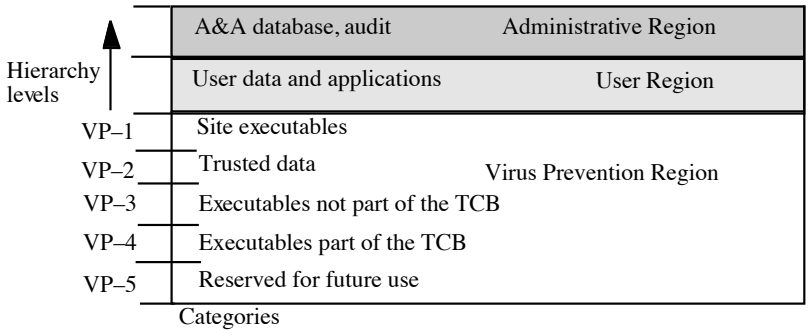
DG/UX System

- Provides mandatory access controls
  - MAC label identifies security level
  - Default labels, but can define others
- Initially
  - Subjects assigned MAC label of parent
    - Initial label assigned to user, kept in Authorization and Authentication database
  - Object assigned label at creation
    - Explicit labels stored as part of attributes
    - Implicit labels determined from parent directory

MAC Regions



IMPL_HI is “maximum” (least upper bound) of all levels
IMPL_LO is “minimum” (greatest lower bound) of all levels

Directory Problem

- Process p at MAC_A tries to create file */tmp/x*
- */tmp/x* exists but has MAC label MAC_B
  - Assume MAC_B dom MAC_A
- Create fails
  - Now p knows a file named x with a higher label exists
- Fix: only programs with same MAC label as directory can create files in the directory
  - Now compilation won't work, mail can't be delivered

Multilevel Directory

- Directory with a set of subdirectories, one per label
  - Not normally visible to user
  - p creating /tmp/x actually creates /tmp/d/x where d is directory corresponding to MAC_A
  - All p's references to /tmp go to /tmp/d
- p cd's to /tmp/a, then to ..
  - System call stat(".", &buf) returns inode number of real directory
  - System call dg_stat(".", &buf) returns inode of /tmp

Object Labels

- Requirement: every file system object must have MAC label

1. Roots of file systems have explicit MAC labels
   - If mounted file system has no label, it gets label of mount point
2. Object with implicit MAC label inherits label of parent

Object Labels

- Problem: object has two names
  - /x/y/z, /a/b/c refer to same object
  - y has explicit label IMPL_HI, b has explicit label IMPL_B
- Case 1: hard link created while file system on DG/UX system

3. Creating hard link requires explicit label
   - If implicit, label made explicit
   - Moving a file makes label explicit

Object Labels

- Case 2: hard link exists when file system mounted
  - No objects on paths have explicit labels: paths have same *implicit* labels
  - An object on path acquires an explicit label: implicit label of child must be preserved

4. Change to directory label makes child labels explicit *before* the change

Object Labels

- Symbolic links are files, and treated as such

5. When resolving symbolic link, label of object is label of target of the link
   - System needs access to the symbolic link itself

Using MAC Labels

- Simple security condition implemented
- *-property not fully implemented
  - Process MAC must equal object MAC
  - Writing allowed only at same security level
- Overly restrictive in practice

MAC Tuples

- Up to 3 MAC ranges (one per region)
- MAC range is a set of labels with upper, lower bound
  - Upper bound must dominate lower bound of range
- Examples
  1. [(Secret, {NUC}), (Top Secret, {NUC})]
  2. [(Secret, ∅), (Top Secret, {NUC, EUR, ASI})]
  3. [(Confidential, {ASI}), (Secret, {NUC, ASI})]

MAC Ranges

1. [(Secret, {NUC}), (Top Secret, {NUC})]
2. [(Secret, ∅), (Top Secret, {NUC, EUR, ASI})]
3. [(Confidential, {ASI}), (Secret, {NUC, ASI})]

- (Top Secret, {NUC}) in ranges 1, 2
- (Secret, {NUC, ASI}) in ranges 2, 3
- [(Secret, {ASI}), (Top Secret, {EUR})] not valid range
  - as (Top Secret, {EUR}) ¬*dom* (Secret, {ASI})

Objects and Tuples

- Objects must have MAC labels
  - May also have MAC label
  - If both, tuple overrides label
- Example
  - Paper has MAC range:
    [(Secret, {EUR}), (Top Secret, {NUC, EUR})]

MAC Tuples

- Process can read object when:
  - Object MAC range (*lr*, *hr*); process MAC label *pl*
  - *pl dom hr*
    - Process MAC label grants read access to upper bound of range
- Example
  - Peter, with label (Secret, {EUR}), cannot read paper
    - (Top Secret, {NUC, EUR}) *dom* (Secret, {EUR})
  - Paul, with label (Top Secret, {NUC, EUR, ASI}) can read paper
    - (Top Secret, {NUC, EUR, ASI}) *dom* (Top Secret, {NUC, EUR})

MAC Tuples

- Process can write object when:
  - Object MAC range (*lr*, *hr*); process MAC label *pl*
  - *pl* ∈ (*lr*, *hr*)
    - Process MAC label grants write access to any label in range
- Example
  - Peter, with label (Secret, {EUR}), can write paper
    - (Top Secret, {NUC, EUR}) *dom* (Secret, {EUR}) and (Secret, {EUR}) *dom* (Secret, {EUR})
  - Paul, with label (Top Secret, {NUC, EUR, ASI}), cannot read paper
    - (Top Secret, {NUC, EUR, ASI}) *dom* (Top Secret, {NUC, EUR})

Formal Model Definitions

- S subjects, O objects, P rights
  - Defined rights: r read, a write, w read/write, e empty
- M set of possible access control matrices
- C set of clearances/classifications, K set of categories, $L = C \times K$ set of security levels
- $F = \{ (f_s, f_o, f_c) \}$
  - $f_s(s)$ maximum security level of subject s
  - $f_c(s)$ current security level of subject s
  - $f_o(o)$ security level of object o

More Definitions

- Hierarchy functions $H: O \rightarrow P(O)$
- Requirements
  1. $o_i \neq o_j \Rightarrow h(o_i) \cap h(o_j) = \varnothing$
  2. There is no set $\{ o_1, \ldots, o_k \} \subseteq O$ such that, for $i = 1, \ldots, k$, $o_{i+1} \in h(o_i)$ and $o_{k+1} = o_1$.
- Example
  - Tree hierarchy; take $h(o)$ to be the set of children of o
  - No two objects have any common children (#1)
  - There are no loops in the tree (#2)

States and Requests

- V set of states
  - Each state is (b, m, f, h)
    - b is like m, but excludes rights not allowed by f
- R set of requests for access
- D set of outcomes
  - y allowed, n not allowed, i illegal, o error
- W set of actions of the system
  - $W \subseteq R \times D \times V \times V$

History

- $X = R^N$ set of sequences of requests
- $Y = D^N$ set of sequences of decisions
- $Z = V^N$ set of sequences of states
- Interpretation
  - At time $t \in N$, system is in state $z_{t-1} \in V$; request $x_t \in R$ causes system to make decision $y_t \in D$, transitioning the system into a (possibly new) state $z_t \in V$
- System representation: $\Sigma(R, D, W, z_0) \in X \times Y \times Z$
  - $(x, y, z) \in \Sigma(R, D, W, z_0)$ iff $(x_t, y_t, z_{t-1}, z_t) \in W$ for all t
  - (x, y, z) called an *appearance* of $\Sigma(R, D, W, z_0)$

Example

- $S = \{ s \}$, $O = \{ o \}$, $P = \{ \underline{r}, \underline{w} \}$
- $C = \{$ High, Low $\}$, $K = \{$ All $\}$
- For every $f \in F$, either $f_c(s) = ($ High, $\{$ All $\})$ or $f_c(s) = ($ Low, $\{$ All $\})$
- Initial State:
  - $b_1 = \{ (s, o, \underline{r}) \}$, $m_1 \in M$ gives s read access over o, and for $f_1 \in$ F, $f_{c,1}(s) =$ (High, {All}), $f_{o,1}(o) =$ (Low, {All})
  - Call this state $v_0 = (b_1, m_1, f_1, h_1) \in V$.

First Transition

- Now suppose in state v_0: $S = \{ s, s' \}$
- Suppose $f_{c,1}(s') =$ (Low, {All})
- $m_1 \in M$ gives s and s' read access over o
- As s' not written to o, $b_1 = \{ (s, o, \underline{r}) \}$
- $z_0 = v_0$; if s' requests r_1 to write to o:
  - System decides $d_1 = \underline{y}$
  - New state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{r}), (s', o, \underline{w}) \}$
  - Here, $x = (r_1)$, $y = (\underline{y})$, $z = (v_0, v_1)$

Second Transition

- Current state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{r}), (s', o, \underline{w}) \}$
  - $f_{c,1}(s) = (\text{High}, \{ \text{All} \}), f_{o,1}(o) = (\text{Low}, \{ \text{All} \})$
- s' requests r_2 to write to o:
  - System decides $d_2 = \underline{n}$ (as $f_{c,1}(s)$ *dom* $f_{o,1}(o)$)
  - New state $v_2 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{r}), (s', o, \underline{w}) \}$
  - So, $x = (r_1, r_2)$, $y = (\underline{y}, \underline{n})$, $z = (v_0, v_1, v_2)$, where $v_2 = v_1$