Lecture 16: Capabilities
Where we were…

- **Authentication**: mechanisms that bind principals to actions
- **Authorization**: mechanisms that govern whether actions are permitted
  - Discretionary Access Control
  - Mandatory Access Control
Access Control Policy

- An **access control policy** specifies which of the **operations** associated with any given **object** each **principal** is authorized to perform.
- Expressed as a relation $\text{Auth}$:

<table>
<thead>
<tr>
<th>Auth</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dac.tex</td>
</tr>
<tr>
<td>ebirrell</td>
<td>r,w</td>
</tr>
<tr>
<td>clarkson</td>
<td>r</td>
</tr>
<tr>
<td>student</td>
<td>r</td>
</tr>
</tbody>
</table>

Access Control Lists

Capability lists
Capability Lists

- The capability list for a principal $P$ is a list
  $\langle O_1, Privs_1 \rangle, \langle O_2, Privs_2 \rangle, \ldots, \langle O_n, Privs_n \rangle$
  
  - e.g., $\langle$ dac.tex, $\{r,w\}\rangle$ $\langle$ dac.pptx, $\{r,w\}\rangle$

- **Capabilities** carry privileges.
  1) **Authorization**: Performing operation $op$ on object $O_i$ requires a principal $P$ to hold a capability $C_i = \langle O_i, Privs_i \rangle$ such that $op \in Privs_i$
  2) **Unforgeability**: Capabilities cannot be counterfeited or corrupted.

- Note: Capabilities are (typically) transferable
Capabilities

• Advantages:
  • Eliminates confused deputy problems
  • Natural approach for user-defined objects

• Disadvantages:
  • Review of permissions?
  • Delegation?
  • Revocation?
  • Privacy?
C-Lists

- OS maintains and stores list of capabilities \( C_i = \langle O_i, Privs_i \rangle \) for each principal (process)
  1) Authorization: OS mediates access to objects, checks process capabilities
  2) Unforgeability: capabilities are stored in protected memory region (kernel memory)
Example: File Descriptor Table

• In Unix etc, a file descriptor is a handle used to reference files and I/O resources.

• File descriptors have modes (read, write) and are stored in per-process file descriptor table.

• File descriptors can be passed between processes using sendmsg()
Example: Google Fuchsia

- new OS in development by Google
- possibly intended as a universal across-platform OS for the IoT era (lots of speculation)
- capability-based microkernel embraces capabilities (handles) for all kernel objects
  - socket, port, virtual memory region, process, thread, etc.
Cryptographically-protected capabilities

• Object owner creates capabilities using a digital signature scheme
• Capabilities are triples $C = \langle O, Privs, \text{Sig}(O, Privs; k) \rangle$
• **Authorization:** $P$ is permitted to perform $op$ on $O$ if $P$ produces a capability for $O$ with $op \in Privs$ and a valid signature
• **Unforgeability:** digital signatures are unforgeable to adversaries who don't know private key $k$
• Note: assumes PKI
Example: OAuth2

- Industry standard authorization protocol
- Used for single sign-on by major IDPs
  - Facebook, Google
- The token may denote an identifier or data + signature
- Facebook tokens confer permissions for various user data (e.g. public_profile, user_friends, user_posts, user_likes)
Restricted Delegation?
Revocation

- **Revocation Tags**
  - Capabilities are tuples $C = \langle O, Privs, rt_c, \text{Sig}(O, Privs, rt_c; k) \rangle$
  - Access to object $O$ is guarded by a reference monitor; monitor maintains a list of revoked tags $rt_c$

- **Capability Chains**
  - Objects can be other capabilities!
  - $P$ is authorized to perform $op$ on $O$ if $P$ holds a capability $C_i$ and $op \in Privs_k$ holds for every capability $C_k$ in the chain from $C_i$ to $C_1$
Keys as capabilities

- Encrypt object
- Decryption method functions as reference monitor:
  - **Authorization:** correct key will decrypt object -> allow access
  - **Unforgeability:** incorrect key will not decrypt
- Note: no notion of separate privileges
Example: Mac keychains

- OSX/iOS password manager
- uses password-based encryption (AES-256) to store username/password credentials
- supports multiple keychains
Example: CryptDB

- Encrypted database system. Inspiration for several application-grade encrypted database systems
- Processes database queries on encrypted data
- Uses chains of keys (starting with user password) to decrypt values/authorize users
  - onion encryption
Attribute-based encryption

- Type of public-key encryption in which secret keys depend on user attributes
- Users can only decrypt a ciphertext if they hold a key for appropriate attributes
- A KDC creates secret keys for users based on attributes
What about privacy?