CS 4410
Operating Systems

Security (2)

Summer 2016
Cornell University
Today

• Access control
• DAC
• MAC
Access control

• Confidentiality and integrity are often enforced using access control.
  – Predefined operations are the sole means by which principals access information.
  – A reference monitor is consulted whenever one of these predefined operations is invoked.
  – The operation is allowed only if the invoker holds the required privileges.
Discretionary Access Control (DAC)

• In a DAC policy, the owner of an object controls the assignment of privileges for this objects to principals.
• DAC policies are what commercial operating systems typically enforce.
• The assignment of privileges by a DAC policy can be depicted using a table Auth that has a row for each principal and a column for each object.
• Any DAC policy can be circumvented if principals are permitted to make arbitrary changes to Auth.
  – Yet as execution of a system proceeds, changes to Auth will inevitably be needed.
Protection Domains

• Having users as the set of principals is too coarse-grained.
• Principle of Least Privilege: the set of operations a principal should be authorized to execute depends on the task to be performed.
• Use *protection domains* as the set of principals, instead.
• Each protection domain is associated with a different set of privileges.
## Protection Domains

<table>
<thead>
<tr>
<th>Domains</th>
<th>notes.txt</th>
<th>beach.img</th>
<th>sort.py</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann@edit</td>
<td>r, w</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>Ann@view</td>
<td></td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>Beth@edit</td>
<td></td>
<td></td>
<td>r, w</td>
</tr>
<tr>
<td>Beth@view</td>
<td></td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>George@edit</td>
<td></td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>George@view</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Protection Domains

• Allow transitions from one protection domain to another as execution of a thread proceeds.

• Different sets of privileges can now be associated with a thread as it progresses from one task to the next.

• In an operating system, system calls may cause protection-domain transitions.
  – Example: change from user mode to kernel mode.
Implementing DAC

- Auth is sparse. So, implementing Auth as an array is not efficient.
- Need data structures that store only the non-empty cells of Auth.
- Two approaches:
  - An access control list encodes the non-empty cells associated with a column (object).
  - A list of capabilities encode the non-empty cells associated with a row (principal).
- Access control lists and capabilities can, in theory, express the same policies.
- In practice, they differ in the cost of performing revocation and review.
Access Control Lists

• The access control list for an object O is a list
  \(<P1; \text{Privileges}1> <P2; \text{Privileges}2> ... <PN; \text{Privileges}N>\)
• Operating system abstractions (e.g., files, sockets, locks) can be protected with access control lists.
• System calls are then the only way to access an operating system abstraction.
  – A reference monitor is embedded in the operating system routine that handles a system call.
• Large operating system abstractions (e.g., files) can store their own access control lists.
• For small operating system abstractions (e.g., locks or ports), the operating system's memory can be used to store the access control lists.
Capabilities

• A capability is a pair <O; Privileges>.
• Any principal that holds this capability is granted Privileges for operations on O.
• Assumption: Capabilities cannot be counterfeited or corrupted.
• An authorized principal P can:
  – create a new object and receive a capability for that object,
  – transfer to other principals one or more capabilities P holds, and
  – revoke capabilities that derive from capabilities P holds.
DAC in Unix: Accessing a file

• Authorization to access a file is partitioned into
  – a potentially expensive check, which is done infrequently,
  – and cheaper checks, which are performed for each file access.
• The expensive check is moved into an additional system call.
  – This open system call for a file must be executed prior to attempting read or
    write system calls on that file.
  – The access control list of the file specifies if the open system call is successful.
• The constraint that open be executed first is enforced because read and
  write require a file handle argument.
• A file handle can be considered as a capability.
• Subsequent read and write systems call use this file handle to access the
  file.
• The hybrid of access control lists and capability-like authorization is not a
  panacea.
• Its latency for revocations can be unbounded, because the access control
  list is not rechecked each time read and write execute.
Mandatory Access Control (MAC)

• With DAC, the owner specifies allowed operation on the object.
• The goals of an institution, however, might not align with those of any individual.
• So rules set by the institution are the more natural basis for authorization.
• MAC: the institution specifies rules for authorization.
Mandatory Access Control (MAC)

• A *classification* $L(D)$ is assigned to each document $D$.

• A *clearance* $L(U)$ is assigned to each person $U$.

• $L$ maps to a set of *labels*.
  – Example: Top Secret (TS), Secret (S), Confidential (C), Unclassified (U).
  – The institution decides $L(D)$ and $L(U)$.

• **Confidentiality Policy.** A person $U$ is permitted to see a document $D$ only if $L(D) \leq L(U)$ holds,
  – where: $U \leq C \leq S \leq TS$. 

MAC: Confidentiality

- **Program Invocation.** \( L(Pgm) \leq L(U) \) must hold for a program Pgm executing on behalf of a user U.

- **Read Restriction.** \( L(F) \leq L(Pgm) \) must hold for program Pgm to read a file F.

- **Write Restriction.** \( L(Pgm) \leq L(F) \) must to hold for a program Pgm to write into a file F.
Today

• Access control
• DAC
• MAC
Coming up…

• Next lecture: Review
• Student evaluation