

# Lecture 5

## October 1, 2021

# 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  $\neg\text{privileged}(S, \text{"change } S_L\text{"})$ , then no sequence of operations can change  $S_L$  to a value that it has not previously assumed
2. If  $\neg\text{privileged}(S, \text{"change } S_L\text{"})$ , then  $\neg\text{privileged}(S, \text{"change } S_L\text{"})$
3. If  $\neg\text{privileged}(S, \text{"change } S_L\text{"})$ , then no value of  $S_L$  can be outside the clearance of  $U_S$
4. For all subjects  $S$ , named objects  $O$ , if  $\neg\text{privileged}(S, \text{"change } O_L\text{"})$ , 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  $\neg\text{privileged}(S, \text{"override } O\text{'s mandatory read access control"})$ , then read access to  $O$  is granted only if  $S_L \text{ dom } O_L$ 
  - Instantiation of simple security condition
6. For all subjects  $S$ , named objects  $O$ , if  $\neg\text{privileged}(S, \text{"override } O\text{'s mandatory write access control"})$ , then write access to  $O$  is granted only if  $O_L \text{ dom } S_L$  and  $C_L \text{ dom } O_L$ 
  - 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:
  - $S_L \text{ dom } D_L$
  - $S$  has discretionary read, search access to  $D$
  - $O_L \text{ dom } S_L$  and  $O_L \neq S_L$
  - $S$  has discretionary write access to  $O$
  - $C_L \text{ dom } O_L$
- 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  $MAC\_B \text{ dom } MAC\_A$
- 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  $lstat(".", \&buf)$  returns information about  $/tmp$



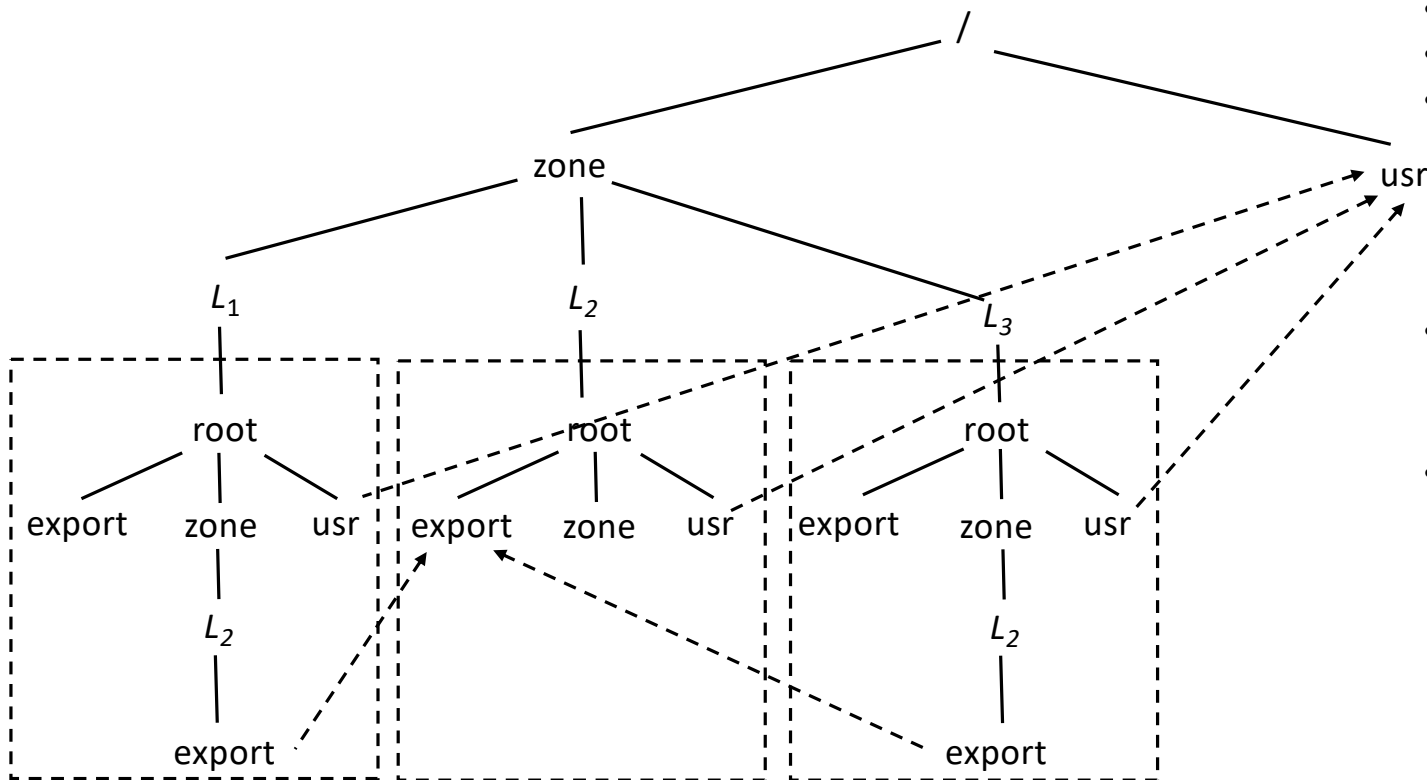
# Labeled Zones

- 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



- $L_1 \text{ dom } L_2$
- $L_3 \text{ dom } L_2$
- 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

# Requirements of Integrity Policies

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a non-production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

# Principles of Operation

- *Separation of duty*: if two or more steps are required to perform a critical function, at least two different people should perform the steps
- *Separation of function*: different entities should perform different functions
- *Auditing*: recording enough information to ensure the abilities to both recover and determine accountability

# Biba Integrity Model

Basis for all 3 models:

- Set of subjects  $S$ , objects  $O$ , integrity levels  $I$ , relation  $\leq \subseteq I \times I$  holding when second dominates first
- $min: I \times I \rightarrow I$  returns lesser of integrity levels
- $i: S \cup O \rightarrow I$  gives integrity level of entity
- $\underline{r}: S \times O$  means  $s \in S$  can read  $o \in O$
- $\underline{w}, \underline{x}$  defined similarly

# Intuition for Integrity Levels

- The higher the level, the more confidence
  - That a program will execute correctly
  - That data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important point: *integrity levels are **not** security levels*

# Information Transfer Path

- An *information transfer path* is a sequence of objects  $o_1, \dots, o_{n+1}$  and corresponding sequence of subjects  $s_1, \dots, s_n$  such that  $s_i \underline{r} o_i$  and  $s_i \underline{w} o_{i+1}$  for all  $i, 1 \leq i \leq n$ .
- Idea: information can flow from  $o_1$  to  $o_{n+1}$  along this path by successive reads and writes



# Strict Integrity Policy

- Dual of Bell-LaPadula model
  1.  $s \in S$  can read  $o \in O$  iff  $i(s) \leq i(o)$
  2.  $s \in S$  can write to  $o \in O$  iff  $i(o) \leq i(s)$
  3.  $s_1 \in S$  can execute  $s_2 \in S$  iff  $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model
- If there is an information transfer path from  $o_1 \in O$  to  $o_{n+1} \in O$ , the low-water-mark policy requires  $i(o_{n+1}) \leq i(o_1)$  for all  $n > 1$ .
- Term “Biba Model” refers to this

# LOCUS and Biba

- Goal: prevent untrusted software from altering data or other software
- Approach: make levels of trust explicit
  - *credibility rating* based on estimate of software's trustworthiness (0 untrusted,  $n$  highly trusted)
  - *trusted file systems* contain software with a single credibility level
  - Process has *risk level* or highest credibility level at which process can execute
  - Must use *run-untrusted* command to run software at lower credibility level