Access Control Matrix

ECS 153 Spring Quarter 2021

Module 7
• 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>ftp, nfs, mail</td>
</tr>
<tr>
<td>toadflax</td>
<td>ftp, mail</td>
<td>ftp, mail, nfs, own</td>
<td>ftp, mail, nfs, own</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>
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:

... picture ...
annie ...
paint ...

At 10AM, time condition not met
ACM is:

... picture ...
annie ...

State Transitions

• Change the protection state of system
• \( |\rightarrow \) 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: \( s \not\in S \)
- Primitive command: \texttt{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: 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: **enter** $r$ 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: 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: `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\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 $g$

  \[
  \text{command} \quad \text{make\textbullet owner} (p, g) \\
  \hspace{1cm} \text{enter own into } A[p, g]; \\
  \hspace{1cm} \text{end}
  \]

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

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

  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 $q$ $r$ and $w$ rights over $f$, if $p$ owns $f$ and $p$ has $c$ rights over $q$
  
  ```
  command grant\cdot read\cdot file\cdot 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 and Basic Results

• Is there 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 systems: yes, there is such an algorithm

• General systems: no, there is no such algorithm
  • Proof: reduce the halting problem to the safety question
  • Proved by Harrison, Ruzzo, and Ullman; often called the HRU result
  • Says nothing about particular classes of systems; this is a generic result
Mono-Operational Commands

• Answer: yes

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

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