8.3 Mandatory Flow Control Models

Mingsen Xu
Advanced Operating System

2011-10-26
Outline

• Mandatory Flow Control Models
  – Information Flow Control
  – Lattice Model
  – Multilevel Security Model
    – Bell-Lapadula Model
    – Biba Model
• State-of-art Work and Literature
• Future Research
• Reference
Mandatory Flow Control Models

• Discretionary access control
  - Methods for controlling accesses to resources traditional operating systems

• Why need mandatory flow control?
  - With the advances in networks and distributed systems, it is necessary to broaden the scope to include the control of information flow between distributed nodes on a system wide basis rather than only individual basis like discretionary control. [1, Randy & Johnson, 1997]

• Example:
  - Parking decal for parking garage is a capability given on a discretionary basis for accessing university resource.
  - Driving on the way to university, we have to obey the mandatory traffic regulation, which is applied to all the traffic on the road.
Information Flow Control

• Concerned with how information is disseminated or propagated from one object to another

• I.F.C. categorizes all system entities (subjects and objects) to different security classes

• The security classes of all entities must be specified clearly and class of an entity never changes after it has been created

• All permissible information flow paths among them are regulated using unambiguous security rules
Lattice Model

• Best-known information flow model

• Formulate the requirements of secure information flow among classes

• Central component is a lattice whose mathematical meaning is a structure consisting of a finite partially ordered set, with a least upper bound and greatest lower bound

• A lattice is a Directed Acyclic Graph (DAG) with a single source and sink
Lattice Model - cont’d

- Lattice model can be formally stated: $FM = < S, O, SC, F, \oplus, \otimes, \rightarrow >$
  - $S$: set of subjects
  - $O$: set of objects
  - $SC$: finite set of security classes
  - $F$: mapping function from $S$ or $O$ to $SC$, object $O$ is bound to a class called security classification, subject $S$ is bound to a class called security clearance
  - $\oplus$: Least upper bound operator on $SC$
  - $\otimes$: Greatest lower bound operator on $SC$
  - $\rightarrow$: Flow relation on pairs of security classes

- $FM$ is considered as secure only if the execution of a sequence of operations cannot cause an information flow that violates the relation $\rightarrow$
Lattice Model - cont’d

• Example: Linear ordered lattice
  - \( SC = \{C_1, \ldots, C_n\}, C_i \rightarrow C_j \text{ iff } i \leq j \)
  - \( C_i \oplus C_j = C_{\text{max}(i,j)}; C_i \otimes C_j = C_{\text{min}(i,j)} \)

\[ C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow \ldots \rightarrow C_{n-1} \rightarrow C_n \]

• Information can only flow upward, and once it reaches to a class \( C_i \), it cannot flow down to any class below \( C_i \)
• Suitable for any system in which all classes need to be totally ordered
Multilevel Security

• Prevent users from obtaining access to information for which they lack authorization

• A special case of the lattice-based information flow model

• Two fully established multilevel security models
  - The Bell-Lapadula Model
  - Biba Model
The Bell-LaPadula Model

- This model mainly concerns confidentiality of the information security

- The simple security property:
  - $SC(S)$ dominates $SC(O)$: Reading information from an object $O$ by a subject $S$ ("no read up")

- The *-property:
  - $SC(O)$ dominates $SC(S)$: Writing information to an object $O$ by a subject $S$ ("no write down")

- A problem with this model is: it does not deal with the integrity of data
The Biba Model

- This model mainly concerns integrity of the information security

- The simple security property:
  - $SC(S)$ dominates $SC(O)$: Writing information to an object $O$ by a subject $S$ ("no write up")
  - The Biba model restricts a subject from writing to a more trusted object.
The Biba Model - Cont’d

- The *-property:
  - SC(O) dominates SC(S): Reading information from an object O by a subject S (“no read down”)
## Comparison

<table>
<thead>
<tr>
<th>Bell-LaPadula Model</th>
<th>Biba Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information confidentiality (prevent leakage)</td>
<td>Information integrity (prevent corruption)</td>
</tr>
<tr>
<td>Read down: $SC(S) \geq SC(O)$</td>
<td>Read up: $SC(S) \leq SC(O)$</td>
</tr>
<tr>
<td>Write up: $SC(S) \leq SC(O)$</td>
<td>Write down: $SC(S) \geq SC(O)$</td>
</tr>
</tbody>
</table>
State-of-art Work and Literature
I. Protecting Privacy using the Decentralized Label Model (DIFC)

- The goal of this work is to make information flow control more practical than was previously possible. (Previously mentioned models have high run-time overhead and unduly restrict what computations can be performed.)

- This paper describes a new decentralized label model for information flow control (DIFC) in systems with mutual distrustful authority.
Decentralized Information Flow Control (DIFC)

• 2. Making information flow explicit in HiStar. [OSDI, 2006]
  - Existing DIFC systems operate as programming language abstractions and integrated into communication primitives in new operating systems.
  - Their advantages are fine-grained control of information flow and high performance, but require a shift in how applications are developed.

• 3. Information Flow Control for Standard OS Abstractions. [SOSP 2007]
  - The paper proposes Flume, which instead provides process-level DIFC as a minimal extension to the communication primitives.
  - It demonstrates that the advantages of DIFC can be brought to bear on standard operating systems and applications.

- This paper extends DIFC to the network. It presents DStar, a protocol and framework that enforces the security requirements of mutually distrustful components.

- DStar leverages OS-level protection on individual machines to provide DIFC in a distributed system. At a high level, DStar controls which messages sent to a machine can affect which messages sent from the machine, therefore securing systems out of untrusted components.

- DStar does not require any fully-trusted processes or machines, and is constructed to avoid hidden channels.
4. Securing Distributed Systems with Information Flow Control

[NSDI 2008]

- DStar enforces DIFC with labels.
- Communication permissions are specified with labels. Each process has a label; whether and in which direction two processes may communicate is a function of their labels.
- This function is actually a partial order, denoted as $\subseteq$.
- Example: Message can flow from S to R, only if $L_S \subseteq L_R$.
  Bidirectional communication is permitted only if in addition, $L_R \subseteq L_S$. (only read down, only write up)
- Since a distributed system cannot simultaneously observe the labels of processes on different machines, DStar also labels messages. S sends a message M to R, S specifies a label $L_m$ for the message, in which DStar enforces that $L_S \subseteq L_m \subseteq L_R$. 

Wednesday, October 26, 2011
5. Airavat: Security and Privacy for MapReduce [NSDI 2011]

- **Overview**
  - Airavat is a system providing end-to-end confidentiality, integrity, and differential privacy for distributed computations on sensitive data.
  - Airavat is based on the popular MapReduce framework.
  - To prevent information leakage, Airavat runs on SELinux and adds SELinux-like mandatory access control to the MapReduce distributed file system.
Future Research

• Information flow control in Cloud computing
  - A fundamental problem is users outsource their sensitive data to clouds, which are beyond the same trusted domain as data owners.
  - A service provider can access multiple virtual machines in clouds, so that sensitive information may be leaked to unauthorized customers.
  - Some prototype enforces the information flow policies at Infrastructure-as-a-Service (IaaS), but not mature yet.

• A practical delegation mechanism also receives many research interests
  - A user may wish to delegate his cloud instance access privileges to others.
  - Delegation mechanism is another essential component for cloud computing.
References


References


Thank You.