Lecture #5

• Knowing in a combined graph
• Basics of Schematic Protection Model
Definition:

- \textit{can•know}(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 \textit{de jure} or \textit{de facto} rules and in \(G_n\) there is an edge from \(x\) to \(y\) labeled \(r\).
Example

y creates (rw to new) z
x takes (r to z) from y
y passes to x through z
x takes (w to z) from y
y posts to x through z
Combined Transfers

The subject can acquire $\alpha$ rights over the last object.

The subject can acquire $r$ rights over the last object.
Combined Transfers

The subject can acquire $g$ rights over the last object.

The subject can acquire $w$ rights over the last object.
Combined Transfers

Just as rights can be transferred over a bridge, information can flow over a connection.
Theorem

can\textbullet know(p, q, G_0) holds if and only if:

(a) can\textbullet share(r, p, q, G_0) holds, or

(b) there is a sequence of subjects u_1, ..., u_n such that all of the following are true:

(i) \ p = u_1 or u_1 rw-initially spans to p;

(ii) \ q = u_n or u_n rw-terminally spans to q; and

(iii) for all i, 1 \leq i < n, there is an rwtg-path between u_i and u_{i+1} with associated word a bridge or connection
can\textbullet{}know(p, q, G_0) holds:

• take n = 2, u_1 = x, and u_2 = y
  
  – p = u_1 or u_1 rw-initially spans to p;
  – q = u_2 or u_2 rw-terminally spans to q; and
  – there is an rw\textbullet{}tg-path between u_1 and u_2 with associated word a bridge or connection

• u_1, u_2 connected with a t edge
Final Example

can\(\cdot\)share\((r, v, z, G_0)\) is false
- no initial span between \(v\) and any subject

can\(\cdot\)know\((v, z, G_0)\) is true
- \(u_1 = w, u_2 = x\)
- \(u_1\ r\w\)-initially spans to \(y\)
- \(u_2\ r\w\)-terminally spans to \(z\)
- there is a bridge between \(u_1\) and \(u_2\)
Final Example Witness

x takes (r to z) from y
w spies on z through x
w passes from z to v

x takes (r to z) from y
w passes from x to v
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 of 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:
  \[ \text{link}(X, Y) = Y/g \in \text{dom}(X) \lor X/t \in \text{dom}(Y) \]

• Broadcast:
  \[ \text{link}(X, Y) = X/b \in \text{dom}(X) \]

• Pull:
  \[ \text{link}(X, Y) = Y/p \in \text{dom}(Y) \]
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 $dom(Z)$
  – $X/rc \in dom(Y)$
  – $link_{i}(Y, Z)$
  – $\tau(Y)/r:c \in f_{i}(\tau(Y), \tau(Z))$
• One filter function per link predicate
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
  - $TS = \{ \text{subjects} \}$, $TO = \{ \text{objects} \}$
  - $RC = \{ tc, gc \}$, $RI = \{ rc, wc \}$
  - $\text{link}(p, q) = p/t \in \text{dom}(q) \lor q/g \in \text{dom}(p)$
  - $f(\text{subject, subject}) = \{ \text{subject, object} \} \times \{ tc, gc, rc, wc \}$
Create Operation

• Must handle type, tickets of new entity
• Relation $cc(a, b)$ [cc for *can-create*]
  – Subject of type $a$ can create entity of type $b$
• Rule of acyclic creates:

\[
\begin{align*}
& a 
\quad \rightarrow \quad b \\
& c 
\quad \rightarrow \quad d
\end{align*}
\]

\[
\begin{align*}
& a 
\quad \rightarrow \quad b \\
& c 
\quad \rightarrow \quad d
\end{align*}
\]
Types

- \( cr(a, b) \): tickets created when subject of type \( a \) creates entity of type \( b \) [\( cr \) for create-rule]
- \( \mathbf{B} \) object: \( cr(a, b) \subseteq \{ b/r:c \in RI \} \)
  - \( \mathbf{A} \) gets \( \mathbf{B}/r:c \) iff \( b/r:c \in cr(a, b) \)
- \( \mathbf{B} \) subject: \( cr(a, b) \) has two subsets
  - \( cr_p(a, b) \) added to \( \mathbf{A} \), \( cr_c(a, b) \) added to \( \mathbf{B} \)
  - \( \mathbf{A} \) gets \( \mathbf{B}/r:c \) if \( b/r:c \in cr_p(a, b) \)
  - \( \mathbf{B} \) gets \( \mathbf{A}/r:c \) if \( a/r:c \in cr_c(a, b) \)
Non-Distinct Types

\[ cr(a, a) \]: who gets what?

- \( \text{self}/r:c \) are tickets for creator
- \( a/r:c \) tickets for created

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

\( cr(a, b) \) attenuating if:

1. \( cr_C(a, b) \subseteq cr_P(a, b) \) and
2. \( a/r:c \in cr_P(a, b) \Rightarrow self/r:c \in cr_P(a, b) \)
Example: Owner-Based Policy

- Users can create files, creator can give itself any inert rights over file
  - \( cc = \{ (user, file) \} \)
  - \( cr(user, file) = \{ file/r:c | r \in RI \} \)
- Attenuating, as graph is acyclic, loop free
Example: Take-Grant

- Say subjects create subjects (type $s$), objects (type $o$), but get only inert rights over latter
  - $cc = \{ (s, s), (s, o) \}$
  - $cr_C(a, b) = \emptyset$
  - $cr_P(s, s) = \{s/tc, s/gc, s/rc, s/wc\}$
  - $cr_P(s, o) = \{s/rc, s/wc\}$
- Not attenuating, as no self tickets provided; subject creates subject
Safety Analysis

- Goal: identify types of policies with tractable safety analyses
- Approach: derive a state in which additional entries, rights do not affect the analysis; then analyze this state
  - Called a *maximal state*
Definitions

- System begins at initial state
- Authorized operation causes legal transition
- Sequence of legal transitions moves system into final state
  - This sequence is a history
  - Final state is derivable from history, initial state
More Definitions

• States represented by $h$
• Set of subjects $SUB^h$, entities $ENT^h$
• Link relation in context of state $h\ link^h$
• Dom relation in context of state $h\ dom^h$
\textbf{\textit{path}^h(X, Y)}

- \(X, Y\) connected by one link or a sequence of links
- Formally, either of these hold:
  - for some \(i\), \(\text{link}_i^h(X, Y)\); or
  - there is a sequence of subjects \(X_0, \ldots, X_n\) such that \(\text{link}_i^h(X, X_0), \text{link}_i^h(X_n, Y)\), and for \(k = 1, \ldots, n\), \(\text{link}_i^h(X_{k-1}, X_k)\)
- If multiple such paths, refer to \(\text{path}_j^h(X, Y)\)
Capacity \( \text{cap}(\text{path}^h(X,Y)) \)

- Set of tickets that can flow over \( \text{path}^h(X,Y) \)
  - If \( \text{link}_{i}^h(X,Y) \): set of tickets that can be copied over the link (i.e., \( f_{i}(\tau(X), \tau(Y)) \))
  - Otherwise, set of tickets that can be copied over all links in the sequence of links making up the \( \text{path}^h(X,Y) \)
- Note: all tickets (except those for the final link) \textit{must} be copyable
Flow Function

- Idea: capture flow of tickets around a given state of the system
- Let there be \( m \) paths between subjects \( X \) and \( Y \) in state \( h \). Then flow function

\[
\text{flow}^h: \text{SUB}^h \times \text{SUB}^h \to 2^{T \times R}
\]

is:

\[
\text{flow}^h(X,Y) = \bigcup_{i=1,\ldots,m} \text{cap}(path_i^h(X,Y))
\]
Properties of Maximal State

- Maximizes flow between all pairs of subjects
  - State is called *
  - Ticket in $flow^*(X,Y)$ means there exists a sequence of operations that can copy the ticket from $X$ to $Y$

- Questions
  - Is maximal state unique?
  - Does every system have one?