Advanced Systems Security: Principles

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University
Access Control – The Right Way

• We said that ordinary operating systems cannot control code controlled by an adversary

• Review formalisms developed for “protection”
  ‣ and show how they are extended to enforce “security”

• Key concepts
  ‣ Reference monitor
    • Enforce access control comprehensively
  ‣ Mandatory protection state
    • Without allowing adversary to modify access control policy
  ‣ Later: Security models
Protection System

• Manages the authorization policy for a system
  ‣ It describes what operations each subject (via their processes) can perform on each object

• Consists of
  ‣ **State**: Protection state
  ‣ **State Ops**: Protection state operations
The Access Matrix

- An access matrix is one way to represent policy.
  - Frequently used mechanism for describing policy
- Columns are objects, subjects are rows.
- To determine if \( S_i \) has right to access object \( O_j \), find the appropriate entry.
- Succinct descriptor for \( O \) (\( |S| \times |O| \) entries)
- Matrix for each right.

<table>
<thead>
<tr>
<th></th>
<th>( O_1 )</th>
<th>( O_2 )</th>
<th>( O_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>( S_3 )</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Access Matrix Protection System

• Protection State
  ‣ Current state of matrix

• Can modify the protection state
  ‣ Via protection state operations
  ‣ E.g., can create objects
  ‣ E.g., owner can add a subject, operation mapping for their objects

• Lampson’s “Protection” paper
  ‣ Can even delegate authority to perform protection state ops
Protection System Problems

• Protection system approach is inadequate
  ‣ Suppose a process runs bad code

• Processes can change their own permissions
  ‣ Processes are untrusted, but can modify policy

• Processes, files, etc. are created and modified
  ‣ Cannot predict in advance (safety problem)

• What do we need to achieve necessary controls?
Claim: *If we can define and enforce a security policy that ensures security goals, then we can prevent such attacks*

- How do we know the policy expresses effective goals?
  - Will look into this in depth later

- How do we know the enforcement mechanism will enforce policy as expected?
  - Look into this today
Mandatory Protection System

• Is a protection system that can be modified only by trusted administration that consists of
  ‣ A mandatory protection state where the protection state is defined in terms of an immutable set of labels and the operations that subject labels can perform on object labels
  ‣ A labeling state that assigns system subjects and objects to those labels in the mandatory protection state
  ‣ A transition state that determines the legal ways that subjects and objects may be relabeled

• MPS is immutable to user-space process
## Mandatory Protection System

### Labeling State
- Process: newproc

### File: newfile
- Labeling State

### File: acct
- Transition State

### Protection State

<table>
<thead>
<tr>
<th></th>
<th>secret</th>
<th>unclassified</th>
<th>trusted</th>
<th>untrusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>secret</td>
<td>read write</td>
<td>read</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>unclassified</td>
<td>read write</td>
<td>read</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>trusted</td>
<td>write</td>
<td>read write</td>
<td>read write</td>
<td>write</td>
</tr>
<tr>
<td>untrusted</td>
<td>read write</td>
<td>read write</td>
<td>read write</td>
<td>read write</td>
</tr>
</tbody>
</table>

### Process: other
- Transition State
Mandatory Protection State

• Immutable table of
  ‣ Subject labels
  ‣ Object labels
  ‣ Operations authorized for former upon latter

• How can you use an MPS to control use of bad code?
  ‣ E.g., Prevent modification of kernel memory?
Mandatory Protection State

- Immutable table of
  - Subject labels
  - Object labels
  - Operations authorized for former upon latter

- How can you use an MPS to control use of bad code?
  - E.g., Prevent modification of kernel memory?
  - (1) if a process reads adversary-accessible object label, remove permission to modify kernel memory
  - (2) if a process reads adversary-accessible object label, remove permission to write to any process with access to kernel memory (transitively)
Labeling State

• Immutable rules mapping
  ‣ Subjects to labels (in rows)
  ‣ Objects to labels (in columns)

• How can you use labeling state to control bad code?
  ‣ E.g., Prevent modification of kernel memory?
Labeling State

- Immutable rules mapping
  - Subjects to labels (in rows)
  - Objects to labels (in columns)

- How can you use labeling state to control bad code?
  - E.g., Prevent modification of kernel memory?
  - Assign all processes that may run bad code
  - With a label that has restricted permissions
  - What about objects created by these processes?
Transition State

• Immutable rules mapping
  ‣ Subject labels to conditions that change their subject labels
  ‣ Object labels to conditions that change their object labels

• How can you use labeling state to control bad code?
  ‣ E.g., Prevent modification of kernel memory?
Transition State

• Immutable rules mapping
  ‣ Subject labels to conditions that change their subject labels
  ‣ Object labels to conditions that change their object labels

• How can you use labeling state to control bad code?
  ‣ E.g., Prevent modification of kernel memory?
  ‣ Prevent bad code from launching a process of a label that can modify kernel memory
  ‣ How do we launch processes with more permissions now?
Managing MPS

• Challenge
  ‣ Determining how to set and manage an MPS in a complex system involving several parties

• Parties
  ‣ What does programmer know about deploying their program securely?
  ‣ What does an OS distributor know about running a program in the context of their system?
  ‣ What does an administrator know about programs and OS?
  ‣ Users?
Reference Monitor

- **Purpose:** Ensure enforcement of security goals
  - Define goals in the mandatory protection system
  - Reference monitor ensures enforcement

- *Every component that you depend upon to enforce your security goals must be a reference monitor*
Reference Monitor

- Components
  - Reference monitor interface (e.g., LSM)
  - Reference validation mechanism (e.g., SELinux)
  - Policy store (e.g., policy binary)
Reference Monitor Guarantees

- **Complete Mediation**
  - The reference validation mechanism must always be invoked

- **Tamperproof**
  - The reference validation mechanism must be tamperproof

- **Verifiable**
  - The reference validation mechanism must be subject to analysis and tests, the completeness of which must be assured
Complete Mediation

- Every security-sensitive operation must be mediated
  - What’s a “security-sensitive operation”?
  - E.g., operation that may not be authorized for every subject

- How do we validate complete mediation?
  - Every security-sensitive operation must be identified
  - E.g., ensure every execution of that operation is checked

- **Mediation**: Does interface mediate?
- **Mediation**: On all resources?
- **Mediation**: Verifably?
**Tamperproof**

- Prevent modification by untrusted entities
  - Interface, mechanism, policy of reference monitor
  - Code and policy that can affect reference monitor mods
- How to detect tamperproofing?
  - Transitive closure of operations
  - Challenge: Often some untrusted operations are present
- **Tamperproof**: Is reference monitor protected?
- **Tamperproof**: Is system TCB protected?
Verification

• Determine correctness of code and policy
  ‣ What defines correct code?
  ‣ What defines a correct policy?

• Test and analyze reference validation mechanism
  ‣ Does code/policy do its job correctly?
  ‣ For all executions

• **Verifiable**: Is TCB code base correct?

• **Verifiable**: Does the MPS enforce the system’s security goals?
Evaluation

- **Mediation**: Does interface mediate?
- **Mediation**: On all resources?
- **Mediation**: Verifiably?
- **Tamperproof**: Is reference monitor protected?
- **Tamperproof**: Is system TCB protected?
- **Verifiable**: Is TCB code base correct?
- **Verifiable**: Does the MPS enforce the system’s security goals?
Take Away

- Mandatory Protection System
  - Means to define security goals that applications cannot impact

- Reference Monitor Concept
  - Requirements for a reference validation mechanism that can correctly enforce an MPS
  - NOTE: This will be a major focus of this course

- Until we come up with coherent approach to validating MPS meets security goals and validating reference monitor guarantees, we will continue to have insecure systems
  - That is the challenge of systems security research