10.7 Controlled Execution Environment

If a process execution environment is uncontrolled, processes can scavenge information by searching memory and accessing the disk drives where files reside. They might also take control of the operating system itself, in which case they have access to everything. Use AUTHORIZATION (245) to define the rights of a subject. From these rights we can set up the rights of processes running on behalf of the subject. Process requests are validated by CONTROLLED OBJECT MONITOR (335) or REFERENCE MONITOR (256) respectively.

Example

Jim the hacker discovers that the customer's files have authorizations and cannot be accessed directly, so he tries another approach. He realizes that processes are not given only the rights of their owners, but also have rights with which they can access memory and other resources belonging to other users. He systematically searches areas of memory and I/O devices being used by other processes until he can scavenge a few credit card numbers that he can use in his illicit activities.

Context

A process executes on behalf of a user or role (a subject). A process must have access rights to use these resources during execution. The set of access rights given to a process define its execution domain. Processes must be able to share resources in a controlled way. The rights of the process are derived from the rights of its invoker.

Problem

Even if direct access to files is restricted, users can do 'tunneling,' attacking them through a lower level. If the process execution environment is uncontrolled, processes can scavenge information by searching memory and accessing disk drives. They might also take control of the operating system itself, in which case they have access to everything.

The solution to this problem must resolve the following forces:

- We need to constrain the execution of processes and restrict them to use only resources that have been authorized based on the rights of the activator of the process.
- Subjects can be users, roles, or groups. We want to deal with them uniformly.

- Resources typically include memory and I/O devices, but can also be files and special instructions. We want to consider them in a uniform way.
- A subject may need to activate several processes, and a process may need to create multiple domains. Execution domains may need to be nested. We want flexibility for our processes.
- Typically, only a subset of a subject's rights needs to be used in a specific execution. We need to provide to a process only the rights it needs during its execution (on the principle of least privileges).
- The solution should put no constraints on implementation.

Solution

Use the AUTHORIZATION (245) pattern to define the rights of a subject. From these rights, we can set up the rights of processes running on behalf of the subject. Process requests are validated by CONTROLLED OBJECT MONITOR (335) or REFERENCE MONITOR (256) respectively.

Structure

The figure on page 348 shows the UML class diagram of CONTROLLED EXECUTION ENVIRONMENT (346). This model combines AUTHORIZATION (245), EXECUTION DOMAIN (343), and REFERENCE MONITOR (256) to let processes operate in an environment with controlled actions based on the rights of their invoker. Process execution follows EXECUTION DOMAIN (343)—as a process executes it creates one or more domains. Domains can be recursively composed. The descriptors used in the process' domains are a subset of the authorizations that the subject has for some `ProtectionObjects` (defined by an instance of AUTHORIZATION (245)). `ProtectionObject` is a superclass of the abstract `Resource` class, and `ConcreteResource` defines a specific resource. Process requests go through a `ReferenceMonitor` that can check the domain descriptors for compliance.

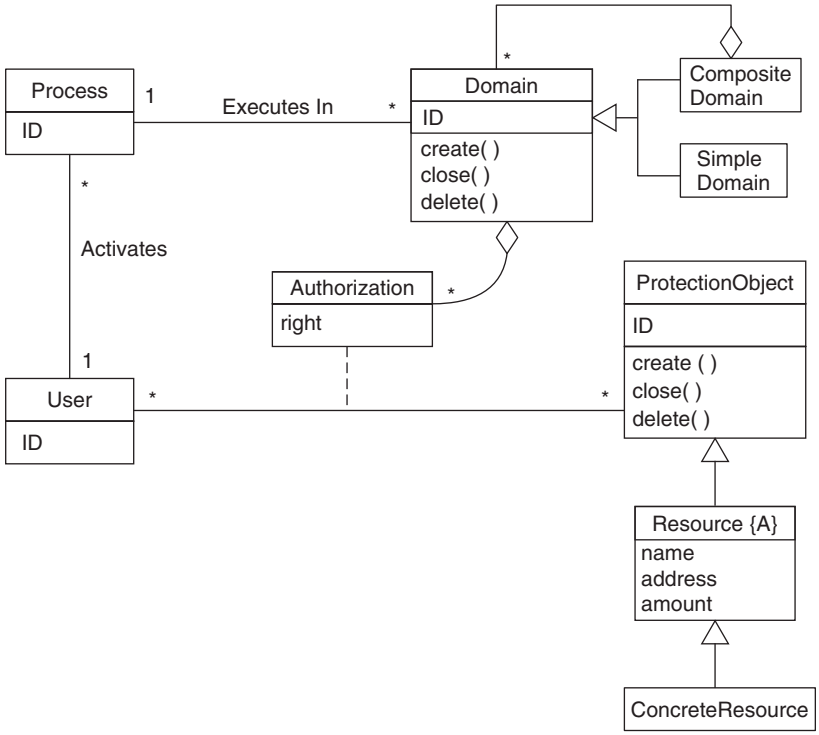
Dynamics

The figure on page 348 shows a sequence diagram representing the use of a right after entering a domain. Here x denotes a segment requested by the process. An instance of REFERENCE MONITOR (256) controls the process requests. This diagram assumes that the descriptors of the domain have been previously set up.

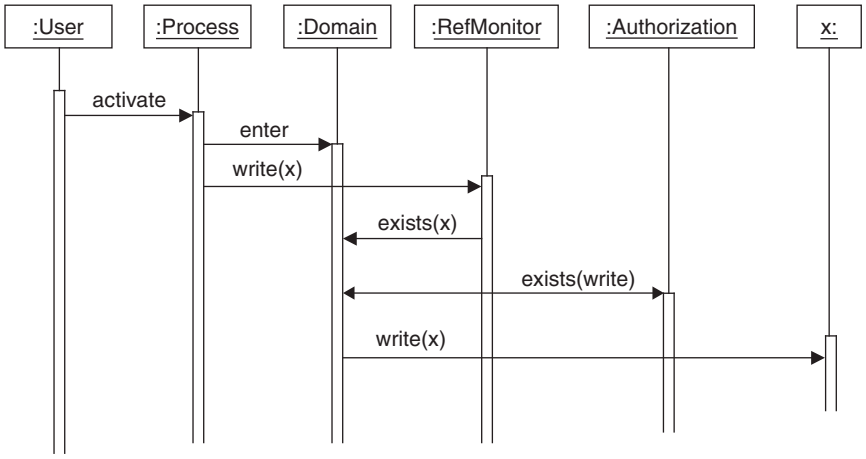
Example Resolved

A new operating system was installed, with mechanisms to make processes operate with the rights of their activator. Jim does not have access to customer files, which

makes his processes also unable to access these files. Now he cannot scavenge in other users' areas, so his illicit actions are thwarted.



Class diagram for CONTROLLED EXECUTION ENVIRONMENT



Sequence diagram for entering a domain and using a right in that domain

Known Uses

The IBM S/38, the IBM S/6000 running AIX, the Plessey 250 [Ham73], and EROS [Sha02] have applied this pattern using capabilities. Property descriptor systems such as the Intel architectures may use this approach although their operating systems not always do so.

Consequences

The following benefits may be expected from applying this pattern:

- We can apply the principle of least privileges to processes based on the rights of their activators. This also provides accountability.
- It can be applied to any type of resource.
- Subjects may activate any number of processes, and processes may have several execution domains.
- The same structure can also provide fault tolerance [Ham73].
- Execution domains are defined according to DOMAIN, and may include any subset of the subject's rights.

The following potential liabilities may arise from applying this pattern:

- Some extra complexity and performance overhead may be required.
- It can be dependent on the hardware architecture.

See Also

This pattern uses AUTHORIZATION (245), EXECUTION DOMAIN (343), and REFERENCE MONITOR (256). CONTROLLED VIRTUAL ADDRESS SPACE (339) pattern may be indirectly used by EXECUTION DOMAIN (343).