ECS 235B, Lecture 8

January 25, 2019

Example: Trusted Solaris

- Provides mandatory access controls
  - Security level represented by *sensitivity label*
  - Least upper bound of all sensitivity labels of a subject called *clearance*
  - Default labels ADMIN_HIGH (dominates any other label) and ADMIN_LOW (dominated by any other label)
- S has controlling user U_S
  - S_L sensitivity label of subject
  - *privileged*(S, P) true if S can override or bypass part of security policy P
  - *asserted* (S, P) true if S is doing so

Rules

C_L clearance of S, S_L sensitivity label of S, U_S controlling user of S, and O_L sensitivity label of O

1. If ¬*privileged*(S, "change S_L"), then no sequence of operations can change S_L to a value that it has not previously assumed
2. If ¬*privileged*(S, "change S_L"), then ¬*asserted*(S, "change S_L")
3. If ¬*privileged*(S, "change S_L"), then no value of S_L can be outside the clearance of U_S
4. For all subjects S, named objects O, if ¬*privileged*(S, "change O_L"), then no sequence of operations can change O_L to a value that it has not previously assumed

Rules (*con't*)

C_L clearance of S, S_L sensitivity label of S, U_S controlling user of S, and O_L sensitivity label of O

5. For all subjects *S,* named objects *O*, if ¬*privileged*(*S*, "override *O*'s mandatory read access control"), then read access to *O* is granted only if S_L *dom* O_L
   - Instantiation of simple security condition
6. For all subjects *S,* named objects *O*, if ¬*privileged*(*S*, "override *O*'s mandatory write access control"), then write access to *O* is granted only if O_L *dom* S_L and C_L *dom* O_L
   - Instantiation of *-property

Initial Assignment of Labels

- Each account is assigned a label range [clearance, minimum]
- On login, Trusted Solaris determines if the session is single-level
  - If clearance = minimum, single level and session gets that label
  - If not, multi-level; user asked to specify clearance for session
    - Must be in the label range
  - In multi-level session, can change to any label in the range of the session clearance to the minimum

Writing

- Allowed when subject, object labels are the same or file is in downgraded directory *D* with sensitivity label D_L and all the following hold:
  - S_L *dom* D_L
  - *S* has discretionary read, search access to *D*
  - O_L *dom* S_L and $O_L \neq S_L$
  - *S* has discretionary write access to *O*
  - C_L *dom* O_L
- Note: subject cannot read object

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*
  - System call stat(".", &buf) returns information about */tmp/d*
  - System call mldstat(".", &buf) returns information about*/tmp*

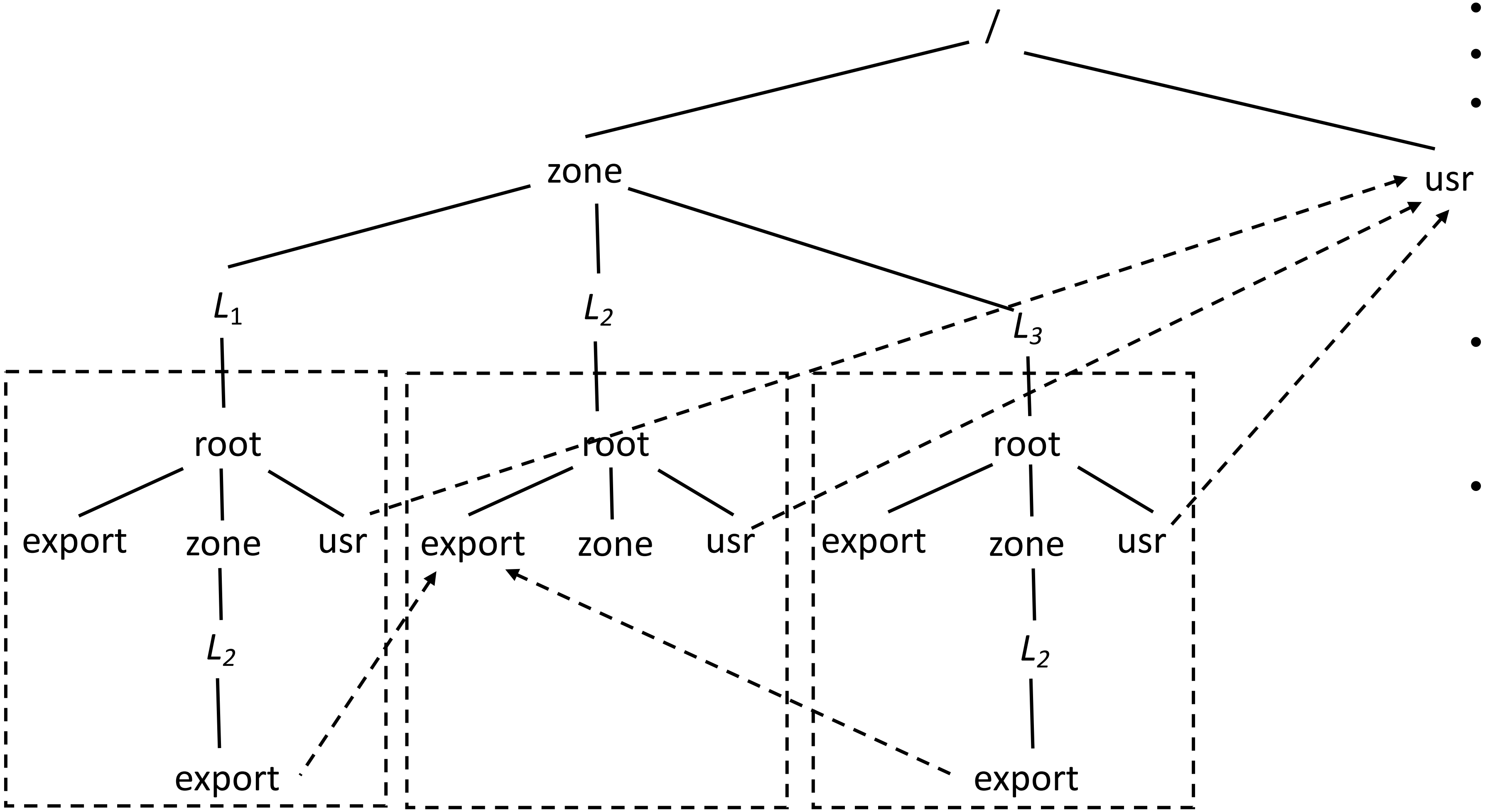
Labeled Zones

- Used in Trusted Solaris Extensions, various flavors of Linux
- *Zone*: virtual environment tied to a unique label
  - Each process can only access objects in its zone
- *Global zone* encompasses everything on system
  - Its label is ADMIN_HIGH
  - Only system administrators can access this zone
- Each zone has a unique root directory
  - All objects within the zone have that zone's label
  - Each zone has a unique label

More about Zones

- Can import (mount) file systems from other zones provided:
  - If importing *read-only*, importing zone's label must dominate imported zone's label
  - If importing *read-write*, importing zone's label must equal imported zone's label
    - So the zones are the same; import unnecessary
  - Labels checked at time of import
- Objects in imported file system retain their labels

Example



- L_1 *dom* L_2
- L_3 *dom* L_2
- Process in L_1 can read any file in the export directory of L_2 (assuming discretionary permissions allow it)
- L_1, L_3 disjoint
  - Do not share any files
- System directories imported from global zone, at ADMIN_LOW
  - So can only be read

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: $O \to P(O)$
- Requirements
  1. $o_i \neq o_j \Rightarrow h(o_i) \cap h(o_j) = \emptyset$
  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
  - <u>y</u> allowed, <u>n</u> not allowed, <u>i</u> illegal, <u>o</u> 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

- $S = \{ s \}$, $O = \{ o \}$, $P = \{ \underline{\mathrm{r}}, \underline{\mathrm{w}} \}$
- $C = \{$ High, Low $\}$, $K = \{$ All $\}$
- For every $f \in F$, either $f_c(s) = ($ High, $\{$ All $\})$ or $f_c(s) = ($ Low, $\{$ All $\})$
- Initial State:
  - $b_1 = \{ (s, o, \underline{\mathrm{r}}) \}$, $m_1 \in M$ 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 $f_{s,1}(s')$ = (Low, {All}), $m_1 \in M$ gives s read access over o and s' write access to o
- As s' not written to o, $b_1 = \{ (s, o, \underline{\text{r}}) \}$
- $z_0 = v_0$; if s' requests r_1 to write to o:
  - System decides $d_1 = \underline{\text{y}}$ (as m_1 gives it that right, and $f_{s,1}(s') = f_o(o)$
  - New state $v_1 = (b_2, m_1, f_1, h_1) \in V$
  - $b_2 = \{ (s, o, \underline{\text{r}}), (s', o, \underline{\text{w}}) \}$
  - Here, $x = (r_1)$, $y = (\underline{\text{y}})$, $z = (v_0, v_1)$

Second Transition

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

Basic Security Theorem

- Define action, secure formally
  - Using a bit of foreshadowing for “secure”
- Restate properties formally
  - Simple security condition
  - *-property
  - Discretionary security property
- State conditions for properties to hold
- State Basic Security Theorem

Action

- A request and decision that causes the system to move from one state to another
  - Final state may be the same as initial state
- $(r, d, v, v') \in R \times D \times V \times V$ is an *action* of $\Sigma(R, D, W, z_0)$ iff there is an $(x, y, z) \in \Sigma(R, D, W, z_0)$ and a $t \in N$ such that $(r, d, v, v') = (x_t, y_t, z_t, z_{t-1})$
  - Request r made when system in state v'; decision d moves system into (possibly the same) state v
  - Correspondence with (x_t, y_t, z_t, z_{t-1}) makes states, requests, part of a sequence