ECS 235B Module 4
Access Control Matrix
- Subjects $S = \{ s_1, \ldots, s_n \}$
- Objects $O = \{ o_1, \ldots, o_m \}$
- Rights $R = \{ r_1, \ldots, r_k \}$
- Entries $A[s_i, o_j] \subset R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ 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>
UNIX/Linux Access Controls

• Files: A is "bishop/a.out (0755), B is /etc/passwd (0644), H is /home/bishop (0711), S is /bin/su (4711)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>S</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>bishop</td>
<td>rwxo</td>
<td>r</td>
<td>x</td>
<td>rwxo</td>
</tr>
<tr>
<td>zheng</td>
<td>rx</td>
<td>r</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>root</td>
<td>rwx</td>
<td>rwo</td>
<td>rwxo</td>
<td>rwx</td>
</tr>
</tbody>
</table>

January 6, 2021; Module 4

ECS 235B, Foundations of Computer and Information Security
UNIX/Linux Access Controls

- Access control matrices are dynamic:
- After bishop executes `chmod 700 /home/bishop`:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>S</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>bishop</td>
<td>rwxo</td>
<td>r</td>
<td>x</td>
<td>rwxo</td>
</tr>
<tr>
<td>muwei</td>
<td></td>
<td>r x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>root</td>
<td>rwx</td>
<td>rwo</td>
<td>rwxo</td>
<td>rwx</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:

\[
\begin{array}{cccc}
\ldots & \text{picture} & \ldots \\
\text{annie} & \text{paint} & \text{annie} \\
\ldots & \ldots & \ldots \\
\end{array}
\]

At 10AM, time condition not met
ACM is:

\[
\begin{array}{cccc}
\ldots & \text{picture} & \ldots \\
\text{annie} & \text{annie} & \text{annie} \\
\ldots & \ldots & \ldots \\
\end{array}
\]
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 *helper_proc* Called

- Static rights of program

<table>
<thead>
<tr>
<th></th>
<th><code>sys_kernel_file</code></th>
<th><code>tmp_file</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main</code></td>
<td>delete</td>
<td>delete</td>
</tr>
<tr>
<td><code>helper_proc</code></td>
<td></td>
<td>delete</td>
</tr>
</tbody>
</table>

- When program starts, current rights:

<table>
<thead>
<tr>
<th></th>
<th><code>sys_kernel_file</code></th>
<th><code>tmp_file</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main</code></td>
<td>delete</td>
<td>delete</td>
</tr>
<tr>
<td><code>helper_proc</code></td>
<td></td>
<td>delete</td>
</tr>
<tr>
<td><code>process</code></td>
<td>delete</td>
<td>delete</td>
</tr>
</tbody>
</table>
After `helper_proc` Called

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

<table>
<thead>
<tr>
<th></th>
<th><code>sys_kernel_file</code></th>
<th><code>tmp_file</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main</code></td>
<td>delete</td>
<td>delete</td>
</tr>
<tr>
<td><code>helper_proc</code></td>
<td></td>
<td>delete</td>
</tr>
<tr>
<td><code>process</code></td>
<td></td>
<td>delete</td>
</tr>
</tbody>
</table>
Quiz

In an access control matrix, do the rights “r”, “w”, and “x” represent “read”, “write”, and “execute” permissions, respectively?

• Yes, because the permission symbols are tied to those permissions (“r” for “read”, “w” for “write”, “x” for “execute”).

• Possibly; the meanings of the permission symbols depends upon the instantiation.

• No, because the meanings of the permission symbols changes as the matrix evolves, so you cannot say what the symbols mean; you can only manipulate them based on the given commands.