Access Control: DAC and MAC/LBAC

Prof. Ravi Sandhu
Executive Director and Endowed Chair

Lecture 2

ravi.utsa@gmail.com
www.profsandhu.com
Authentication, Authorization, Audit

AAA

- Authentication
  - Who are You?
- Authorization
  - What are You Allowed to Do?
- Audit
  - What Did You Do?

siloed → integrated
Access Control

Discretionary Access Control (DAC), 1970

Mandatory Access Control (MAC), 1970

Role Based Access Control (RBAC), 1995

Attribute Based Access Control (ABAC), ????
Access Matrix Model
Access Matrix Model

Objects (and Subjects)

F          G

U

r w own

r

V

r w own

rights
Basic Abstractions

- Subjects
- Objects
- Rights

The rights in a cell specify the access of the subject (row) to the object (column)
Users and Subjects

- A subject is a program (application) executing on behalf of a user.

- A user may at any time be idle, or have one or more subjects executing on its behalf.

- User-subject distinction is important if subject’s rights are different from a user’s rights:
  - Usually a subset
  - In many systems a subject has all the rights of a user.

- A human user may manifest as multiple users (accounts, principals) in the system.
Users and Subjects

J0E

J0E.T0P-SECRET
J0E.SECRET
J0E.CONFIDENTIAL
J0E.UNCLASSIFIED

USER
SUBJECTS
Users and Subjects

JANE

- JANE.CHAIRPERSON
- JANE.FACULTY
- JANE.EMPLOYEE
- JANE.SUPER-USER

USER

SUBJECTS
An object is anything on which a subject can perform operations (mediated by rights)

Usually objects are passive, for example:
- File
- Directory (or Folder)
- Memory segment
with CRUD operations (create, read, update, delete)

But, subjects can also be objects, with operations
- kill
- suspend
- resume
Access Matrix Model

Objects (and Subjects)

<table>
<thead>
<tr>
<th>I</th>
<th>U</th>
<th>F</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>r w own</td>
<td>parent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subjects

© Ravi Sandhu

World-Leading Research with Real-World Impact!
Implementation

- Access Control Lists
- Capabilities
- Relations
each column of the access matrix is stored with the object corresponding to that column
### Capabilities

<table>
<thead>
<tr>
<th>U</th>
<th>F/r, F/w, F/own, G/r</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>G/r, G/w, G/own</td>
</tr>
</tbody>
</table>

Each row of the access matrix is stored with the subject corresponding to that row.
# Relations

<table>
<thead>
<tr>
<th>Subject</th>
<th>Access</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>r</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>w</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>own</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>r</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>r</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>w</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>own</td>
<td>G</td>
</tr>
</tbody>
</table>

commonly used in relational database management systems
ACLs versus Capabilities

- **Authentication**
  - ACL's require authentication of subjects and ACL integrity
  - Capabilities require integrity and propagation control

- **Access review**
  - ACL's are superior on a per-object basis
  - Capabilities are superior on a per-subject basis

- **Revocation**
  - ACL's are superior on a per-object basis
  - Capabilities are superior on a per-subject basis

- **Least privilege**
  - Capabilities provide for finer grained least privilege control with respect to subjects, especially dynamic short-lived subjects created for specific tasks
ACLs versus Capabilities

- **Authentication**
  - ACL's require authentication of subjects and ACL integrity
  - Capabilities require integrity and propagation control

- **Access review**
  - ACL's are superior on a per-object basis
  - Capabilities are superior on a per-subject basis

- **Revocation**
  - ACL's are superior on a per-object basis
  - Capabilities are superior on a per-subject basis

- **Least privilege**
  - Capabilities provide for finer grained least privilege control with respect to subjects, especially dynamic short-lived subjects created for specific tasks

Most Operating Systems use ACLs often in abbreviated form: owner, group, world
Content-Dependent Controls

- content dependent controls
  - you can only see salaries less than 50K, or
  - you can only see salaries of employees who report to you

- beyond the scope of Operating Systems and are provided by Database Management Systems
context dependent controls

- cannot access classified information via remote login
- salary information can be updated only at year end
- company's earnings report is confidential until announced at the stockholders meeting

can be partially provided by the Operating System and partially by the Database Management System

more sophisticated context dependent controls such as based on past history of accesses definitely require Database support
Information from an object which can be read can be copied to any other object which can be written by a subject.

Suppose our users are trusted not to do this deliberately. It is still possible for Trojan Horses to copy information from one object to another.
Trojan Horse Vulnerability of DAC

User B cannot read file F
User B can read contents of file F copied to file G
Copy Difference for rw

- Read of a digital copy is as good as read of original
- Write to a digital copy is not so useful
DAC Subtleties

- Chains of grants and revokes
- Inheritance of permissions
- Negative rights
Denning’s Axioms for Information Flow
Denning’s Axioms

\[ < \text{SC}, \rightarrow, \oplus > \]

- **SC**: set of security classes
- \( \rightarrow \subseteq \text{SC} \times \text{SC} \): flow relation (i.e., can-flow)
- \( \oplus: \text{SC} \times \text{SC} \rightarrow \text{SC} \): class-combining operator
Denning’s Axioms

\[ \langle SC, \rightarrow, \oplus \rangle \]

1. SC is finite

2. \( \rightarrow \) is a partial order on SC
   (i.e., reflexive, transitive, anti-symmetric)

3. SC has a lower bound L such that L \( \rightarrow A \) for all \( A \in SC \)

4. \( \oplus \) is a least upper bound (lub) operator on SC

Justification for 1 and 2 is stronger than for 3 and 4.
In practice we may have a partially ordered set (poset).
Denning’s Axioms Imply

- SC is a universally bounded lattice
- There exists a Greatest Lower Bound (glb) operator $\otimes$ (also called meet)
- There exists a highest security class H
Lattice Structures

Top Secret

Secret

Confidential

Unclassified

can-flow

Hierarchical Classes

reflexive and transitive edges are implied but not shown
Lattice Structures

Compartments and Categories

{ARMY, CRYPTO}

{ARMY}

{}
Lattice Structures

Compartments and Categories

{ARMY, NUCLEAR, CRYPTO}

{ARMY, NUCLEAR} {ARMY, CRYPTO} {NUCLEAR, CRYPTO}

{ARMY} {NUCLEAR} {CRYPTO}

{}
Lattice Structures

Hierarchical Classes with Compartments

The product of 2 lattices is a lattice.
Lattice Structures

Hierarchical Classes with Compartments

The product of 2 lattices is a lattice
Smith’s Lattice

- With large lattices a vanishingly small fraction of the labels will actually be used
  - Smith's lattice: 4 hierarchical levels, 8 compartments
  - number of possible labels = $4 \times 2^8 = 1024$
    Only 21 labels are actually used (2%)
- Consider 16 hierarchical levels, 64 compartments which gives $10^{20}$ labels
Extending a POSET to a Lattice

such extension is always possible
BLP Model for Confidentiality
BLP Basic Assumptions

- \( \text{SUB} = \{S_1, S_2, \ldots, S_m\} \), a fixed set of subjects
- \( \text{OBJ} = \{O_1, O_2, \ldots, O_n\} \), a fixed set of objects
- \( \text{R} = \{r, w\} \), a fixed set of rights
- \( D \), an \( m \times n \) discretionary access matrix with \( D[i,j] \subseteq R \)
- \( M \), an \( m \times n \) current access matrix with \( M[i,j] \subseteq R \)
BLP Model (Liberal ★-Property)

- Lattice of confidentiality labels \( \Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_p\} \)
- Static assignment of confidentiality labels \( \lambda: \text{SUB} \cup \text{OBJ} \rightarrow \Lambda \)
- \( M, \) an \( m \times n \) current access matrix with
  - \( r \in M[i,j] \Rightarrow r \in D[i,j] \land \lambda(S_i) \geq \lambda(O_j) \) simple security
  - \( w \in M[i,j] \Rightarrow w \in D[i,j] \land \lambda(S_i) \leq \lambda(O_j) \) liberal ★-property
BLP Model (Strict ★-Property)

- Lattice of confidentiality labels $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_p\}$
- Static assignment of confidentiality labels $\lambda: \text{SUB} \cup \text{OBJ} \rightarrow \Lambda$
- $M$, an $m \times n$ current access matrix with
  - $r \in M[i,j] \Rightarrow r \in D[i,j] \land \lambda(Si) \geq \lambda(Oj)$  simple security
  - $w \in M[i,j] \Rightarrow w \in D[i,j] \land \lambda(Si) = \lambda(Oj)$  strict ★-property
BLP vis a vis Lattices

dominance ≥ can-flow

Top Secret
Secret
Confidential
Unclassified
it is risky to visualize lattices as total orders but it is ok sometimes
BLP vis a vis Lattices

\[ \text{dominance} \geq \text{can-flow} \]

H (High)

L (Low)

often 2 levels suffice to make the main point
★-Property

- Applies to subjects not to users
  - Users are trusted (must be trusted) not to disclose secret information outside of the computer system
  - A user can login (create a subject) with any label dominated by the user’s clearance
  - Subjects are not trusted because they may have Trojan Horses embedded in the code they execute

- ★-property prevents deliberate leakage and does not address
  - inference
  - covert channels

- Simple-security and ★-Property do not account for
  - encryption
Biba Model for Integrity
BLP Revisited

dominance \geq \text{can-flow}

HS (High Secrecy)

LS (Low Secrecy)
Biba Inverted Flow

- HI (High Integrity)
- LI (Low Integrity)

\[
\text{dominance} \geq \text{can-flow}
\]
Biba and BLP Aligned: BLP Style

- HS (High Secrecy)
- LI (Low Integrity)
- LS (Low Secrecy)
- HI (High Integrity)

One-directional flow is the key point.
No need for opposite directions for confidentiality and integrity.
One-directional flow is the key point
No need for opposite directions for confidentiality and integrity
BLP-Biba Unified Lattice: BLP Style

dominance can-flow

≥
BLP versus Biba

- BLP and Biba are fundamentally equivalent and interchangeable
- Lattice-based access control is a mechanism for enforcing one-way information flow, which can be applied to confidentiality or integrity goals
- We will use the BLP formulation:
  - high confidentiality, low integrity at the top
  - low confidentiality, high integrity at the bottom
The Chinese Wall Lattice for
Separation of Duty
Chinese Wall Policy

- A commercial security policy for separation of duty driven confidentiality
- Mixture of free choice (discretionary) and mandatory controls
- Requires some kind of dynamic labelling
A consultant can access information about at most one company in each conflict of interest class.
Chinese Wall Example

BANKS

A

B

OIL COMPANIES

X

Y
Chinese Wall Lattice

SYSHIGH

A, X  A, Y  B, X  B, Y

A, -  -, X  -, Y  B, -

SYSLOW
Conclusion
BLP enforces one-directional information flow in a lattice of security labels

BLP can enforce one-directional information flow policies for

- Confidentiality
- Integrity
- Separation of duty
- Combinations thereof
Covert Channels
Covert Channels

- A covert channel is a communication channel based on the use of system resources not normally intended for communication between subjects (processes)
Covert Channels

High User → High Trojan Horse Infected Subject ↓

COVERT CHANNEL

Low Trojan Horse Infected Subject ← Low User

Information is leaked unknown to the high user
Covert Channels

High User → High Trojan Horse Infected Subject

Information is leaked unknown to the high user

Low User ← Low Trojan Horse Infected Subject

★-property prevents overt leakage of information and does not address covert channels
Side Channels

User 1

User 1’s Subject

User 2

User 2’s Trojan Horse Infected Subject

Information is leaked unknown to the User 1
Covert Channels versus Side Channels

- Covert channels require a cooperating sender and receiver
- Side channels do not require a sender but nevertheless information is leaked to a receiver
Coping with Covert/Side Channels

- Identify the channel
  - close the channel or slow it down
  - detect attempts to use the channel
  - tolerate its existence
Storage Channels

- Also known as Resource Exhaustion Channels
- Given 5GB pool of dynamically allocated memory
  - HIGH PROCESS (sender)
    - bit = 1 ⇒ request 5GB of memory
    - bit = 0 ⇒ request 0GB of memory
  - LOW PROCESS (receiver)
    - request 5GB of memory
    - if allocated then bit = 0 otherwise bit = 1
Timing Channels

➢ Also known as Load Sensing Channels

➢ Given 5GB pool of dynamically allocated memory

❖ HIGH PROCESS (sender)
  
  bit = 1 ⇒ enter computation intensive loop
  
  bit = 0 ⇒ go to sleep

❖ LOW PROCESS (receiver)
  
  perform a task with known computational requirement
  
  if completed promptly then bit = 0 otherwise bit = 1