ECS 235B Module 34
Policy Composition
Composition of Policies

• Two organizations have two security policies
• They merge
  • How do they combine security policies to create one security policy?
  • Can they create a coherent, consistent security policy?
The Problem

• Single system with 2 users
  • Each has own virtual machine
  • Holly at system high, Lara at system low so they cannot communicate directly

• CPU shared between VMs based on load
  • Forms a *covert channel* through which Holly, Lara can communicate
Example Protocol

• Holly, Lara agree:
  • Begin at noon
  • Lara will sample CPU utilization every minute
  • To send 1 bit, Holly runs program
    • Raises CPU utilization to over 60%
  • To send 0 bit, Holly does not run program
    • CPU utilization will be under 40%

• Not “writing” in traditional sense
  • But information flows from Holly to Lara
Policy vs. Mechanism

• Can be hard to separate these

• In the abstract: CPU forms channel along which information can be transmitted
  • Violates *-property
  • Not “writing” in traditional sense

• Conclusion:
  • Bell-LaPadula model does not give sufficient conditions to prevent communication, or
  • System is improperly abstracted; need a better definition of “writing”
Composition of Bell-LaPadula

• Why?
  • Some standards require secure components to be connected to form secure (distributed, networked) system

• Question
  • Under what conditions is this secure?

• Assumptions
  • Implementation of systems precise with respect to each system’s security policy
Issues

• Compose the lattices
• What is relationship among labels?
  • If the same, trivial
  • If different, new lattice must reflect the relationships among the levels
Example
Analysis

• Assume $S < \text{HIGH} < TS$
• Assume SOUTH, EAST, WEST different
• Resulting lattice has:
  • 4 clearances ($\text{LOW} < S < \text{HIGH} < \text{TS}$)
  • 3 categories (SOUTH, EAST, WEST)
Same Policies

• If we can change policies that components must meet, composition is trivial (as above)

• If we cannot, we must show composition meets the same policy as that of components; this can be very hard
Different Policies

• What does “secure” now mean?
• Which policy (components) dominates?
• Possible principles:
  • Any access allowed by policy of a component must be allowed by composition of components (autonomy)
  • Any access forbidden by policy of a component must be forbidden by composition of components (security)
Implications

• Composite system satisfies security policy of components as components’ policies take precedence

• If something neither allowed nor forbidden by principles, then:
  • Allow it (Gong & Qian)
  • Disallow it (Fail-Safe Defaults)
Example

• System X: Bob can’t access Alice’s files
• System Y: Eve, Lilith can access each other’s files
• Composition policy:
  • Bob can access Eve’s files
  • Lilith can access Alice’s files
• Question: can Bob access Lilith’s files?
Solution (Gong & Qian)

• Notation:
  • \((a, b)\): \(a\) can read \(b\)’s files
  • \(AS(x)\): access set of system \(x\)

• Set-up:
  • \(AS(X) = \emptyset\)
  • \(AS(Y) = \{ (Eve, Lilith), (Lilith, Eve) \}\)
  • \(AS(X \cup Y) = \{ (Bob, Eve), (Lilith, Alice), (Eve, Lilith), (Lilith, Eve) \}\)
Solution (Gong & Qian)

• Compute transitive closure of AS(X∪Y):
  • \( AS(X∪Y)^+ = \{ (Bob, Eve), (Bob, Lilith), (Bob, Alice), (Eve, Lilith), (Eve, Alice), (Lilith, Eve), (Lilith, Alice) \} \)

• Delete accesses conflicting with policies of components:
  • Delete (Bob, Alice)

• (Bob, Lilith) in set, so Bob can access Lilith’s files
Idea

• Composition of policies allows accesses not mentioned by original policies
• Generate all possible allowed accesses
  • Computation of transitive closure
• Eliminate forbidden accesses
  • Removal of accesses disallowed by individual access policies
• Everything else is allowed
• Note: determining if access allowed is of polynomial complexity