Formal methods and access control

Dr. Hale
University of Nebraska at Omaha
Information Security and Policy– Lecture 8
Today’s topics:

Access control basics

Access Control Model
Matrix and protection states
Access control lists and capability model
Looking at access control…
Definition

A state of access control is said to be safe if no permission can be leaked to an unauthorized or uninvited individual.

• Access control systems come with a wide variety of features and administrative capabilities.

• Security models are formal presentations of the security policy enforced by the access control system and are useful for proving theoretical limitations of a system.
Types of Access Control Policies

- **Discretionary Access Control (DAC, IBAC)**
  - individual user sets access control mechanisms to allow or deny access to an object
  - Based on identity of subject and object involved
  - e.g. Diary

- **Mandatory Access Control (MAC)**
  - system controls access to objects and individual cannot alter that access
  - e.g. public court information, military systems

- **Originator Controlled Access Control (ORCON)**
  - originator (creator) of information controls who can access and disseminate information, not the owner
  - e.g. NDAs on code changes, licensing agreements

- **Role Based Access Control (RBAC)**
  - access control decisions based on the a user’s role in an Organization
  - Roles may be expressed hierarchically
  - Can implement DAC and MAC

- **Attributed Based Access Control**
  - logical access control based on collections of attributes of objects and users
  - authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes

- **Others exist that are domain specific or are used for solutions to specific access problem**
Access Control Models

- Regulate the logical access to information with the system
- Maintained by a collection of policies and enforcement mechanisms
- 4 processes that build on each other:
  - identification: Obtain the identity of the entity requesting access
  - authentication: Confirm the identity of the entity
  - authorization: Determine which actions the entity can perform
  - accountability: Document the activities of the entity and system
- Built on principles for
  - Least privilege – minimum access required for duties
  - Need to know – specific data at specific times
  - Separation of duties – segregating access responsibilities to limit powers
Definition

Access *control lists, matrices, and capability tables* are mechanisms that govern the rights and privileges of users

- Can control access to file storage systems, object brokers, or other network communications devices.

A *capability table* specifies which subjects and objects that users or groups can access

- Often considered user profiles or user policies
- Can take the form of complex matrices
Access Control Tables

• **Restrict access according to user, time, duration, and file to regulate the following**
  – Who can use the system
  – What authorized users can access the system
  – Where authorized users can access the system from
  – When authorized users can access the system
  – How authorized users can access the system

• **Administrators assign user privileges as rights**

• **Rights can include**
  – Generic access (read, write, execute)
  – Domain specific
  – Functions that determine rights given the current state or historical access or states
  – Functions that determine rights given other current rights
Access Control Matrix

• Tool to describe current protection state
  – Privileges possessed by subjects (active entity) with respect to other entities
    • State transitions change elements of matrix
      – Matrix evolves by the autonomous activities of the subjects
    • The set of protection states of the system is represented by the triple (S, O, A) where S is the set of Subjects, O is the set of Objects, and A is the matrix of rights
  – Relies on an authorization scheme
    • Rules that direct how the protection state can be changed
Access Control Matrix as an Abstract Model of the Protection State

<table>
<thead>
<tr>
<th>Subjects (S)</th>
<th>Objects (O)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>o_1, ..., o_m, s_1, ..., s_n</td>
</tr>
<tr>
<td>s_2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>s_n</td>
<td></td>
</tr>
</tbody>
</table>

- **Subjects** $S = \{ s_1, \ldots, s_n \}$
  - each are subjects and objects that own themselves
- **Objects** $O = \{ o_1, \ldots, o_m \}$
  - *Could be devices, processes, messages, systems*
  - Subjects are objects (active) but not vice versa
- **Rights** $R = \{ r_1, \ldots, r_k \}$
  - $r$ (read), $w$ (write), $x$ (execute), $a$ (append), $o$ (own)
  - meaning of a right may vary depending on the object involved
- **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 $o_j$

*can think of $R$ in terms of reachability as well (a different $R$, from before)*
Access Control by Boolean Expression Evaluation

• ACM controls access to objects
  – Objects are records and fields
  – Subjects are authorized users with attributes
  – Verbs define type of access (rights)
  – Rules associated with objects, verb pair

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

- **Subject (s) Abe**
  - role (clerk), group (courthouse)
- **Verb (activity) sign**
  - Default: Deny
- **Object tax-doc**
  - Access Rule for tax-doc
    - sign: ‘clerk’ in s.role and ‘courthouse’ in s.group and 0800 ≤ hour ≤ 1700 and “Monday” ≤ day ≤ “Friday”

maps to policy:
∀ s ∈ Subjects, t ∈ Times, d ∈ Days,
sign(s) (role(s) = clerk) ∧ (0800 ≤ t ≤ 1700) ∧ d ∈ {M, T, W, Th, F}
Access Control Matrix for Abe

• Protection state changes according to hour and day

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>tax_doc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• At 1am on Monday

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Abe</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• At 3pm on Wednesday

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sign</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• At 3pm on Saturday

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Abe</td>
<td></td>
</tr>
</tbody>
</table>

Access Control
State Transitions

• Change the protection state of system –
  – $X_0 = (S_0, O_0, A_0)$ be the initial state
  – $T = [\tau_1, \tau_2, \ldots]$ commands
• Commands are transformation procedures that follow the authorization scheme
  • Change the triple
    – Alter subject or object set based on $\tau$
    – Change entries in the access control matrix rights
  • Use parameters to state how the change is made
• Given the initial state and the authorization scheme, it is a formal process to characterize all of the protection states that are reachable
Primitive Commands, $\tau$

• To maintain proper logical values for pre- and post-conditions
  – Protection before state: (S,O,A)
  – Protection after state: (S', O', A')

• **create subject** $s$
  – Creates new **row** and **column** in ACM, but does not alter rights
  – Precondition (subject does not exist): $s \not\in S$
  – Postconditions:
    $S' = S \cup \{ s \} \land$
    $O' = O \cup \{ s \} \land$
    $(\forall y \in O)[a'[s, y] = \emptyset] \land$
    $(\forall x \in S)[a'[x, s] = \emptyset] \land$
    $a'[s, s] = \{ "own" \} \land$
    $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$
    [subject exists]
    [subject object exists]
    [initialize access to all objects to null, i.e. deny]
    [ensure no other subject has access to the new subject object]
    [establish ownership of self]
    [everything else stays the same as it was before]

• **create object** $o$
  – Creates new **column** in ACM, but does not alter rights

• **destroy subject** $s$
  – Deletes **row**, **column** from ACM

• **destroy object** $o$
  – Deletes **column** from ACM

Access Control
Sample Command Logic

- Allows for provability
- enter $r$ into $A[s, o]$
  - Adds $r$ rights for subject $s$ over object $o$
  - Precondition: $s \in S$, $o \in O$
  - Postconditions:
    - $S' = S \land O' = O \land$
    - $a'[s, o] = a[s, o] \cup \{r\} \land$
    - $(\forall x \in S' \land \forall y \in O' - \{o\} \land a'[x, y] = a[x, y]) \land$
    - $(\forall x \in S' - \{s\} \land \forall y \in O') \land a'[x, y] = a[x, y]$
- delete $r$ from $A[s, o]$
  - Removes $r$ rights from subject $s$ over object $o$
- Make subject $p$ the owner of file $g$
  - command make-owner$(p, g)$
    - enter own into $A[p, g]$;
  - end
- Conditional commands
  - Let $p$ give $q r$ and $w$ rights over $f$, if $p$ owns $f$ and $p$ has copy $(c)$ rights over $q$
    - command grant-read-file$(p, f, q)$
      - 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

Access Control
Copy Rights

- Allows possessor to give rights to another
- Often attached to only the applicable right
  - r is read right that cannot be copied
  - rc is read right that can be copied
- Depending on the model, the copy flag may copied when giving r rights
Owning Rights

• Usually the possessor (owner) can change entries in ACM column by adding and deleting rights for others with respect to that object
  – 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

Principle: Attenuation of Privilege

• says you can’t give rights you do not possess
  – Restricts addition of rights within a system
  – Usually *ignored* for owner since owner gives self rights, gives them to others, deletes self rights.

Access Control
Two Approaches

• ACL – Access Control List for specifying object access
• Capability Lists - for specifying subject capabilities
Access Control Lists

- Uses the columns of access control matrix
- ACLs:
  - $Obj_1$: \{(Allen, rwxo) (Bea, rx) (Cody, rx)\}
  - $Obj_2$: \{(Allen, r) (Bea, rwo) (Cody, r)\}
  - $Obj_3$: \{(Allen, rw) (Cody, rwo)\}
- The normal use is if not named, no rights over file
  - Based on Principle of Fail-Safe Defaults
  - Extended to composed policies

<table>
<thead>
<tr>
<th></th>
<th>$Obj_1$</th>
<th>$Obj_2$</th>
<th>$Obj_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allen</td>
<td>rwxo</td>
<td>r</td>
<td>rw</td>
</tr>
<tr>
<td>Bea</td>
<td>rx</td>
<td>rwo</td>
<td></td>
</tr>
<tr>
<td>Cody</td>
<td>rx</td>
<td>r</td>
<td>rwo</td>
</tr>
</tbody>
</table>
ACL Usage

• Who can modify the ACL?
  – Creator is given *own* right for modification
  – Can be a something available like a copy flag that allows a right to be transferred, so ownership not needed

• ACL application to privileged users varies across vendors and with respect to abbreviated or full blown entries

• Denying access
  – If ACL entry denies user access, then deny access
  – If the user is not in file’s ACL nor in any group named in file’s ACL then deny access
  – If there are conflicts, the norm is to deny access if any entry denies access

Access Control
Capability Lists

- Rows of access control matrix
- C-Lists:
  - Allen: \{ (\text{Obj} \_1, \text{rwxo}) (\text{Obj} \_2, \text{r}) (\text{Obj} \_3, \text{rw}) \} 
  - Bea: \{ (\text{Obj} \_1, \text{rx}) (\text{Obj} \_2, \text{rwo}) \} 
  - Cody: \{ (\text{Obj} \_1, \text{rx}) (\text{Obj} \_2, \text{r}) (\text{Obj} \_3, \text{rwo}) \} 

<table>
<thead>
<tr>
<th></th>
<th>\text{Obj} _1</th>
<th>\text{Obj} _2</th>
<th>\text{Obj} _3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allen</td>
<td>rwxo</td>
<td>r</td>
<td>rw</td>
</tr>
<tr>
<td>Bea</td>
<td>rx</td>
<td>rwo</td>
<td></td>
</tr>
<tr>
<td>Cody</td>
<td>rx</td>
<td>r</td>
<td>rwo</td>
</tr>
</tbody>
</table>
ACLs vs. Capabilities

• Theoretically equivalent
  1. Given a subject, what objects can it access, and how? *(answered by C-Lists)*
  2. Given an object, what subjects can access it, and how? *(answered by ACLs)*

• Second question has in past been of most interest making ACL-based emerge as more common

• First question becomes more important for incident response
None
Matt Hale, PhD
University of Nebraska at Omaha
Interdisciplinary Informatics
mlhale@unomaha.edu
Twitter: @mlhale_