for this pattern. This is a specific implementation of REFERENCE MONITOR (256). The modification shows how the system associates the rules to the secure object in question.

Dynamics

The next figure shows the dynamics of secure subject access to a secure object. Here the request is sent to the REFERENCE MONITOR (256) where it checks the `Access Rules`. If the access is allowed, it is performed and result returned to the subject. Note that here, a handle or ticket is returned to the `Subject` so that future access to the secure object can be directly performed without additional checking.



Sequence diagram for validating an access request

Example Resolved

A REFERENCE MONITOR (256) mediates all requests. There are now no unchecked requests, so a hacker cannot get access to unauthorized files or memory areas.

Known Uses

Windows NT. The Windows NT security subsystem provides security using the patterns described here. It has the following three components (see [Har01], [Kel97], and [Mic00]):

- Local Security Authority
- Security Account Manager
- Security Reference Monitor

The Local Security Authority (LSA) and Security Account Manager (SAM) work together to authenticate the user and create the user's access token. The security reference monitor runs in kernel mode and is responsible for the enforcement of access validation. When an access to an object is requested, a comparison is made between the file's security descriptor and the Secure ID (SID) information stored in the user's access token. The security descriptor is made up of Access Control Entries (ACE's) included in the object's Access Control List (ACL). When an object has an ACL the SRM checks each ACE in the ACL to determine if access is to be granted. After the Security Reference Monitor (SRM) grants access to the object, further access checks are not needed, as a handle to the object that allows further access is returned the first time.

Types of object permissions are no access, read, change, full control, and special access. For directory access, the following are added: list, add, and read.

Windows use the concept of a handle for access to protected objects within the system. Each object has a Security Descriptor (SD) that contains a Discretionary Access Control List (DACL) for the object. Each also process has a security token that contains an SID which identifies the process. This is used by the kernel to determine whether access is allowed. The ACL contains Access Control Entries (ACE's) that indicate what access is allowed for a particular process SID. The kernel scans the ACL for the rights corresponding to the requested access.

A process requests access to the object when it asks for a handle using, for example, a call to `CreateFile()`, which is used both to create a new file or open an existing file. When the file is created, a pointer to an SD is passed as a parameter. When an existing file is opened, the request parameters, in addition to the file handle, contain the desired access, such as `GENERIC_READ`. If the process has the desired rights for the access, the request succeeds and an access handle is returned, so that different handles to the same object may have different accesses [Har01].

Once the handle is obtained, additional access to read a file will not require further authorization. The handle may also be passed to another trusted function for further processing.

Java 1.2 Security. The Java security subsystem provides security using the patterns described here. The Java Access Controller builds access permissions based on permission and policy. It has a `checkPermission` method that determines the codesource object of each calling method and uses the current `Policy` object to determine the permission objects associated with it. Note that the `checkPermission` method will traverse the call stack to determine the access of all calling methods in the stack. The `java.policy` file is used by the security manager that contains the grant statements for each codesource.

Consequences

The following benefits may be expected from applying this pattern:

- Each access request can be intercepted and accepted or rejected depending on the authorization rules.
- The access rules can implement an access matrix defining different types of access for each subject. We can add content-dependent rules if required.

The following potential liabilities may arise from applying this pattern:

- There is a need to protect the authorization rules. However, the same mechanism that protects resources can also protect the rules.
- There is an overhead involved in controlling each access. This is specially heavy for content-dependent rules. However, some accesses may be compiled for efficiency.

See Also

The REFERENCE MONITOR (256) is the pattern from which this pattern is derived.