Formal Security Models

Based on Slides prepared by A. Jones and Y. Lin.
Material based on C. Landwehr paper

Why Formal Models?

- Regulations are generally descriptive rather than prescriptive, so they don’t tell you how to implement
- Systems must be secure
  - security must be demonstrable → proofs
  - therefore, formal security models

Military Security

- Classification levels
  - unclassified
  - classified: confidential, secret, top secret
- Compartments
  - topic specific
- Clearance - ability to access a certain level/compartment of sensitive information
Formal Models - Basic Concepts

- Finite state machine model
  - this structure is the basis for all models in this paper
- Lattice model
- Access matrix model
- Security kernel (small enough for verification)
- Information-flow model

Lattice Model (for military application)

- Sensitivity levels a, b
- Compartments c, d
  - (a,c) >= (b,d)
- iff a >= b and c contains d
- Implies greatest lower bound -- (undass, no compartments
- least upper bound -- (top secret, all compartments)

Access Matrix Model

- Three principal components: object, subject, rules
- Access matrix (subject X object)
  - read, write, append, and execute
- Reference monitor - checks each access
- Two approaches
  - capability list (row-wise)
  - access control list (column-wise)
### Access Matrix

<table>
<thead>
<tr>
<th>Subjects</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O4</th>
<th>S1</th>
<th>S2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rx</td>
<td>kill</td>
<td>kill</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rwx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
<td>r</td>
<td>rx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Take-Grant Model

- Use graphs to model access control
- Access right: read, write, take, grant
- Each directed arc represents a capability
  - arc from one object to another
  - labeled with access right
- Compact representation of sparse access matrix

### Take-Grant Model (cont)

- Set of rules for rewriting graph
  - E.g. take rule: A has take right to B, then A can acquire all rights to any object that B has
  - Rules control deletion & creation of arcs, objects
Take Grant Model (cont)

- Question asked of model: given initial graph plus rules, can A ever get right R to object X?
- I.e. question of graph transformation
- Undecidable for general graphs
- But decidable for specific graphs & rules
- Defined predicates: "can know", "can tell", "can steal"
Bell & LaPadula Model

- Captures military classification
- Use finite state machine
- Formally define a state to be secure, then consider transitions (that maintain security)
- Uses subjects & objects of access matrix
- Adds military security
  - subject has clearance & current class.n level
  - each object has a classification

Bell & LaPadula Model (cont)

- Four modes of access
  - read-only, append, execute, and read-write
- Ownership -- owner can pass access modes to owned object to other subjects
- Core of operating system is a monitor (security kernel) that checks all accesses
- Minimum code; prove its properties
- In practice, it is difficult to isolate all security-relevant functions to a small kernel

Bell & LaPadula Model (cont'd)

- Properties for a state to be secure
  - simple security property (restricts “reading up”)
  - the star-property (prohibits “writing down”)
- Tranquility principle
  - no operation may change the classification of an active object
Bell and LaPadula Model (cont'd)

- Rules of transition: create object, change security level, rescind access, give access, etc
- Trusted subjects
  - not to compromise security even if some accesses violate the star-property
- “Flat” set of objects
  - atomic objects, each with a single classification
  - no hierarchy

Problems of B-L Model

- Static representation is restrictive
- Although hierarchies of objects are added in later version, no corresponding appropriate set of axioms
- No clear guidance to determine trusted processes
- In practice, declassification is a problem

Problems of B-L Model (cont'd)

- Allow information to be transmitted improperly through control variables (storage channels)
- Their final forms don't contain storage channels, but timing channels can exist
- Many operations that are in fact secure will be disallowed by the model
Information-Flow Model

- Focus on operations that transfer information between objects
- Five components
  - objects -- hold information
  - processes -- active agents
  - security classes -- disjoint classes of information
  - flow relation -- given 2 classes, determine if information is allowed to flow from one to other

Information-Flow Model (conf'd)

- Flow relation forms a lattice
- Information flow (x->y)
  - explicit -- opn.s causing flow are independent of value of x, e.g. assignment operation, x=y
  - implicit -- conditional assignment (if x then y=z)
- A program is secure if it does not specify any information flows that violate the given flow relation

Information-Flow Model (conf'd)

- Program is secure if it does not specify any information flows that violate the given flow relation
- Consider static binding vs dynamic binding
Programs as Channels for Information Transmission

- Each of the models views a program as a medium for information transmission
- Key question
  - what information is conveyed by the execution of a program?
  - what deductions about protected information are possible?

Programs as Channels for Information Transmission (cont'd)

- Filters (Jones and Lipton)
  - views policy as function that maps from input domain of program to some subset of that domain
  - protection mechanism as a filter that assures that policy is followed

Discussion and Conclusion

- Each model defines its own world and its own concept of security in that world
- Appropriateness of a particular model depends on the application for which it is to be used
Discussion & Conclusion (cont’d)

- Common problem: an operation is either secure or not
  - not helpful in making trade-offs between security and performance
  - not true in the physical world, e.g., “safes”
- Formal verification or security properties of systems is an active research topic
- Most assume a security kernel

Discussion & Conclusion (cont’d)

- Models can be divided into three groups
  - controlling direct access to objects
  - information flows among objects
  - an observer’s ability to make inference
- Formal models of computer security are needed in order to ask or answer whether a computer system is secure

Relevant Specification Languages

Based on materials from
- I. Cerveto, NRL
Languages to Specify What?

- Message flow
- Message constituents
- Operating environment
- Protocol goals

Desirable Properties

- Unambiguous
- Simple
- Flexible
  - Adapts to protocols
- Powerful
  - Applies to a wide class of protocols
- Insightful
  - Gives insight about protocols

Language Families

- "Usual notation"
  - (user interfaces)
  - Knowledge logic
    - BAN
    - Process theory
      - Spi-calculus
      - Strands
    - MSR
    - FDR, Casper
    - Petri nets
  - Inductive methods
- Temporal logic
- Automata
  - CAPSL
  - NRL Protocol Analyzer
  - Mur(
  - ...

Why so many?
  - Experience from nature fields
  - Unifying problem
  - Scientifically intriguing
  - Funding opportunities
  - Convergence of approaches