

Formal Methods for Web 2.0 Security Protocols - Position Paper

Andrew Cirillo

Radha Jagadeesan

Corin Pitcher

James Riely

DePaul University

E-mail: {acirillo,rjagadeesan,cpitcher,jriely}@cs.depaul.edu

Challenges We argue that current formal approaches to cryptographic protocols are at an inappropriate level of abstraction when used to address Web 2.0 Security and Privacy issues.

1. They lack an explicit and primitive notion of identity. The foundations — usually based on an (applied) pi-calculus — are organized around channel names and communication on named channels. Identities are established indirectly: e.g., an identity is prescribed by setting up protocols to ensure that the only sender (or receiver) on a channel name is the entity under consideration.
2. The network model is too detailed. The traditional model of the network as an opponent in cryptographic protocols enables the opponent to forge, alter and remove messages. In contrast, the basic protocols underlying, say, identity management frameworks assume integrity of messages. This greater abstraction permits the users and architects of Web 2.0 components to focus on semantic issues, such as trust and privacy, rather than on issues related to active attacks on the underlying cryptographic protocols that facilitate the assumption of integrity.
3. Programming abstractions do not address content composition. Composition of content involves subtle combinations of authentication, naming, delegation, and trust, e.g., `del.icio.us` stating that `janedoe` has tagged a “Podcasting in Education” URL as `web2.0`.

Our Approach In recent work [1], we describe a model of execution where every piece of code is associated with a principal — we could say that the code is located at the principal. To address item 2, message-passing interaction between principals is unforgeable. To address item 3, principals can express the intent of data in a message, e.g., a claim that a `web2.0` tag for a URL originates with `janedoe`.

Each principal has its own local notion of trust. The trust beliefs at each principal can be modeled by a security lattice of principals — a principal is (locally) more trustworthy if she is lower in the security lattice. In concordance with the distributed context, we do not demand global consistency of local security lattices. The code located at a principal executes in the context of the trust lattice of the principal, using the local trust lattice to answer questions about the local ordering of principals in the trust lattice. We envision that the local security lattice evolves dynamically, with potential addition/removal of new principals and their ordering relationships during execution of the program.

In this context, one can revisit the traditional methods of static and dynamic analysis to enforce and infer security properties.

Events Calendar We highlight key aspects of our approach [1] in the context of a shared calendar service of events populated by user-supplied data, which is tied to users via some flexible authentication scheme. Basic operations of such a service include: users querying the calendar service for relevant events; and users posting details of events to the calendar service.

We consider two security properties. First, a query may be for events from users authenticated by a proper subset of the identity providers (e.g., America Online only) that the calendar service accepts (e.g., any OpenId provider). The security property that users examining query results expect is that events were indeed posted by a user authenticated by America Online. Second, a shared calendar service may have a dynamic access control policy for posting that is determined by users in a moderator role. The security property that users examining query results expect is that events were indeed posted by a user acceptable to moderators.

A combination of correspondence assertions and logical deduction can be used to formalize these security properties. However, logical inconsistencies, also found in other contexts such as logical spreadsheets [2], arise from inconsistent trust relationships. For example, two users may disagree on whether Alice posted an event if they disagree on whether Alice’s identity provider is trustworthy. To address this problem we adopt a subset of authorization logic [3] that extends Datalog, for reasons of tractability, as a means to control inconsistent viewpoints at different levels of trust. The resulting logic is integrated with a type-and-effect system for correspondence assertions and with model checking to address formal verification of such security properties.

References

- [1] A. Cirillo, R. Jagadeesan, C. Pitcher, and J. Riely. Do As I SaY! Programmatic access control with explicit identities. In *20th IEEE Computer Security Foundations Symposium*, 2007.
- [2] M. Kassoﬀ, L.-M. Zen, A. Garg, and M. Genesereth. PrediCalc: A logical spreadsheet management system. In *31st International Conference on Very Large Databases (VLDB)*, pages 1247–1250, 2005.
- [3] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: theory and practice. *ACM Trans. Comput. Syst.*, 10(4):265–310, 1992.