A Graph-based Approach to Specifying Security Constraint Policies

James Hoagland

Karl Levitt

Raju Pandey

Computer Security Research Laboratory

Department of Computer Science

University of California, Davis

{hoagland,levitt,pandey}@cs.ucdavis.edu

Security Policies

Security policies might consist of various elements, i.e.:

- ❏ when the policy applies
- ❏ what actions to take when it applies (i.e., provide security mechanisms)
- ❏ a constraint on the state the system must be in
- ❏ what to do iff this constraint is violated

Here we focus on the first and third: security constaints on a system.

Approach

Goals of work:

- an easy way to *formally* specify security policies
- have the method be application-independent

Our approach to specifying security policies:

- specify policies in a formal language
- policies consist of a set of constraints
- each constraint is represented by a graph
- constraints get checked against the system and violations reported
- the constraint graphs depict
  - when the policies apply (the *antecedent*)
  - what the requirement then is (the *consequent*)

System Model

Model the system to apply the policy to as:

- objects with attributes and values
- methods being invoked between objects

Graph-based Constraint Language

Language has nodes and edges:

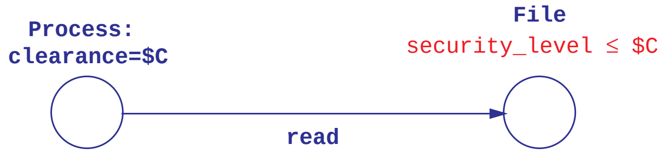
- ❑ nodes are a pattern for objects
- ❑ edges are a pattern for method invocations
  - source node is the invoking object
  - destination node is the invoked object

Nodes and edges have annotations:

- ❑ antecedent and consequent boolean expressions
- ❑ these predicates further restrict what objects and method invocations can match the constraint
- ❑ predicates can refer to:
  - object attribute values (nodes) or method parameter values (edges)
  - variables (bound like in Prolog, on first use)

Example Policy Specification Using Graphs

Example: a process with a certain clearance level can only read a file with lower or equal security level



Note:

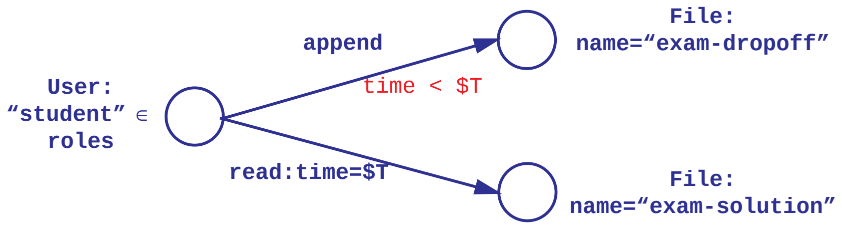
- ❑ **blue parts** are the antecedent or trigger (when the policy applies)
- ❑ red parts are the consequent or requirement (what must then be the case)

Exam Scenario Constraint Graph

An online exam is to be given for a class at a university. Part of the design is:

- ❏ completed exams are to be dropped off in a file
- ❏ solutions are to be available electronically to students after they turn in their exam, but not before

Policy: If a student appends to the exam dropoff file and reads the exam solution file, then the time of the append must be earlier than the time of the read.



Key: **antecedent is blue**; consequent is red

Future Work

Formally develop constraint language

- ❏ define system model formally
- ❏ fully define semantics of the language
- ❏ characterize the language's ability to express policies

Policy violation detection

- ❏ design and implement policy enforcement mechanism for Java

Composition of policies

- ❏ investigate different ways to compose policies
  - peer and prioritized policiesXC