David Molnar
CS252 Project Proposal

Verifying Isolation-Related Security Properties

A *one-way channel* between two components A and B is a channel that allows A to talk to B, but B cannot talk to A. Such a channel provides *isolation* between the components - the ability of B to affect A is limited. A classic example of such a channel is the NRL Pump [2]. A *resettable component* is a component that can have its state wiped and reset to a "known good" state - this provides temporal isolation between different resettings of the same component. I will apply techniques from formal verification to reason about these and other isolation primitives. There are two types of verification to explore: first, verifying the design of a primitive, and second, verifying security properties of whole systems that use isolation primitives.

Example: An Airport e-Mail Kiosk

Consider an airport kiosk machine where people can check e-mail. The threat here is that someone (call her Alice) may install a keylogger and capture the next user's (call him Bob) passwords. Bob needs some assurance that the kiosk he uses isn't running such a keylogger.

The kiosk problem is similar to, but easier than, the e-voting problem. It is similar, because a voting machine must not leak information about Bob's vote to Alice, just as the kiosk must not leak Bob's keystrokes to Alice. The kiosk problem is easier, however, because it does not have the high availability and privacy requirements of e-voting.

We assume that both Alice and Bob can carry USB keys which have been formatted with an OS that boots into a RAM disk. We now consider a kiosk with the following components: A) A USB port, B) A CPU, display, and so on, modified to have no stable storage, and C) A "resetting component" that, on input from the user, resets component B) and clears its state. For example, C might be a power switch.

We arrange the system with a one-way channel from A to B, and a one-way channel from C to B. Given this organization, we can make some security claims about the system, even if we trust only components A and C, but do not trust component B.

More specifically, if Bob uses C to reset B before logging in, then loads a new OS image from his own USB key via A, then Bob can be sure that B doesn't have any malicious code loaded by Alice. Even if B has some kind of malicious code designed into it that can survive a reset, Bob can be sure his USB key is safe. The resetting property and the one-way channel allows us to limit trust in the system to two small and unchanging components, plus trust in the statelessness of component B. Can we formalize and automatically check security properties of this and similar designs?

Timeline and Goals

- March 15-April 1. Gain familiarity with Murϕ tool by replicating Murϕ covert channel analysis of the NRL Pump [1]. Develop library of informally justified examples of isolation-related security property, similar to airport kiosk example. Explore other verification tools as needed.
- April 1-May 5. Identify a small number of security properties, including at least one whole-system property, that are tractable for verification. Formalize properties and carry out verification of a design.

References

[1] Foo. An automatic analysis of the NRL pump. In *Foo*, 2003.

[2] D. Goldschlag. Several secure store-and-forward devices. In *ACM CCS*, 1996.