April 5: ACM and Safety

- Protection State Transitions
  - Commands
  - Conditional Commands
- Special Rights
  - Principle of Attenuation of Privilege
- Harrison-Ruzzo-Ullman result
  - Corollaries
State Transitions

- Change the protection state of system
- \( \vdash \) represents transition
  - \( X_i \vdash_\tau X_{i+1} \): command \( \tau \) moves system from state \( X_i \) to \( X_{i+1} \)
  - \( X_i \vdash^* Y \): 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: \texttt{create object } \( o \)
• Postconditions:
  - \( S' = S, \quad O' = O \cup \{o\} \)
  - \((\forall x \in S')[a'[x, o] = \varnothing]\)
  - \((\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]\)
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: \textbf{destroy subject} \( s \)
- Postconditions:
  
  - \( S' = S - \{ s \}, \quad O' = O - \{ s \} \)
  
  - \((\forall y \in O') [a'[s, y] = \emptyset], \quad (\forall x \in S') [a'[x, s] = \emptyset]\)
  
  - \((\forall x \in S')(\forall y \in O') [a'[x, y] = a[x, y]]\)
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\-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 $g$
  
  command `make·owner(p, g)`
  
  enter own into $A[p, g]$;

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

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

```plaintext
command grant\cdot read\cdot file\cdot 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 $q$

```plaintext
command grant·read·file·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 Right

- Allows possessor to give rights to another
- Often attached to a right, 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 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.
Key Points

• Access control matrix simplest abstraction mechanism for representing protection state
• Transitions alter protection state
• 6 primitive operations alter matrix
  – Transitions can be expressed as commands composed of these operations and, possibly, conditions
Foundational Results

• Overview
• Harrison-Ruzzo-Ullman result
  – Corollaries
• Take-Grant Protection Model
• SPM and successors
• Expressiveness and comparing model properties
Overview

- Safety Question
- HRU Model
- Take-Grant Protection Model
- SPM, ESPM
  - Multiparent joint creation
- Expressive power
- Typed Access Matrix Model
- Comparing properties of models
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 \textit{initially}
  – Alternately: not present \textit{in the previous state}

• If a system $S$, beginning in initial state $s_0$, cannot leak right $r$, it is \textit{safe with respect to the right $r$}.
Safety Question

• 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 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 creates into one

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