Collaborative Research: Access Control and Downgrading for Information Flow Assurance (CNS-0627748 and CNS-0627338, 9/2006—8/2009)

Anindya Banerjee (Kansas State University) and David A. Naumann (Stevens Institute of Technology)

http://www.cis.ksu.edu/~ab and http://www.cs.stevens.edu/~naumann

Project Goals

- Express confidentiality and integrity (i.e., information flow) requirements as such with clear meaning for requirements analysts and implementers
- Account for access controls and other means to satisfy information flow requirements clearly in designs
- Check conformance of designs and implementations with information flow policies;
  - -- *Verify TCB wrt. policies that account for interaction with less trustworthy components*;
  - -- *Check object code before installation*.
- Develop validation tools for checking compliance based on well-understood theory and algorithms.
- Perform case studies on several applications, e.g., Mental Poker, CodeBlue, conference management systems,etc.

Problems Addressed

■What are the desirable information flow policies?

Provide useful semantics for information flow policy, in particular policies that allow qualified reclassification (downgrading/endorsement) after certain events by authorized parties; *Example: audit log may be written only by appending and with a valid timestamp.* Want policies amenable to automated checking, to have precise meaning, and to have intuitive reading.

■How should policies be specified?

Extend lattice-based notions of multi-level security to encompass access control disciplines like role-based trust management.

Extract information flow policies from declarative specifications of (a) extant access control policies; (b) policies catering for multiparty negotiation, *where the underlying lattice structure is not always apparent.*

Make policies state/event-dependent; required, for example, in role-based systems where principals in a role change over time.

■What techniques are effective for achieving high assurance?

Use formal methods because they (a) can be automated; (b) guarantee through correctness that verification implies assurance; (c) can be engineered to be scalable; (d) are exhaustive and can analyze all executions on all input. Extant approaches for checking information flow mostly use security type systems.

Technical Approach and Impact

Approach. Combine Hoare style pre-/post-condition specifications using *independence predicates* with conventional security typing. Use verification on small program fragments that do declassification. Develop corresponding program logic for fragment of Java with goal of automatic verification.

Impact. (a) Effective/intuitive specification of a broad class of information flow policies involving downgrading and sophisticated access policies. (b) Effective conformance checking for designs and implementations.

Collaboration with IBM Research, Microsoft Research, Mobius Project, INRIA, Rockwell Collins.

Technical description

Typically, correctness of information flow for lattice-based policies (L <= H) specified using *noninterference* (NI)*: For any two program runs, L indistinguishable initial states yield resulting states that are also L indistinguishable.*

Generalize NI to observable inputs and outputs. Write *independence predicate* x# to denote that variable x is indistinguishable for any *two states* s1, s2. Then NI can be written as *flowspec* {x1#,…,xn#} P {y1#,…,ym#} where the xi's are observable inputs and yi's are observable outputs.

- **Allows writing a *relational Hoare logic extended to two runs of a program* with proof rules**.
- **Can further generalize to, e.g., (x mod 3)# and use usual Hoare style assertions as well**.
- **Directly describes NI and even intransitive NI**; **allows specifying who/when/what/where dimensions of declassification.**

Benefit: Express baseline policies by labeling component interfaces with security types. Augment baseline policy by flowspecs expressing reclassification for specific code fragments. Incorporate ordinary state predicates that capture conditions under which reclassification is allowed.

Example: Auction with two phases, getBids and announceWinner. Principals Alice and Bob.
Phase 1: Alice is not influenced by Bob's bid: {bidB#} getBids {bidB#}
Phase 2: Only value of highest bid but not identity of bidder is revealed:

{max(bidA, bidB)# /\ (bidA >= bidB)# } announceWinner {result#}

National Science Foundation
WHERE DISCOVERIES BEGIN

NSF Cyber Trust Principal Investigators Meeting
January 28-30, 2007
Atlanta, Georgia