PURDUE UNIVERSITY

# CS526: Information Security

Chris Clifton

August 31, 2004

Access Control Matrix

CERIAS Center for Education and Research in Information Assurance and Security

---

# Description

objects (entities)

subjects

| | $o_1$ | … | $o_m$ | $s_1$ | … | $s_n$ |
|---|---|---|---|---|---|---|
| $s_1$ | | | | | | |
| $s_2$ | | | | | | |
| … | | | | | | |
| $s_n$ | | | | | | |

- Subjects $S = \{ s_1,\ldots,s_n \}$
- Objects $O = \{ o_1,\ldots,o_m \}$
- Rights $R = \{ r_1,\ldots,r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject $s_i$ has rights $r_x, \ldots, r_y$ over object $o_j$

2

# Example 2

- Procedures *inc_ctr*, *dec_ctr*, *manage*
- Variable *counter*
- Rights +, –, *call*

| | *counter* | *inc_ctr* | *dec_ctr* | *manage* |
|---|---|---|---|---|
| *inc_ctr* | + | | | |
| *dec_ctr* | – | | | |
| *manage* | | *call* | *call* | *call* |

3

# 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

4

# Example

- Subject annie
  - Attributes role (artist), groups (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 < 5

5

# Protection State Transitions

- State $X_i = (S_i, O_i, A_i)$
- Transitions $\tau_i$
  - Single transition $X_i \vdash_{\tau_{i+1}} X_{i+1}$
  - Series of transitions $X \vdash^* Y$
- Access control matrix may change
  - Change command c associated with transition
  - $X_i \vdash_{c_{i+1}(p_{i+1},\ldots,p_{i+1})} X_{i+1}$
- Commands often called *transformation procedures*

6

## Special Privileges: Copy, Ownership

- Copy (or grant)
  - Possessor can extend privileges to another
- Own right
  - Possessor can change their own privileges
- Principle of Attenuation of Privilege
  - A subject may not give rights it does not possess

8

## Primitive Commands

- Create Object *o*
  - Adds *o* to objects with no access
  - $S' = S$, $O' = O \cup \{o\}$, $(\forall x \in S')[a'[x,o] = \varnothing]$, $(\forall x \in S')(\forall y \in O)[a'[x,y] = a[x,y]]$
- Create Subject *s*
  - Adds *s* to objects, subjects, sets relevant access control to $\varnothing$
- Enter *r* into *a*[*s*,*o*]
- Delete *r* from *a*[*s*,*o*]
- Destroy subject *s*, destroy object *o*

9

## Create Subject

- Precondition: $s \notin S$
- Primitive command: **create subject** *s*
- Postconditions:
  - $S' = S \cup \{ s \}, O' = O \cup \{ s \}$
  - $(\forall y \in O')[a'[s, y] = \varnothing], (\forall x \in S')[a'[x, s] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

11

## Create Object

- Precondition: $o \notin O$
- Primitive command: **create object** *o*
- Postconditions:
  - $S' = S, O' = O \cup \{ o \}$
  - $(\forall x \in S')[a'[x, o] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

12

## Add Right

- Precondition: $s \in S$, $o \in O$
- Primitive command: enter $r$ into $a[s, o]$
- Postconditions:
  - $S' = S$, $O' = O$
  - $a'[s, o] = a[s, o] \cup \{ r \}$
  - $(\forall x,y \in S \mathrm{x} O - \{ s,o \})$   $[a'[x, y] = a[x, y]]$

13

## Delete Right

- Precondition: $s \in S$, $o \in O$
- Primitive command: **delete** $r$ **from** $a[s, o]$
- Postconditions:
  - $S' = S$, $O' = O$
  - $a'[s, o] = a[s, o] - \{ r \}$
  - $(\forall x,y \in S \mathrm{x} O - \{ s,o \})$   $[a'[x, y] = a[x, y]]$

14

# Destroy Subject

- Precondition: $s \in S$
- Primitive command: **destroy subject** *s*
- Postconditions:
  - $S' = S - \{ s \}$, $O' = O - \{ s \}$
  - $(\forall y \in O')[a'[s, y] = \varnothing]$, $(\forall x \in S')[a'[x, s] = \varnothing]$
  - $(\forall x \in S')(\forall y \in O')\ [a'[x, y] = a[x, y]]$

15

# Destroy Object

- Precondition: $o \in o$
- Primitive command: **destroy object** *o*
- Postconditions:
  - $S' = S$, $O' = O - \{ o \}$
  - $(\forall x \in S')[a'[x, o] = \varnothing]$
  - $(\forall x \in S')(\forall y \in O')\ [a'[x, y] = a[x, y]]$

16

## 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**

17

## Mono-Operational Commands

- Make process *p* the owner of file *g*
  **command** *make•owner*(*p*, *g*)
      **enter** *own* **into** *A*[*p*, *g*]**;**
  **end**
- Mono-operational command
  - Single primitive operation in this command

18

## Conditional Commands

- Let *p* give *q* *r* rights over *f*, if *p* owns *f*
  **command** *grant•read•file•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

19

## Multiple Conditions

- Let *p* give *q* *r* and *w* rights over *f*, if *p* owns *f* and *p* has *c* rights over *q*
  **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**

20

## 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

21

## 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

22

## 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.

23

## 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

24

## CS526: Information Security
## Chris Clifton

September 2, 2004

Decidability

## What is *Secure*?

- A secure system doesn't allow violations of policy
  - Is this a good definition?
  - Can we use it?
- Alternative view: based on rights
  - Start with access control matrix *A*
  - *Leak*: commands can add right *r* to an element of *A* not containing *r*
  - *Safe*: System is *safe with respect to r* if *r* cannot be leaked

26

## Formally:

- Given
  - initial state $X_0 = (S_0, O_0, A_0)$
  - Set of primitive commands *c*
- Can we reach a state $X_n$ where $\exists$s,o such that $A_n$[s,o] includes a right *r* not in $A_0$[s,o]?
  - If so, the system is not safe
  - But is "safe" secure?

*Are commands correctly implemented?*

27

## Example: Unix File System

- Access Control Matrix
  - Root has access to all files
  - Owner has access to their own files
- Safe with respect to file access right?
  - No chmod/chown command
  - Only "root" can get root privileges
  - Only user can authenticate as themselves

*Is "Safe" definition useful?*

28

## Solution: Trust

- Safety doesn't distinguish leak from authorized transfer of rights
- Subjects authorized to receive transfer of rights deemed "trusted"
  – Eliminate trusted subjects from matrix

29

## Decidability Result *(Harrison, Ruzzo, Ullman)*

- Given a system where each command consists of a single *primitive* command, There exists an algorithm that will determine if a protection system with initial state $X_0$ is safe with respect to right *r*.
- Proof: determine minimum commands *k* to leak
  – Delete/destroy: Can't leak (or be detected)
  – Create/enter: new subjects/objects "equal", so treat all new subjects as one
  – If *n* rights, leak possible, must be able to leak $n(|S_0|+1)(|O_0|+1)+1$ commands
- Enumerate all possible to decide

30

# Decidability: Non-Primitive Commands

- It is undecidable if a given state of a given protection system is safe for a given generic right
- Proof: Reduction from halting problem
  - Symbols, states $\Rightarrow$ rights
  - Tape cell $\Rightarrow$ subject (can create new subjects)
  - Right *own*: $s_i$ *owns* $s_i+1$ for $1 \leq i < k$
  - Cell $s_i$ $A \Rightarrow s_i$ has $A$ rights on itself
  - Cell $s_k \Rightarrow s_k$ has end rights on itself
  - State $p$, head at $s_i \Rightarrow s_i$ has $p$ rights on itself

31

# Example:

Turing Machine

| | | | | |
|---|---|---|---|---|
| *A* | *B* | *C* | *D* | … |

Matrix

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|
| $s_1$ | *A* | *own* | | |
| $s_2$ | | *B* | *own* | |
| $s_3$ | | | *C, p* | *own* |
| $s_4$ | | | | *D, end* |

32

# Mapping

| 1 | 2 | 3 | 4 | |
|---|---|---|---|---|
| A | B | X | D | … |

head

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | |
|---|---|---|---|---|---|
| $s_1$ | A | *own* | | | |
| $s_2$ | | B | *own* | | |
| $s_3$ | | | X | *own* | |
| $s_4$ | | | | D $k_1$ end | |
| | | | | | |

After $\delta(k, \mathrm{C}) = (k_1, \mathrm{X}, \mathrm{R})$ where $k$ is the current state and $k_1$ the next state

33

# Command Mapping

$\delta(k, \mathrm{C}) = (k_1, \mathrm{X}, \mathrm{R})$ at intermediate becomes

```
command ck,C(s3, s4)
if own in A[s3,s4] and k in A[s3,s3]
     and C in A[s3,s3]
then
  delete k from A[s3,s3];
  delete C from A[s3,s3];
  enter X into A[s3,s3];
  enter k1 into A[s4,s4];
end
```

34

# Commands:

- Halting problem Turing Machine: Symbols *A*, *B*; states *p, q*
- $C_{p,A}(s_i, s_{i-1})$ (move left)
  - if *own* $\in a[s_{i-1}, s_i]$ and $p \in a[s_i, s_i]$ and $A \in a[s_i, s_i]$
    - Delete *p* from $a[s_i, s_i]$, *A* from $a[s_i, s_i]$
    - Enter *B* into $a[s_i, s_i]$, *q* into $a[s_{i-1}, s_{i-1}]$
- Similar commands for move right, move right at end of tape
- Simulates Turing machine
  - Leaks halting state $\Rightarrow$ halting state in the matrix $\Rightarrow$ Halting state reached

*This is undecidable!*

35

# Mapping

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| A | B | X | Y | *b* |

head

$\Rightarrow$

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|
| $s_1$ | A | *own* | | | |
| $s_2$ | | B | *own* | | |
| $s_3$ | | | X | *own* | |
| $s_4$ | | | | Y | *own* |
| $s_5$ | | | | | *b* $k_2$ end |

After $\delta(k_1, \text{D}) = (k_2, \text{Y}, \text{R})$ where $k_1$ is the current state and $k_2$ the next state

36

## Command Mapping

δ($k_1$, D) = ($k_2$, Y, R) at end becomes

```
command crightmost_{k,C}(s_4, s_5)
if end in A[s_4,s_4] and k_1 in A[s_4,s_4]
      and D in A[s_4,s_4]
then
  delete end from A[s_4,s_4];
  create subject s_5;
  enter own into A[s_4,s_5];
  enter end into A[s_5,s_5];
  delete k_1 from A[s_4,s_4];
  delete D from A[s_4,s_4];
  enter Y into A[s_4,s_4];
  enter k_2 into A[s_5,s_5];
end
```

37

## Rest of Proof

- Protection system exactly simulates a TM
  - Exactly 1 *end* right in ACM
  - 1 right in entries corresponds to state
  - Thus, at most 1 applicable command
- If TM enters state $q_f$, then right has leaked
- If safety question decidable, then represent TM as above and determine if $q_f$ leaks
  - Implies halting problem decidable
- Conclusion: safety question undecidable

38

## Other Results
## *(most from the same authors)*

- Set of unsafe systems recursively enumerable
- Without create primitive, safety in P-SPACE
  - Like halting problem reduction, but no unlimited tape
- Without delete/destroy, still undecidable
  - Decidable if at most one condition allowed per command
  - Still holds if delete allowed

39

## Where does this leave us?

- Safety decidable for some models
  - Are they practical?
- Safety only works if maximum rights known in advance
  - Policy must specify all rights someone could get, not just what they have
  - Where might this make sense?
- Next: Example of a decidable model
  - Take-Grant Protection Model

40

# Mono-Operational Commands

- Answer: *yes*
- Sketch of proof:

Consider minimal sequence of commands $c_1$, …, $c_k$ to leak the right.

–Can omit **delete**, **destroy**

–Can merge all **create**s into one

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

41