Security Models

Computer Security Lecture 9

David Aspinall

School of Informatics
University of Edinburgh

14th February 2013

Outline

System security policies and models

A **security policy** describes requirements for a system.

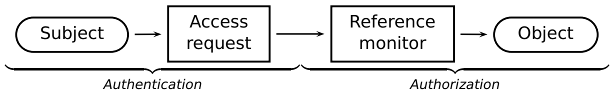
A **security model** is a framework in which a policy can be described.

There are two basic paradigms:

- **access control**
- **information flow control**

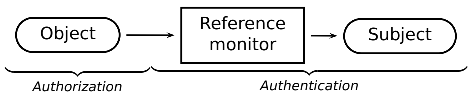
Access control

A guard controls whether a principal (the **subject**) is allowed access to a resource (the **object**).



Information flow control

A guard controls whether information may flow from a resource (the object) to a principal (the subject).



This is the dual notion, sometimes used when confidentiality is the primary concern.

Access operations: modes and rights

- To define types of access, we define some fundamental **access modes** and **access rights**.
- Modes are ways of accessing objects; rights are combinations of modes.
- Access rights are the model's level of granularity for defining security policy. Each real operation requires particular access rights.
- We will consider the access modes and rights of the influential **Bell-LaPadula** (BLP) model.
  - BLP enforces confidentiality
  - Other models enforce integrity, or a combination

Access operations in BLP

The **access modes** of BLP are:

observe	examine contents of an object
alter	change contents of an object

The **access rights** and their profiles are:

	observe	alter
exec		
read	✓	
append		✓
write	✓	✓

Profiles and names of rights differ between systems, or even for different subject kinds. E.g., sometimes have a delete. In Unix, exec for directories indicates ability to read the directory. The profiles of rights are used to define security properties in the model.

Who sets the policy?

Discretionary Access Control (DAC)

The **owner** of a resource decides who may access that resource. Policy is set on a case-by-case basis.

Mandatory Access Control (MAC)

The decision for accessing resources is controlled **system-wide** by a uniform policy.

- In practice a mixture of DAC and MAC may apply.

Ownership and identity

- Owners of resources may be principals in the system: *subjects* themselves under access control.
- BLP does not (directly) consider operations to modify access controls (e.g., chown in Windows), nor explain when such operations are safe.
- The **identity** of subjects is also flexible: e.g., *identity changes* during operations (SUID programs in Unix).
- Again, this doesn't fit BLP.

Access control structures

- How are access control rights defined? Many schemes, but ultimately modelled by:
  - A set S of subjects, a set O of objects
  - A set A of operations (modelled by access rights), we'll consider $A = \{\text{exec}, \text{read}, \text{append}, \text{write}\}$.
  - An **access control matrix**

$$M = (M_{so})_{s \in S, o \in O}$$

  where each entry $M_{so} \subseteq A$ defines rights for s to access o.
- Example matrix for $S = \{\text{Alice}, \text{Bob}\}$ and three objects:

	`bob.doc`	`edit.exe`	`fun.com`
Alice	{}	{exec}	{exec, read}
Bob	{read, write}	{exec}	{exec, read, write}

Representing the access control matrix

- Implementing M directly is impractical, so different schemes are used. Complementary possibilities: either use **capabilities** (store M by rows) or use **access control lists** (store M by columns)
- A **capability** is an unforgeable token that specifies a subject's access rights. Pros: can pass around capabilities; good fit with discr. AC. Cons: difficult to revoke, or find out who has, access to a particular resource (must examine all capabilities). Interest reinstated recently with distributed and mobile computation.
- An **access control list** (ACL) stores the access rights to an object with the object itself. Pros: good fit with object-biased OSes. Cons: difficult to revoke, or find out, permissions of a particular subject (must search all ACLs).

Multi-level security

- **Multi Level Security** (MLS) systems originated in the military. A **security level** is a label for subjects and objects, to describe a policy.
- Security levels are ordered:

$$\text{unclassified} \leq \text{confidential} \leq \text{secret} \leq \text{topsecret}.$$

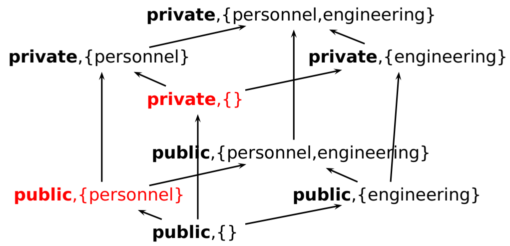
- Ordering can express policies like "no write-down" which means that a high-level subject cannot write down to a low-level object. (A user with confidential clearance cannot write an unclassified file: it might contain confidential information read earlier.)
- In practice, we need more flexibility. We may want *categorizations* as well, for example, describing departments or divisions in an organization. Then individual levels may not be comparable...

Security lattices

- A *lattice* is a set L equipped with a partial ordering $\leq$ such every two elements $a, b \in L$ has a *least upper bound* $a \vee b$ and a *greatest lower bound* $a \wedge b$. A finite lattice must have top and bottom elements.
- In security, if $a \leq b$, we say that b **dominates** a.
  - system low is the bottom, dominated by all others.
  - system high is the top, which dominates all others.
- Lattices are useful for MLS policies because:
  - for two *objects* at levels a and b, there is a minimal security level $a \vee b$ for a *subject* to access both;
  - for two *subjects* at levels a and b, there is a maximal security level $a \wedge b$ for an *object* which must be readable by both.

A Lattice Construction [Gollmann]

- take a set of *classifications* H and linear ordering $\leq_H$
- take a set C of categories; *compartments* are subsets of C
- *security levels* are pairs (h, c) with $h \in H$ and $c \subseteq C$
- ordering $(h_1, c_1) \leq (h_2, c_2) \iff h_1 \leq h_2, c_1 \subseteq c_2$ gives a lattice.



Bell-LaPadula Model (BLP)

- BLP (1973) is state machine model for confidentiality.
- Permissions use an AC matrix and **security levels**. The **security policy** prevents information flowing from a high level to a lower level.
- Assume subjects S, objects O, accesses A as before.
- A set L of security levels, with a partial ordering $\leq$.
- The state set $\mathcal{B} \times \mathcal{M} \times \mathcal{F}$ captures the current permissions and subjects accessing objects. It has three parts:
  - $\mathcal{B}$ possible current accesses
  - $\mathcal{M}$ permissions matrices
  - $\mathcal{F}$ security level assignments
- A BLP state is a triple (b, M, f).

BLP state set

- $\mathcal{B} = \mathcal{P}(S \times O \times A)$ is the set of all possible current accesses.
  An element $b \in B$ is a set of tuples (s, o, a) meaning s is performing operation a on an object o.
- $\mathcal{M}$ is the set of permission matrices $M = (M_{so})_{s \in S, o \in O}$.
- $\mathcal{F} \subset L^S \times L^S \times L^O$ is the set of security level assignments.
  An element $f \in \mathcal{F}$ is a triple (f_S, f_C, f_O) where
  - $f_S : S \to L$ gives the **maximal security level** each subject can have;
  - $f_C : S \to L$ gives the **current security level** of each subject (st $f_C \leq f_S$), and
  - $f_O : O \to L$ gives the **classification** of all objects.

BLP Mandatory Access Control Policy

Consider a state (b, M, f), where b is the set of current accesses.

Simple security property

The **ss-property** states for each access $(s, o, a) \in b$ where $a \in \{\text{read}, \text{write}\}$, then $f_O(o) \leq f_S(s)$ (*no read-up*).

Star property

The $*$**-property** states for each access $(s, o, a) \in b$ where $a \in \{\text{append}, \text{write}\}$, then $f_C(s) \leq f_O(o)$ (*no write-down*) and moreover, we must have $f_O(o') \leq f_O(o)$ for all o' with $(s, o', a') \in b$ and $a' \in \{\text{read}, \text{write}\}$ (*o must dominate any other object s can read*).

Together these form the *mandatory access control* policy for BLP.

BLP Discretionary Control and Security

The access control matrix M allows DAC as well.

Discretionary security property

The **ds-property**: for each access $(s, o, a) \in b$, we have that $a \in M_{so}$ (*discretionary access controls are obeyed*).

- **Definition of Security:** The state (b, M, f) is **secure** if the three properties above are satisfied.

Notice that BLP's notion of security is entirely captured in the current state.

Current clearance level

- Unfortunately, the $*$-property means a high-level subject cannot send messages to a low-level subject. This is unrealistic!
- There are two ways out:
  1. temporarily downgrade a high-level subject, which is why the model includes the current **clearance level** setting f_C, or
  2. identify a set of **trusted subjects** allowed to violate the $*$-property.
- Approach 1 works because the current state describes exactly what each subject knows. So if a subject (e.g. a process) is downgraded, it cannot access higher-level material, so may safely write at any lower level than its maximum.
- When subjects are people with high-level clearances, approach 2 works: we trust someone to violate the property in the model, e.g., by publishing part of a secret document.

Basic security theorem

- A transition from state v_1 to v_2 is **secure** simply if both states v_1 and v_2 are secure.
- This leads to a rather simple and general theorem:

Basic security theorem

If all state transitions in a system are secure and the initial state of the system is secure, then every subsequent state is also secure.

(NB: this follows immediately by induction, it has nothing to do with the properties of BLP!)

- The point: we can reduce checking the system for all possible inputs to checking that each kind of possible state transition preserves security. Of course, to do this we need a concrete instance of the model which describes possible transitions.

Summary

- A **security model** is a framework for formalising **security policies**
- Access control enforcement uses a **reference monitor**
- Operations have access modes used to define properties
- **Bell-LaPadula** (BLP) access control model:
  - For confidentiality
  - Discretionary (DAC) and mandatory (MAC) access
  - MAC via multi-level security lattice
  - ss-property: no read-up
  - $*$-property: no write down, direct or indirect
  - DAC via access control matrix (ds-property)

References

See Chapters 5, 11 (also 7 and 8) of Gollmann, and Parts 2–3 of Bishop.

- Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.*. Wiley & Sons, 2nd Edition, 2008.
- Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2003.
- Dieter Gollmann. *Computer Security*. John Wiley & Sons, 3rd Edition, 2011.

Recommended Reading

Chapters 5 and 11 of Gollmann.
Chapters 4 and 8 of Anderson.