Mapping the Security Landscape: A Role for Language Techniques

Abstract of Invited Lecture

Fred B. Schneider*

Department of Computer Science
Cornell University
Ithaca, New York 14558
U.S.A
fbs@cs.cornell.edu

Over the last decade, programming language techniques have been applied in non-obvious ways to building secure systems. This talk will not only survey that work in *language based security* but show that the theoretical underpinnings of programming languages are a good place to start for developing a much needed foundation for software system security.

Research developments in language design, compilers, program analysis, type checking, and program rewriting has brought increased assurance in what a software system does and does not do. This was needed, welcome, but not surprising. What is surprising are the new approaches to engineering secure systems that have emerged from language-based security research. Inline reference monitors, for example, enable enforcement of a broad class of authorization policies specified separately from the components they concern; and proof carrying code provides a way to relocate trust from one system component to another, providing designers with flexibility in setting the trusted computing base.

Our sophistication in the engineering of secure systems has improved without the benefit of a rigorous mathematical foundation. This is probably a matter of luck and certainly a matter for concern. Too much of security engineering is based on anecdotes, driven by past attacks, and directed by a few familiar kinds of security policies. We have still to identify abstract classes of enforcement mechanisms, attacks, or policies, much less understand connections between them. In short, we have made little progress in mapping the security landscape.

Yet some features of the security landscape do start to become apparent if the abstractions and metaphors of programming language semantics are applied to software system security. And there is reason to believe the approach will yield more fruit, too. We will discuss what is known, what we might strive to know, and the role that programming language semantics can play.

Acknowledgment. I am grateful to my collaborators Michael Clarkson, Kevin Hamlen, Andrew Myers, Greg Morrisett, and Riccardo Pucella.

* Supported in part by AFOSR grant F9550-06-0019, National Science Foundation Grants 0430161 and CCF-0424422 (TRUST), and a gift from Microsoft Corporation.

L. Caires and V.T. Vasconcelos (Eds.): CONCUR 2007, LNCS 4703, p. 1, 2007.