Computer Security
An introduction, with emphasis on language-based methods

Course at the 1st North American Summer School in
Logic, Language, and Information

Martín Abadi
Department of Computer Science
University of California at Santa Cruz
June 24–28, 2002

These lectures are an introduction to computer security and to the role of formal models, logics, and programming calculi in that area. Because of the breadth of the subject, the lectures are deliberately partial.

After a general overview, the lectures stress the application of techniques originally developed in the study of programming languages. In particular, these techniques play a significant role in fine-grained program protection and in the analysis of protocols for secure communication.

Security is one of the stated objectives of some language run-time systems. For example, typical Java virtual machines permit the execution of code that is not fully trusted. They aim to enforce security by a combination of static and dynamic checks. These include bytecode verification (a technique for checking programs before execution) and stack inspection (a technique for associating permissions with stack frames). It should not be surprising that language-based techniques are relevant in this context. Specifically, bytecode verification is a form of typing; stack inspection may be seen as an approximation to run-time dependency analysis. Concepts from programming-language theory, such as type soundness and full abstraction, sometimes play a role in guiding or assessing this machinery.

Security is also the focus of much work on network protocols. The lectures cover some example protocols (fragments of SSL, SSH, etc.) and informal principles. (The lectures do not however include a detailed survey of cryptographic algorithms; we tend to treat those algorithms as black boxes.) Security protocols are notoriously prone to design and implementations er-
rors. Therefore, various methods have been proposed for developing and analyzing these protocols. Some of the methods are informal; others rely on formal calculi—in particular, logics and process algebras. These methods have led to increased confidence in some existing protocols and have occasionally been helpful in the invention of new ones. More frequently, these methods have contributed to the identification of dangerous blunders. The lectures discuss some informal guidelines for protocol design and three particular formal methods (“BAN logic”, the applied calculus, and automated checking with logic-programming tools).

The slides for the lectures will be available, at least in draft form, sometime before the lectures, on the web at:


References

The lectures are partly based on papers available at:

http://www.cs.ucsc.edu/~abadi/allpapers.html

Particularly relevant are:

- Prudent Engineering Practice for Cryptographic Protocols
- A Logic of Authentication
- Mobile Values, New Names, and Secure Communication
- Analyzing Security Protocols with Secrecy Types and Logic Programs
- A Type System for Java Bytecode Subroutines

The following two books may provide useful background and complementary material:

  http://www.cacr.math.uwaterloo.ca/hac/

Additional references will be provided during NASSLI.

Prerequisites

The main prerequisites are some familiarity with computers and programming, and ease with formal notation.