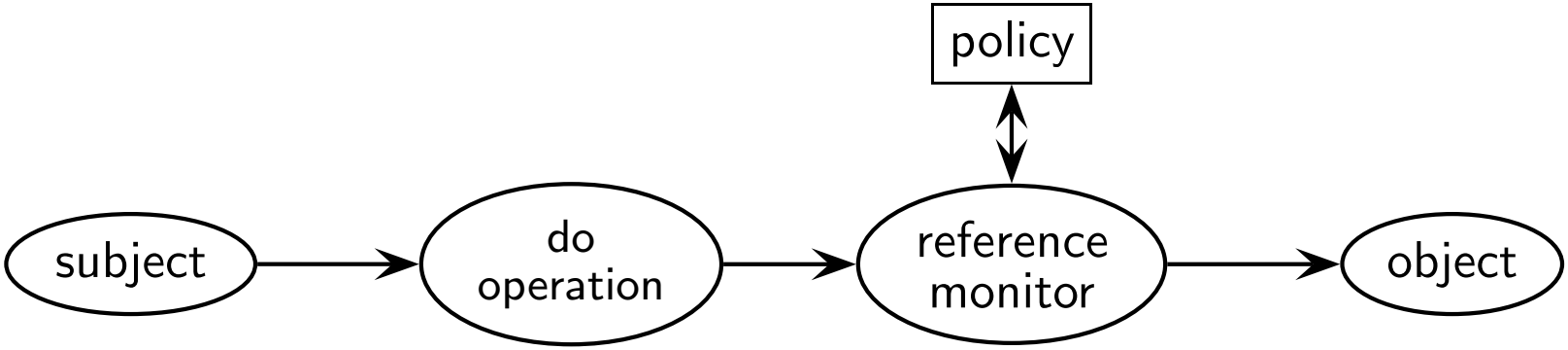
Access Control

Simon Foley

February 2, 2016

The Reference Monitor



Reference monitor: conceptualization of protection mechanism.

- objects: the set of protected entities that have state, for example, directories, files, memory segments, ...
- subjects: the set of active objects, for example, processes, ...
- protection policy: a set of rules that define the operations that a subject may carry out (do) on an object.

A reference validation mechanism (RVM) is an implementation of a reference monitor. It must be tamper-proof, cannot be bypassed and be the subject of analysis and testing for completeness.

Reference Validation Mechanism

The reference monitor must mediate every request by a subject to carry out an operation on an object.

Reference monitors can operate at different levels of granularity, for example,

- Security Kernel mediating low-level machine/OS instructions.
- DBMS access control mediating access/queries.
- Java2 access control mediating method-object calls.
- Application system mediating transactions.
- Firewall providing host-based access control on packets.
- ...

Trusted Computing Base is the totality of protection mechanisms within a computer system—including hardware, firmware, and software—the combination of which is responsible for enforcing protection policies.

The Access Matrix Model

Abstract interpretation for protection policy defined in terms of: set of subjects S, set of objects O, permissions P, and a matrix M (current access state), where $M[s, o]$ gives the permissions that subject s holds on object o.

Example, $O = \{\text{File1}, \text{File2}, \text{InetSocket}, \text{ProcAlice}, \text{ProcBob}\}$, $S = \{\text{ProcBob}, \text{ProcAlice}\}$, $P = \{\text{read}, \text{write}\}$, and

M	File1	File2	InetSocket	ProcAlice	ProcBob
ProcAlice	read	write			
ProcBob	read write	read	write		

In this case, $M[\text{ProcAlice}, \text{File1}] = \{\text{read}\}$ means that the Alice process may 'do' the action (permission) read on File1.

If permission p is not in cell $M[s, o]$ then subject s may not do the corresponding action on object o.

Matrix operations define how how the accesses are allowed change. For example, Unix `chmod` changes permissions users have to files.

The Access Matrix Model

Permissions defined for *any* kind of operation, not just `read`, `write`, `execute`. For example, permissions push, pop, etc., for a stack object.

The access matrix model is used to understand the meaning of access control in theory. It has been used to answer a number of fundamental questions about protection.

- Modeling protection using the access matrix model is equivalent to a Turing machine and therefore any kind of protection policy to be implemented by a computer can be represented in terms of the Matrix model.
- *Safety Problem*: Determining whether, starting at current state, a subject could access an object in some future access state is, in general, undecidable (equivalent to the halting problem). This assumes that the policy is itself an object(s) and may be accessed/changed by subjects in a controlled way.

In practice, we don't use a matrix to *implement* policies; it was originally developed to explore questions such as the above.

Policy Implementation Example: Unix Permissions

Every user has a unique user identifier. Distinguish access rights of file owner from access rights of others. The owner of a file may decide its access right permissions.

Example. User simon owns the file exam and does:
`> chmod u=rw exam`

owner	other	
`rw`	`--`	exam(owner=simon)

Only the owner of this file may have read/write access.

User simon writes an assignment, with text in file assn:
`> chmod u=rw,o=r assn`

owner	other	
`rw`	`r`	assn(owner=simon)

Owner may read/write access, everybody else may read.

Interpret this in the access-control matrix model.

Policy Implementation Example: Unix Permissions

Unix also organizes users by group and distinguishes group access rights from owner and other access rights.

Users may be members of one or more groups.

Groups and membership configured by the security administrator (root).

The owner of a file may configure its access right permissions.

Example. Introduce groups CS4615 and staff. User simon is in both groups. Student Alice is in group CS4615.

```
> chmod u=rw,g=r test
```

owner	group	other
rw	r	--

test(owner=simon; group=CS4615)

Alice may read the test but not modify it. Student Bob, who is not in group CS4615 may not access the file.

Policy Implementation Example: Access Control Lists

Associate an Access Control List (ACL) with each object

- ACL gives details about who may access (and how) the object.
- ACLs may be modified by the owner; more flexible than groups.
- ACL checked by protection mechanism before access is granted

Example. Some versions of unix support ACLs (POSIX P1003.6).

```
> getacl test
# file:  test
# owner:  simon
# group:  CS4615
#
user::rw-
group::r--
other::r--
```

simon grants tutor tony read access to the test file

```
> setacl -u user:tony:r-- test
> getacl test
# file:  test
# owner:  simon
# group:  CS4615
#
user::rw-
user:tony:r--
group::r--
other::r--
```

Interpret ACLs as columns in the Access Matrix Model.

Policy Implementation Example: Capabilities

Capability is an unforgeable token that specifies subject access rights.

Each subject owns a collection of capabilities (capability list).
Must present valid capability before access granted by mechanism.

Example. In an OS kernel, each process has a Segment descriptor table that provides pointers (capabilities) to segments/pages of virtual memory. HW memory protection ensures that memory may only be accessed via this table

Example. A web-browser presents an authenticator cookie to a web-site in order to gain access to a particular web-page. Recall that the authenticator cookie is computed as $C = h_k(\textit{userid}, \textit{path}, \ldots)$ where $h_k()$ is a keyed one-way hash function with key k known only to web-sever. Cookie C is a software capability that cannot be forged (but it can be copied).

How might you interpret capabilities in the Access Matrix Model?