Breakout group: How can we design systems with known security properties?

NSF/IARPA/NSA Workshop on the Science of Security

November 17-18, 2008

Systems with Known Security Properties

- Two easy edge case approaches:
  - Good security properties (at least safety properties), utility = ∅.
  - Good utility properties, security properties = ∅.
- Challenge is going beyond these edges.
  - Good security properties for a specified utility
  - Known security properties for a specified utility
  - Small steps may be helpful.
- Is there even a clear distinction between security properties and utility properties?

Security by Design

- Can security by design be accomplished without a science of security?
  - Maybe even an incomplete science can provide a useful scaffolding.
- Designing systems that meet specified goals vs. being able to determine the properties of a given system.
  - Either way, only interesting if security properties involved are relevant.
- Composability seems critical to working at an understandable level of complexity.
- Models and assumptions: sometimes a "model" is an assumption that is known to be false.
  - Can be useful anyway.
- Example: melding of cryptographic reductions approach and formal methods ideal cryptography approach.

Ideas from Discussion

- Partly a function of what your security policy is (military systems with well specified security policies vs. user applications with unclear policies)
- Question: Has there ever been a system that was proven secure but was later broken? (Discussion: we don't even have "systems" that satisfy the first part of the question, though for smaller components the answer is yes.) In fact, the things that we can currently prove security statements about are much much smaller than deployed systems. Even as components, they are simplified beyond what is actually implemented.
- In security, call for science of security comes from too many examples of attacks not respecting model, model fixed, repeat. Arms race of fixes/proofs and attacks.

Discussion, cont'd.

- Some security properties may be more amenable to proving than others.
- Could we come up with an exemplary proof (e.g. work of Don Good on stack correctness) of security for a particular example? Would it make sense? Lead to anything of interest? Have other benefits?
- What's the best way to deploy limited resources? Diverse set of approaches, plus extra effort where threat is high. (Buffer overflows, web scripts, anything directly exposed to outside world that doesn't protect itself.)
- How can you guarantee system-level properties out of component-level arguments? Need to adjust notion of architecture to support decomposition of security properties. Only realistic in domains where security is the over-riding concern, which is not often.

Discussion, cont'd.

- What about building secure systems out of bad/unknown components? Can we borrow ideas from fault-tolerance/safety? Difficulties include lack of diversity, adversarial setting. But these could be points to tackle in order to be able to use fault-tolerance ideas. Also a good place for biological approaches, since life is also a large complex system built out of failure-prone components. Has this community really looked at all the relevant methodologies from other fields? What is security "safety factor"?
- E.g: building bridges is a complex task involving cost, uncertainty, risk, load, etc. Just because this is hard, doesn't mean it can't be done.
- Do we have (can we create) a core security methodology from which we can build?
- Need more interaction and balance between scientific approach and engineering?

Discussion, cont'd.

- Is this inherent, or do we just need the great insight that will change everything?
- Are proofs really necessary? Are there ways other than proofs to know security properties? Example: type systems let you build-in constraints rather than proving them (i.e., the system builds the proof).
- Why is this really so hard? We have formally verified compilers. They are complicated too (they have to work for every possible program). Answers: concurrency, others?
- Approach: design systems simply so that certain properties are respected. (E.g., hope for virtual machine monitors or other small TCB's with security-relevant tasks to be more secure than general-purpose systems.)
- Design principles.

Discussion, cont'd.

- Analogy to building cathedrals. Good components. Still, early cathedrals fell down. We need to understand out components better before we start putting them together.
- Existing point solutions are not bad, they are just evidence that we are early in the process. We are in the first phase of learning about things. Is there now a body (or bodies) of knowledge that we can leverage to get to the next step?
- Concentrate of parts that expose interfaces? Vulnerabilities can only be exploited if they can be accessed.
- Can we find scientific boundary conditions that can be proven? Upper (lower) bounds?
- Often security is for cross-domain solutions. Purpose (need to send data to enable) vs. anti-purpose (need to not send data to enable). Adding interaction makes things complicated.

Discussion, cont’d.

- In some areas we have rigor, but culture of community does not always support it. Research community wants more publications, standards bodies don’t attach value to rigor unless it does not detract from other aspects.
- Harder to say that a secure system is secure than to demonstrate that an insecure one is insecure. Can we reduce to physical security of some components? Environment must be included.
- Is it help to focus on what is securable (i.e., what might potentially be able to be secured) even if we can’t yet say what is secure?
- As an example, is intrusion detection an area where we should take a scientific approach to carrying it out, or should we try to prove lower bounds, or both?