Lecture 3
September 26, 2021
Types of Security Policies

• Military (governmental) security policy
  • Policy primarily protecting confidentiality

• Commercial security policy
  • Policy primarily protecting integrity

• Confidentiality policy
  • Policy protecting only confidentiality

• Integrity policy
  • Policy protecting only integrity
Access Control Matrix

• Access Control Matrix Model
• Protection State Transitions
  • Commands
  • Conditional Commands
• Special Rights
• Principle of Attenuation of Privilege
Description

- Subjects $S = \{ s_1, \ldots, s_n \}$
- Objects $O = \{ o_1, \ldots, o_m \}$
- Rights $R = \{ r_1, \ldots, r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_{x}, \ldots, r_{y} \}$ means subject $s_i$ has rights $r_x, \ldots, r_y$ over object $o_j$
Example 1

- Processes $p, q$
- Files $f, g$
- Rights $r, w, x, a, o$

<table>
<thead>
<tr>
<th></th>
<th>$f$</th>
<th>$g$</th>
<th>$p$</th>
<th>$q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>rwo</td>
<td>$r$</td>
<td>rwxo</td>
<td>$w$</td>
</tr>
<tr>
<td>$q$</td>
<td>$a$</td>
<td>ro</td>
<td>$r$</td>
<td>rwxo</td>
</tr>
</tbody>
</table>
Example 2

- Host names *telegraph*, *nob*, *toadflax*
- Rights *own*, *ftp*, *nfs*, *mail*

<table>
<thead>
<tr>
<th></th>
<th>telegraph</th>
<th>nob</th>
<th>toadflax</th>
</tr>
</thead>
<tbody>
<tr>
<td>telegraph</td>
<td>own</td>
<td>ftp</td>
<td>ftp</td>
</tr>
<tr>
<td>nob</td>
<td>ftp, mail, nfs, own</td>
<td>ftp, nfs, mail</td>
<td></td>
</tr>
<tr>
<td>toadflax</td>
<td>ftp, mail</td>
<td>ftp, mail, nfs, own</td>
<td></td>
</tr>
</tbody>
</table>
Example 3

- Procedures *inc_ctr, dec_ctr, manage*
- Variable *counter*
- Rights +, −, *call*

<table>
<thead>
<tr>
<th></th>
<th>counter</th>
<th>inc ctr</th>
<th>dec ctr</th>
<th>manage</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>inc_ctr</em></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>dec_ctr</em></td>
<td>−</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>manager</em></td>
<td></td>
<td><em>call</em></td>
<td><em>call</em></td>
<td><em>call</em></td>
</tr>
</tbody>
</table>
State Transitions

• Change the protection state of system

• $\mid$ represents transition
  • $X_i \mid_{\tau} X_{i+1}$: command $\tau$ moves system from state $X_i$ to $X_{i+1}$
  • $X_i \mid^{*} Y$: a sequence of commands moves system from state $X_i$ to $Y$

• Commands often called *transformation procedures*
Primitive Operations

- **create subject** $s$; **create object** $o$
  - Creates new row, column in ACM; creates new column in ACM

- **destroy subject** $s$; **destroy object** $o$
  - Deletes row, column from ACM; deletes column from ACM

- **enter** $r$ **into** $A[s, o]$
  - Adds $r$ rights for subject $s$ over object $o$

- **delete** $r$ **from** $A[s, o]$
  - Removes $r$ rights from subject $s$ over object $o$
Creating File

• Process $p$ creates file $f$ with $r$ and $w$ permission

  command create\cdot file(p, f)
  
  create object $f$;
  enter own into $A[p, f]$;
  enter $r$ into $A[p, f]$;
  enter $w$ into $A[p, f]$;
  end
Mono-Operational Commands

• Make process $p$ the owner of file $g$

```plaintext
command make\cdot owner(p, g)
    enter own into A[p, g];
end
```

• Mono-operational command
  • Single primitive operation in this command
Conditional Commands

• Let $p$ give $q$ read rights over $f$, if $p$ owns $f$

```
command grant•read•file•1(p, f, q)
  if own in A[p, f]
  then
    enter $r$ into A[q, f];
end
```

• Mono-conditional command
  • Single condition in this command
Multiple Conditions

- Let $p$ give $q$ read and write rights over $f$, if $p$ owns $f$ and $p$ has $c$ rights over $q$

  
  ```plaintext
  command grant•read•file•2(p, f, q)
  if own in A[p, f] and c in A[p, q]
  then
    enter $r$ into A[q, f];
    enter $w$ into A[q, f];
  end
  ```

  

Copy Flag and Right

• Allows possessor to give rights to another
• Often attached to a right (called a flag), so only applies to that right
  • $r$ is read right that cannot be copied
  • $rc$ is read right that can be copied
• Is copy flag copied when giving $r$ rights?
  • Depends on model, instantiation of model
Own Right

- Usually allows possessor to change entries in ACM column
  - So owner of object can add, delete rights for others
  - May depend on what system allows
    - Can’t give rights to specific (set of) users
    - Can’t pass copy flag to specific (set of) users
Attenuation of Privilege

• Principle says you can’t increase your rights, or give rights you do not possess
  • Restricts addition of rights within a system
  • Usually *ignored* for owner
    • Why? Owner gives herself rights, gives them to others, deletes her rights.
What Is “Secure”?

• Adding a generic right $r$ where there was not one is "leaking"
  • In what follows, a right leaks if it was not present *initially*
  • Alternately: not present *in the previous state* (not discussed here)

• If a system $S$, beginning in initial state $s_0$, cannot leak right $r$, it is *safe with respect to the right $r*"
  • Otherwise it is called *unsafe with respect to the right $r*"
Safety Question

• Does there exist an algorithm for determining whether a protection system $S$ with initial state $s_0$ is safe with respect to a generic right $r$?

• Here, “safe” = “secure” for an abstract model
Mono-Operational Commands

• Answer: yes

• Sketch of proof:
  Consider minimal sequence of commands $c_1, ..., c_k$ to leak the right.
  • Can omit delete, destroy
  • Can merge all creates into one

Worst case: insert every right into every entry; with $s$ subjects and $o$ objects initially, and $n$ rights, upper bound is $k \leq n(s+1)(o+1)$
General Case

• Answer: \textit{no}

• Sketch of proof:
  
  Reduce halting problem to safety problem
  
  • Map head motion of Turing machine into entering, deleting rights in the access control matrix
  
  • Turing machine symbols mapped into rights
  
  • Head position, end of tape indicated by special rights
  
  • Head motion represented by commands; two sets for R motion
    
    • One for mid-tape, one for end of tape
  
  • So protection system simulates a Turing machine \textit{exactly}
  
  • TM halts when it enters state $q_f$; this means right has leaked
Confidentiality Models

• Overview
  • What is a confidentiality model
• Bell-LaPadula Model
  • General idea
  • Informal description of rules
• Tranquility
• Declassification
Confidentiality Policy

• Goal: prevent the unauthorized disclosure of information
  • Deals with information flow
  • Integrity incidental

• Multi-level security models are best-known examples
  • Bell-LaPadula Model basis for many, or most, of these
Bell-LaPadula Model, Step 1

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest

- Levels consist are called *security clearance* \( L(s) \) for subjects and *security classification* \( L(o) \) for objects
Example

<table>
<thead>
<tr>
<th>security level</th>
<th>subject</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Secret</td>
<td>Tamara</td>
<td>Personnel Files</td>
</tr>
<tr>
<td>Secret</td>
<td>Samuel</td>
<td>E-Mail Files</td>
</tr>
<tr>
<td>Confidential</td>
<td>Claire</td>
<td>Activity Logs</td>
</tr>
<tr>
<td>Unclassified</td>
<td>Ulaley</td>
<td>Telephone Lists</td>
</tr>
</tbody>
</table>

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaley can only read Telephone Lists
Reading Information

• Information flows up, not down
  • “Reads up” disallowed, “reads down” allowed

• Simple Security Condition (Step 1)
  • Subject s can read object o iff, \( L(o) \leq L(s) \) and s has permission to read o
    • Note: combines mandatory control (relationship of security levels) and discretionary
      control (the required permission)
  • Sometimes called “no reads up” rule
Writing Information

• Information flows up, not down
  • “ Writes up” allowed, “ writes down” disallowed

• *-Property (Step 1)
  • Subject $s$ can write object $o$ iff $L(s) \leq L(o)$ and $s$ has permission to write $o$
    • Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  • Sometimes called “no writes down” rule
Basic Security Theorem, Step 1

• If a system is initially in a secure state, and every transition of the system satisfies the simple security condition, step 1, and the *-property, step 1, then every state of the system is secure
  • Proof: induct on the number of transitions