Access Control

KAMI VANIEA
Today

• Shellshock demo
• Access and information flow
• Access control mechanisms
• Multi-level security
• The BLP security model
Shellshock Demo
Access and information flow
System security policies and models

• A security policy describes requirements for a system.
• A security model is a framework with 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 a principal (the subject) is allowed access to a resource (the object).

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

- Access control
  - Starts with the subject (user) and asks if the user has access to the object.

- Information flow control
  - Starts with the object (information) and asks if that information can be known to the subject.
Access operations: modes and rights

• To define types of access, we define some fundamental **access modes** and **access rights**
• Modes are a way 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 access modes and rights of the influential Bell-LaPadula (BLP) model
  ◦ BLP enforces confidentiality
  ◦ Other models enforce integrity, or a combination
  ◦ Access management almost never considers availability
Access operations in BLP

- The access modes of BLP are:
  - **Observe**: examine contents of an object
  - **Alter**: change the contents of an object

- The access rights and their profiles are:

<table>
<thead>
<tr>
<th></th>
<th>Observe</th>
<th>Alter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Append</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Write</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

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 profile of rights 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 set on a case-by-case basis.

- Mandatory Access Control (MAC)
  - The decision of 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 principles in the system: subjects themselves under access control.
- BLP does not (directly) consider operations to modify access controls (e.g., chown in Linux), 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 mechanisms
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 (modeled by access rights), we consider $A=\{\text{exec, read, append, write}\}$
- 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 $S=\{\text{Alice, Bob}\}$ and three objects

<table>
<thead>
<tr>
<th></th>
<th>Bob.doc</th>
<th>Edit.exe</th>
<th>Fun.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>{}</td>
<td>{exec}</td>
<td>{exec, read}</td>
</tr>
<tr>
<td>Bob</td>
<td>{read, write}</td>
<td>{exec}</td>
<td>{exec, read, write}</td>
</tr>
</tbody>
</table>
Representing the access control matrix

- Implementing $M$ directly is impractical, so different schemes are used. Complimentary 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
  - Cons: difficult to revoke or find out who has access to a particular resources (you have to examine all capabilities)
- 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, all permissions for a particular subject
alice VM from the exercise

Three access rights
- r – read
- w – write
- x – exec

Three subjects:
- owner
- group
- world/other

- Access control list for each file and folder in the /var/www folder
- index.php can be read and written to by alice, anyone on the computer can read it
Multi-level security
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:

  unclassified ≤ confidential ≤ secret ≤ 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* $a \lor b$ and a *greatest lower bound* $a \land b$. 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 $a \lor b$ for a *subject* to access both;
  - for two *subjects* at levels $a$ and $b$, there is a maximal security level $a \land b$ 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.
Running Example

Classifications (H)
- Admin
- Manager
- User

Categories (C)
- H (Hippo project)
- W (Walrus project)
Orderings:
(User, {}) ≤ (User, {W})
(User, {}) ≤ (User, {H})
Orderings:
(User, {}) ≤ (User, {W})
(User, {}) ≤ (User, {H})
(User, {W}) ≤ (User, {H, W})
(User, {H}) ≤ (User, {H, W})
Orderings:

(User, {}) ≤ (User, {W})
(User, {}) ≤ (User, {H})
(User, {W}) ≤ (User, {H, W})
(User, {H}) ≤ (User, {H, W})
(User, {}) ≤ (Manager, {}) 
(User, {}) ≤ (Manager, {H, W}) 
(User, {H, W}) ≤ (Manager, {H, W})

Admin, {} ≤ Admin, {H, W} ≤ Admin, {W} ≤ Admin, {H}

Manager, {} ≤ Manager, {H, W} ≤ Manager, {W} ≤ Manager, {H}

User, {} ≤ User, {W} ≤ User, {H, W} ≤ User, {H} ≤ User, {}
Orderings:
(User, {}) ≤ (User, {W})
(User, {}) ≤ (User, {H})
(User, {W}) ≤ (User, {H, W})
(User, {H}) ≤ (User, {H, W})
(User, {}) ≤ (Manager, {}) 
(User, {H, W}) ≤ (Manager, {H, W})
(User, {W}) ≤ (Manager, {W})
(User, {H}) ≤ (Manager, {H})
Orderings:

(Manager, {}) ≤ (Manager, {W})
(Manager, {}) ≤ (Manager, {H})
(Manager, {W}) ≤ (Manager, {H, W})
(Manager, {H}) ≤ (Manager, {H, W})
(Manager, {}) ≤ (Admin, {H})
(Manager, {H, W}) ≤ (Admin, {H, W})
(Manager, {W}) ≤ (Admin, {W})
(Manager, {H}) ≤ (Admin, {H})
Orderings:
(Admin, {}) ≤ (Admin, {W})
(Admin, {}) ≤ (Admin, {H})
(Admin, {W}) ≤ (Admin, {H, W})
(Admin, {H}) ≤ (Admin, {H, W})
Outline

Access and information flow

Access control mechanisms

Multi-level security

The BLP security model

Summary
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 $B \times M \times F$ captures the current permissions and subjects accessing objects. It has three parts:
  - $B$ possible current accesses
  - $M$ permissions matrices
  - $F$ security level assignments
- A BLP state is a triple $(b, M, f)$. 
BLP state set

- $B = \mathcal{P}(S \times O \times A)$ 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 $o$.
- $M$ is the set of permission matrices $M = (M_{so})_{s \in S, o \in O}$.
- $F \subset L^S \times L^S \times L^O$ is the set of security level assignments.
  An element $f \in F$ is a triple $(f_S, f_C, f_O)$ where
    - $f_S: S \rightarrow L$ gives the maximal security level each subject can have;
    - $f_C: S \rightarrow L$ gives the current security level of each subject (st $f_C \leq f_S$), and
    - $f_O: O \rightarrow L$ gives the classification of all objects.
BLP Model – B

- $b$ is the set of all possible current accesses.
- An element of $b$ is a set of tuples (subject, action, object)

\[
b = [(Alice, FileA, write),
     (Alice, FileA, read),
     (Alice, FileA, exec),
     (Bob, FileA, write),
     (Bob, FileA, read),
     (Charlie, FileA, read)]
\]
BLP Model – M

- M is the set of permission matrices

<table>
<thead>
<tr>
<th></th>
<th>FileA</th>
<th>FileB</th>
<th>FileC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>{read, write, exec}</td>
<td>{read, write}</td>
<td>{exec}</td>
</tr>
<tr>
<td>Bob</td>
<td>{read, write}</td>
<td>{read}</td>
<td>{}</td>
</tr>
<tr>
<td>Charlie</td>
<td>{read, write}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>
BLP Model – F

- F is the set security level assignments
- An element of F is a triple \((f_S, f_C, f_O)\)
- \(f_S\) - Maximal security level
- \(f_C\) - Current security level
- \(f_O\) - Classification

\[L_S = [Alice \leftrightarrow (Admin, \{W, H\}),
            Bob \leftrightarrow (Manager, \{H\}),
            Charlie \leftrightarrow (User, \{H\})]\]

\[L_C = [Alice \leftrightarrow (Manager, \{H\}),
            Bob \leftrightarrow (Manager, \{H\}),
            Charlie \leftrightarrow (User, \{H\})]\]

\[L_O = [FileA \leftrightarrow (User, \{H\}),
            FileB \leftrightarrow (Manager, \{H\}),
            FileC \leftrightarrow (Admin, \{H\})]\]
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.
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.
Questions
References

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


Recommended Reading

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