Foundational Results

- Safety Question
- HRU Model
- Take-Grant Protection Model
- Expressive power
- Typed Access Matrix Model

What Is "Secure"?

- Giving a generic right r to a subject who did not initially possessed it is called "leaking"
- If a system S, beginning in initial state s_0, cannot leak right r, it is *safe* with respect to the right r.
- Leaking a right is not inherently bad
  - Legitimate transfer of rights by owner
- Safety Question
  - Does there exist an algorithm for determining whether a protection system S with initial state s_0 is *safe* with respect to a generic right r?

Formally:

- Given
  - initial state $X_0 = (S_0, O_0, A_0)$
  - Set of primitive commands c
- Can we reach a state X_n $(X_0 \vdash^* X_n)$ where $\exists\ s \in S$ and $\exists\ o \in O$ 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 a "safe" system a secure system?
    Are commands correctly implemented?

3

Trust

- Safety does not distinguish a leak of a right from an authorized transfer of rights
- Subjects authorized to receive transfer of rights deemed "trusted"
  - Eliminate trusted subjects from matrix
- Trivial cases of safety
  - r = *read*, *own* $\in a[s,o]$, command *can•grant•read•if•own*
  - No command includes the *enter* primitive command

How about the general case?

4

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 **create**s into one (since new subjects are all equal)

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

5

General Case

- Answer: *no*
- Sketch of proof:

  Reduce halting problem to safety problem

  Turing Machine review:

  - Infinite tape in one direction
  - States K, symbols M; distinguished blank b
  - Transition function $\delta(k, m) = (k', m', \text{L})$ means in state k, symbol m on tape location replaced by symbol m', head moves to left one square, and enters state k'
  - Halting state is q_f; TM halts when it enters this state

6

Mapping

1	2	3	4	
A	B	C	D	…

head

Current state is k

	s_1	s_2	s_3	s_4	
s_1	A	*own*			
s_2		B	*own*		
s_3			C k	*own*	
s_4				D end	

7

Mapping

1	2	3	4	
A	B	X	D	…

head

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

	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	

8

Command Mapping

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

```
command c_{k,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
```


Mapping

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

head

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

Command Mapping

$\delta(k_1, \mathrm{D}) = (k_2, \mathrm{Y}, \mathrm{R})$ 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
```


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

Other Results

- Set of unsafe systems is recursively enumerable
- Delete **create** primitive; then safety question is complete in **P-SPACE**
- Delete **destroy**, **delete** primitives (this system is called monotonic): safety question is undecidable
- Safety question for monoconditional, monotonic protection systems is decidable
- Safety question for monoconditional protection systems with **create**, **enter**, **delete** (and no **destroy**) is decidable.

13

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
- Can the safety of a particular system, with specific rules, be established?

14

Take-Grant Protection Model

- A specific (not generic) system
  - System represented as a directed graph
  - Set of graph rewriting rules for state transitions
- Safety is decidable, and in time linear with the size of the system
- Goal: find conditions under which rights can be transferred from one entity to another in the system

15

System

o objects (files, …)

● subjects (users, processes, …)

⊗ don't care (either a subject or an object)

$G \vdash_x G'$ apply a rewriting rule x (witness) to G to get G'

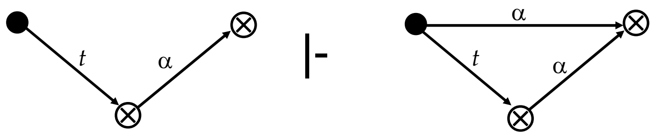
$G \vdash^* G'$ apply a sequence of rewriting rules (witness) to G to get G'

$R = \{ t, g, r, w, \ldots \}$ set of rights

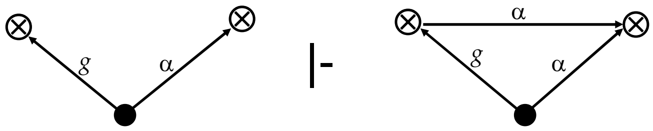
16

Rules

take



grant

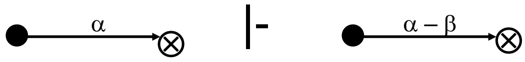


17

More Rules

create

remove



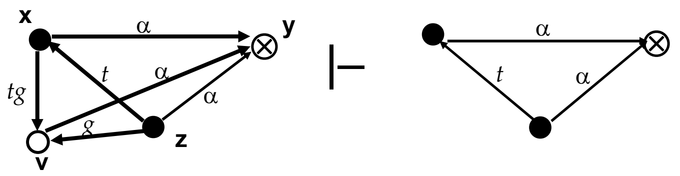
These four rules are called the *de jure* rules

18

Example: Shared Buffer

- Initially s has grant rights for processes p and q.
- s sets up a shared buffer for p,q with the following steps
  - s creates new object b
  - s grants ({r,w} to b) to p
  - s grants ({r,w} to b) to q

Symmetry



1. **x** creates (*tg* to new) **v**
2. **z** takes (*g* to **v**) from **x**
3. **z** grants (α to **y**) to **v**
4. **x** takes (α to **y**) from **v**

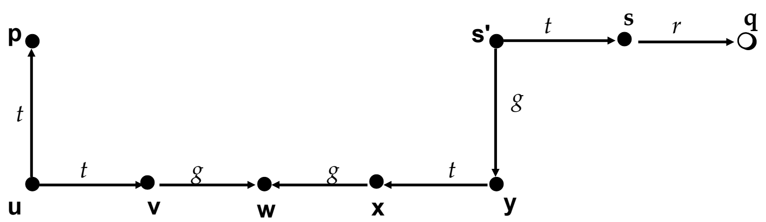
Similar result for grant

Islands

- *tg*-path: path of distinct vertices connected by edges labeled *t* or *g*
  - Call them "tg-connected"
- island: maximal *tg*-connected subject-only subgraph
  - Any right one vertex has can be shared with any other vertex

21

Example



22

can•share Predicate

Definition:

can•share(r, **x**, **y**, G_0) if, and only if, there is a sequence of protection graphs G_0, …, G_n such that $G_0 \vdash^* G_n$ using only *de jure* rules and in G_n there is an edge from **x** to **y** labeled r.

23

can•share Properties

- If x and y are subjects in an island, then *can•share*(r, **x**, **y**, G_0)
  - Proof by induction using the properties of tg-connected subjects
- General result: *can•share*(r, **x**, **y**, G_0) is decidable using an algorithm of complexity O(|V| + |E|) where V and E are the vertices and edges in the graph
  - Proof omitted (Exercise)

24

Key Question

- Characterize class of models for which safety is decidable
  - Existence: Take-Grant Protection Model is a member of such a class
  - Universality: in general, the question undecidable, so for some models it is not decidable
- What is the dividing line?

25

Typed Access Matrix Model

- Like ACM, but with set of types T
  - All subjects, objects have types
  - Set of types for subjects TS
- Protection state is (S, O, τ, A)
  - $\tau: O \rightarrow T$ specifies type of each object
  - If **X** subject, τ(**X**) in TS
  - If **X** object, τ(**X**) in $T - TS$
- Same rules as ACM except for create

26

Create Rules

- Subject creation
  - **create subject** *s* **of type** *ts*
  - *s* must not exist as subject or object when operation executed
  - $ts \in TS$
- Object creation
  - **create object** *o* **of type** *to*
  - *o* must not exist as object when operation executed
  - $to \in T - TS$

Create Subject

- Precondition: $s \notin S$
- Primitive command: **create subject** *s* **of type** *t*
- Postconditions:
  - $S' = S \cup \{ s \}$, $O' = O \cup \{ s \}$
  - $(\forall y \in O)[\tau'(y) = \tau(y)]$, $\tau'(s) = t$
  - $(\forall y \in O')[a'[s, y] = \varnothing]$, $(\forall x \in S')[a'[x, s] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

Create Object

- Precondition: $o \notin O$
- Primitive command: **create object** *o* **of type** *t*
- Postconditions:
  - $S' = S, O' = O \cup \{ o \}$
  - $(\forall y \in O)[\tau'(y) = \tau(y)], \tau'(o) = t$
  - $(\forall x \in S')[a'[x, o] = \varnothing]$
  - $(\forall x \in S)(\forall y \in O)[a'[x, y] = a[x, y]]$

Definitions

- MTAM (Monotonic TAM) Model: TAM model without **delete**, **destroy**
- $\alpha(x_1{:}t_1, \ldots, x_n{:}t_n)$ create command
  - t_i child type in α if any of **create subject** x_i **of type** t_i or **create object** x_i **of type** t_i occur in body of α
  - t_i parent type otherwise

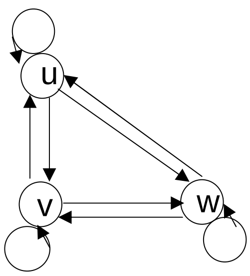
Cyclic Creates

command *havoc*($s_1 : u$, $s_2 : u$, $o_1 : v$, $o_2 : v$, $o_3 : w$, o_4 : w)
  create subject s_1 **of type** u ;
  create object o_1 **of type** v ;
  create object o_3 **of type** w ;
  enter r **into** $a[s_2, s_1]$;
  enter r **into** $a[s_2, o_2]$;
  enter r **into** $a[s_2, o_4]$;
End

What kind of types are u, v and w?

31

Creation Graph



- *u, v, w* child types
- *u, v, w* also parent types
- Graph: lines from parent types to child types
- This graph has cycles

32

Theorems

- Safety decidable for systems with acyclic MTAM schemes
- Safety for acyclic ternary MATM decidable in time polynomial in the size of the initial ACM
  - "ternary" means commands have no more than 3 parameters
  - Equivalent in expressive power to MTAM

33

Key Points

- Safety problem undecidable
- Limiting scope of systems can make problem decidable
- Types critical to safety problem's analysis

34