Retrofitting Legacy Code for Security
(Invited Talk)

Somesh Jha

University of Wisconsin, Madison, USA

Research in computer security has historically advocated Design for Security, the principle that security must be proactively integrated into the design of a system. While examples exist in the research literature of systems that have been designed for security, there are few examples of such systems deployed in the real world. Economic and practical considerations force developers to abandon security and focus instead on functionality and performance, which are more tangible than security. As a result, large bodies of legacy code often have inadequate security mechanisms. Security mechanisms are added to legacy code on-demand using ad hoc and manual techniques, and the resulting systems are often insecure.

This talk advocates the need for techniques to retrofit systems with security mechanisms. In particular, it focuses on the problem of retrofitting legacy code with mechanisms for authorization policy enforcement. It introduces a new formalism, called fingerprints, to represent security-sensitive operations. Fingerprints are code templates that represent accesses to security-critical resources, and denote key steps needed to perform operations on these resources. This talk develops both fingerprint mining and fingerprint matching algorithms.

Fingerprint mining algorithms discover fingerprints of security-sensitive operations by analyzing source code. This talk presents two novel algorithms that use dynamic program analysis and static program analysis, respectively, to mine fingerprints. The fingerprints so mined are used by the fingerprint matching algorithm to statically locate security-sensitive operations. Program transformation is then employed to statically modify source code by adding authorization policy lookups at each location that performs a security-sensitive operation.

These techniques have been applied to three real-world systems. These case studies demonstrate that techniques based upon program analysis and transformation offer a principled and automated alternative to the ad hoc and manual techniques that are currently used to retrofit legacy software with security mechanisms. Time permitting, we will talk about other problems in the context of retrofitting legacy code for security. I will also indicate where ideas from model-checking have been used in this work.