Overview

- Safety Question
- HRU Model
- Take-Grant Protection Model
- SPM, ESPM
  - Multiparent joint creation
- Expressive power
- Typed Access Matrix Model

What Is “Secure”?

- Adding a generic right $r$ where there was not one is “leaking”
- 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

• 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 $r$?
  – 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 $k \leq n(s+1)(o+1)$
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 $\delta(k, m) = (k', m', L)$ 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

Mapping

- Current state is $k$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>(\text{head})</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>...</td>
<td>(\text{head})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>A</td>
<td>\textit{own}</td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td>B</td>
<td>\textit{own}</td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td>C</td>
<td>$k$</td>
<td>\textit{own}</td>
</tr>
<tr>
<td>$s_4$</td>
<td></td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>
Mapping

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

Command Mapping

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

```
command $c_{k,C}(s_3, s_4)$
if own in $A[s_3, s_4]$ and $k$ in $A[s_3, s_3]$ and $C$ in $A[s_3, s_3]$
then
  delete $k$ from $A[s_3, s_3]$;
  delete $C$ from $A[s_3, s_3]$;
  enter $X$ into $A[s_3, s_3]$;
  enter $k_1$ into $A[s_4, s_4]$;
end
```
Mapping

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

Command Mapping

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

```plaintext
command crightmost, 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; then safety question is undecidable
  – Systems are monotonic
• Safety question for monoconditional, monotonic protection systems is decidable
• Safety question for monoconditional protection systems with create, enter, delete (and no destroy) is decidable.
Take-Grant Protection Model

- A specific (not generic) system
  - Set of rules for state transitions
- Safety 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

System

- objects (files, …)
- subjects (users, processes, …)
  - don't care (either a subject or an object)

\[ G \vdash \ x \ G' \quad \text{apply a rewriting rule } x \ (\text{witness}) \text{ to } G \text{ to get } G' \]
\[ G \vdash^* G' \quad \text{apply a sequence of rewriting rules (witness) to } G \text{ to get } G' \]
\[ R = \{ t, g, r, w, \ldots \} \quad \text{set of rights} \]
Rules

take

grant

More Rules

create

remove

These four rules are called the *de jure* rules
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
Initial, Terminal Spans

- initial span from $x$ to $y$: $x$ subject, $tg$-path between $x$, $y$ with word in $\{ \overline{tg} \} \cup \{ \nu \}$
  - $x$ can give rights it has to $y$
- terminal span from $x$ to $y$: $x$ subject, $tg$-path between $x$, $y$ with word in $\{ \overline{t^*} \} \cup \{ \nu \}$
  - $x$ can acquire any rights $y$ has

Bridges

- bridge: $tg$-path between subjects $x$, $y$, with associated word in
  $\{ \overline{t^*}, \overline{t^*}, \overline{tg}, \overline{t^*}, \overline{tg}, \overline{t^*} \}$
  - rights can be transferred between the two endpoints
  - not an island as intermediate vertices are objects
Example

- islands: \{ p, u \} \{ w \} \{ y, s' \}
- bridges: u, v, w; w, x, y
- initial span: p (associated word v)
- terminal span: s's (associated word t)

can•share Predicate

Definition:
- can•share(\( r, x, y, G_0 \)) if, and only if, there is a sequence of protection graphs \( G_0, \ldots, G_n \) 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 \( r \).
can\textbullet share Theorem

- can\textbullet share(r, x, y, G_0) if, and only if, there is an edge from x to y labeled r in G_0, or the following hold simultaneously:
  - There is an s in G_0 with an s-to-y edge labeled r
  - There is a subject x' = x or initially spans to x
  - There is a subject s' = s or terminally spans to s
  - There are islands I_1,\ldots, I_k connected by bridges, and x' in I_1 and s' in I_k

Outline of Proof

- s has r rights over y
- s' acquires r rights over y from s
  - Definition of terminal span
- x' acquires r rights over y from s'
  - Repeated application of sharing among vertices in islands, passing rights along bridges
- x' gives r rights over y to x
  - Definition of initial span
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, question undecidable, so for some models it is not decidable

• What is the dividing line?

Schematic Protection Model

• Type-based model
  – Protection type: entity label determining how control rights affect the entity
    • Set at creation and cannot be changed
  – Ticket: description of a single right over an entity
    • Entity has sets of tickets (called a domain)
    • Ticket is $X/r$, where $X$ is entity and $r$ right
  – Functions determine rights transfer
    • Link: are source, target “connected”?
    • Filter: is transfer of ticket authorized?
Link Predicate

• Idea: $link_i(X, Y)$ if $X$ can assert some control right over $Y$
• Conjunction or disjunction of:
  – $X/z \in dom(X)$
  – $X/z \in dom(Y)$
  – $Y/z \in dom(X)$
  – $Y/z \in dom(Y)$
  – true

Examples

• Take-Grant:
  $$link(X, Y) = Y/g \in dom(X) \lor X/t \in dom(Y)$$
• Broadcast:
  $$link(X, Y) = X/b \in dom(X)$$
• Pull:
  $$link(X, Y) = Y/p \in dom(Y)$$
Filter Function

- Range is set of copyable tickets
  - Entity type, right
- Domain is subject pairs
- Copy a ticket \( X/r:c \) from \( dom(Y) \) to \( dom(Z) \)
  - \( X/rc \in dom(Y) \)
  - \( link_i(Y, Z) \)
  - \( \tau(Y)/r:c \in f(\tau(Y), \tau(Z)) \)
- One filter function per link function

Example

- \( f(\tau(Y), \tau(Z)) = T \times R \)
  - Any ticket can be transferred (if other conditions met)
- \( f(\tau(Y), \tau(Z)) = T \times RI \)
  - Only tickets with inert rights can be transferred (if other conditions met)
- \( f(\tau(Y), \tau(Z)) = \emptyset \)
  - No tickets can be transferred
Example

• Take-Grant Protection Model
  – \( TS = \{ \text{subjects} \} \), \( TO = \{ \text{objects} \} \)
  – \( RC = \{ tc, gc \} \), \( RI = \{ rc, wc \} \)
  – \( \text{link}(p, q) = p/t \in \text{dom}(q) \lor q/t \in \text{dom}(p) \)
  – \( f(\text{subject, subject}) = \{ \text{subject, object} \} \times \{ tc, gc, rc, wc \} \)

Create Operation

• Must handle type, tickets of new entity
• Relation can\( \text{create}(a, b) \)
  – Subject of type \( a \) can create entity of type \( b \)
• Rule of acyclic creates:
Types

- $cr(a, b)$: tickets introduced when subject of type $a$ creates entity of type $b$
- **B** subject: $cr(a, b) \subseteq \{ b/r:c \in RI \}$
- **B** object: $cr(a, b) \subseteq \{ b/r:c \in RI \}$

**B** subject: $cr(a, b)$ has two parts
- $cr_P(a, b)$ added to **A**, $cr_C(a, b)$ added to **B**
- **A** gets $B/r:c$ if $b/r:c$ in $cr_P(a, b)$
- **B** gets $A/r:c$ if $a/r:c$ in $cr_C(a, b)$

Non-Distinct Types

$cr(a, a)$: who gets what?
- **self/r:c** are tickets for creator
- **a/r:c** tickets for created

$cr(a, a) = \{ a/r:c, self/r:c \mid r:c \in R \}$
### Attenuating Create Rule

$cr(a, b)$ attenuating if:

1. $cr_c(a, b) \subseteq cr_p(a, b)$ and
2. $a/r:c \in cr_p(a, b) \Rightarrow self/r:c \in cr_p(a, b)$

### Safety Result

- If the scheme is acyclic and attenuating, the safety question is decidable
Expressive Power

• How do the sets of systems that models can describe compare?
  – If HRU equivalent to SPM, SPM provides more specific answer to safety question
  – If HRU describes more systems, SPM applies only to the systems it can describe

HRU vs. SPM

• SPM more abstract
  – Analyses focus on limits of model, not details of representation
• HRU allows revocation
  – SMP has no equivalent to delete, destroy
• HRU allows multiparent creates
  – SPM cannot express multiparent creates easily, and not at all if the parents are of different types because can create allows for only one type of creator
Multiparent Create

- Solves mutual suspicion problem
  - Create proxy jointly, each gives it needed rights
- In HRU:
  
  ```
  command multicreate(s_0, s_1, o)
  if r in a[s_0, s_1] and r in a[s_1, s_0]
  then
    create object o;
    enter r into a[s_0, o];
    enter r into a[s_1, o];
  end
  ```

SPM and Multiparent Create

- can create extended in obvious way
  - cc ⊆ TS × … × TS × T
- Symbols
  - X_1, …, X_n parents, Y created
  - R_1,i, R_2,i, R_3, R_4,j ⊆ R
- Rules
  - cr_{P,i}(τ(X_1), …, τ(X_n)) = Y/R_{1,i} ∪ X/R_{2,i}
  - cr_{C}(τ(X_1), …, τ(X_n)) = Y/R_3 ∪ X_i/R_{4,1} ∪ … ∪ X_n/R_{4,n}
Example

- Anna, Bill must do something cooperatively
  - But they don’t trust each other
- Jointly create a proxy
  - Each gives proxy only necessary rights
- In ESPM:
  - Anna, Bill type $a$; proxy type $p$; right $x \in R$
  - $cc(a, a) = p$
  - $cr_{Anna}(a, a, p) = cr_{Bill}(a, a, p) = \emptyset$
  - $cr_{proxy}(a, a, p) = \{ Anna/x, Bill/x \}$