ECS 235B, Lecture 1

January 7, 2019
Access Control Matrix
Overview

• Access Control Matrix Model
  • Boolean Expression Evaluation
  • History

• 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><code>inc_ctr</code></th>
<th><code>dec_ctr</code></th>
<th><code>manage</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>inc_ctr</code></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>dec_ctr</code></td>
<td>–</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>manager</code></td>
<td></td>
<td><code>call</code></td>
<td><code>call</code></td>
<td><code>call</code></td>
</tr>
</tbody>
</table>
Boolean Expression Evaluation

• ACM controls access to database fields
  • Subjects have attributes
  • Verbs define type of access
  • Rules associated with objects, verb pair

• Subject attempts to access object
  • Rule for object, verb evaluated, grants or denies access
Example

• Subject annie
  • Attributes role (artist), group (creative)

• Verb paint
  • Default 0 (deny unless explicitly granted)

• Object picture
  • Rule:
    paint: ‘artist’ in subject.role and
    ‘creative’ in subject.groups and
    time.hour ≥ 0 and time.hour ≤ 4
ACM at 3AM and 10AM

At 3AM, time condition met
ACM is:

At 10AM, time condition not met
ACM is:
History

• Problem: what a process has accessed may affect what it can access now

• Example: procedure in a web applet can access other procedures depending on what procedures it has already accessed
  • S set of *static rights* associated with procedure
  • C set of current rights associated with each executing process
  • When process calls procedure, rights are $S \cap C$
Example Program

// This routine has no filesystem access rights
// beyond those in a limited, temporary area
procedure helper_proc()
  return sys_kernel_file

// But this has the right to delete files
program main()
  sys_load_file(helper_proc)
  tmp_file = helper_proc()
  sys_delete_file(tmp_file)

• sys_kernel_file contains system kernel

• tmp_file is in limited area that helper_proc() can access
Before \textit{helper\_proc} Called

- Static rights of program
  
  \begin{tabular}{c|c|c}
    & \textit{sys\_kernel\_file} & \textit{tmp\_file} \\
  \hline
  main & delete & delete \\
  helper\_proc & & delete \\
  
  \end{tabular}

- When program starts, current rights:

  \begin{tabular}{c|c|c}
    & \textit{sys\_kernel\_file} & \textit{tmp\_file} \\
  \hline
  main & delete & delete \\
  helper\_proc & & delete \\
  process & delete & delete \\
  
  \end{tabular}
After *helper_proc* Called

- Process rights are intersection of static, previous “current” rights:

<table>
<thead>
<tr>
<th></th>
<th><em>sys_kernel_file</em></th>
<th><em>tmp_file</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>main</em></td>
<td>delete</td>
<td>delete</td>
</tr>
<tr>
<td><em>helper_proc</em></td>
<td>delete</td>
<td>delete</td>
</tr>
<tr>
<td><em>process</em></td>
<td></td>
<td>delete</td>
</tr>
</tbody>
</table>
State Transitions

• Change the protection state of system
• |– represents transition
  • $X_i |\rightarrow_{\tau} X_{i+1}$: command $\tau$ moves system from state $X_i$ to $X_{i+1}$
  • $X_i |\rightarrow^{*} Y$: 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 $o$
Create Subject

• Precondition: \( s \notin S \)
• Primitive command: create subject \( s \)
• Postconditions:
  • \( S' = S \cup \{ s \}, \ O' = O \cup \{ s \} \)
  • (\( \forall y \in O' \) \( [A'[s, y] = \emptyset] \), (\( \forall x \in S' \) \( [A'[x, s] = \emptyset] \))
  • (\( \forall x \in S \) (\( \forall y \in O \) \( [A'[x, y] = A[x, y]] \))
Create Object

• Precondition: $o \notin O$
• Primitive command: \texttt{create object} $o$
• Postconditions:
  • $S' = S$, $O' = O \cup \{ o \}$
  • $(\forall x \in S') [A'(x, o) = \emptyset]$
  • $(\forall x \in S)(\forall y \in O) [A'[x, y] = A[x, y]]$
Add Right

• Precondition: \( s \in S, o \in O \)
• Primitive command: \textbf{enter} \( r \) \textbf{into} \( A[s, o] \)
• Postconditions:
  • \( S' = S, O' = O \)
  • \( A'[s, o] = A[s, o] \cup \{ r \} \)
  • \((\forall x \in S')(\forall y \in O' - \{ o \}) [A'[x, y] = A[x, y]]\)
  • \((\forall x \in S' - \{ s \})(\forall y \in O') [A'[x, y] = A[x, y]]\)
Delete Right

• Precondition: $s \in S, o \in O$

• Primitive command: delete $r$ from $A[s, o]$

• Postconditions:
  • $S' = S, O' = O$
  • $A'[s, o] = A[s, o] - \{ r \}$
  • $(\forall x \in S')(\forall y \in O' - \{ o \}) [A'[x, y] = A[x, y]]$
  • $(\forall x \in S' - \{ s \})(\forall y \in O') [A'[x, y] = A[x, y]]$
Destroy Subject

• Precondition: $s \in S$

• Primitive command: \texttt{destroy subject} $s$

• Postconditions:
  • $S' = S - \{s\}$, $O' = O - \{s\}$
  • $(\forall y \in O') [A'[s, y] = \emptyset]$, $(\forall x \in S') [A'[x, s] = \emptyset]$
  • $(\forall x \in S')(\forall y \in O') [A'[x, y] = A[x, y]]$
Destroy Object

• Precondition: \( o \in O \)
• Primitive command: \textbf{destroy object} \( o \)
• Postconditions:
  • \( S' = S, \ O' = O - \{ \ o \} \)
  • \((\forall x \in S') \ [A'[x, \ o] = \emptyset]\)
  • \((\forall x \in S')(\forall y \in O') \ [A'[x, y] = A[x, y]]\)
Creating File

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

```
command create·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 $g$

\[
\text{command } \textit{make}\cdot\textit{owner}(p, g) \\
\text{enter } \textit{own} \text{ into } A[p, g]; \\
\text{end}
\]

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

• Let \( p \) give \( q \) \( r \) rights over \( f \), if \( p \) owns \( f \)

\[
\text{command } \text{grant}\cdot \text{read}\cdot \text{file}\cdot 1(p, f, q) \\
\quad \text{if own in } A[p, f] \\
\quad \text{then} \\
\quad \quad \text{enter } r \text{ into } A[q, f]; \\
\text{end}
\]

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

• Let \( p \) give \( q \) \( r \) and \( w \) rights over \( f \), if \( p \) owns \( f \) and \( p \) has \( c \) rights over \( q \)

\[
\text{command} \quad \text{grant} \cdot \text{read} \cdot \text{file} \cdot 2(p, f, q) \\
\quad \text{if own in } A[p, f] \text{ and } c \text{ in } A[p, q] \\
\quad \text{then} \\
\quad \quad \text{enter } r \text{ into } A[q, f]; \\
\quad \quad \text{enter } w \text{ into } A[q, f]; \\
\text{end}
\]
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.
Key Points

• Access control matrix simplest abstraction mechanism for representing protection state

• Transitions alter protection state

• 6 primitive operations alter matrix
  • Transitions can be expressed as commands composed of these operations and, possibly, conditions