Retrofitting Software for Defense-in-Depth

PIs: Trent Jaeger (PSU), Vinod Ganapathy (Rutgers), Chritian Skalka (UVM), Gang Tan (PSU)

https://www.cse.psu.edu/~trj1/retrofit4security.html

Goal: Retrofit legacy software with a combination of security mechanisms optimally via automated methods

Programmers manually retrofit their programs to add security controls for:

Privilege Separation
Authorization
Auditing

Can we automate retrofitting across all these security controls optimally?

Retrofitting for privilege separation should enable balance of security and performance

Retrofitting for authorization should apply hooks to enforce expected policies

Retrofitting for auditing should enable logging to answer expected retrospective queries

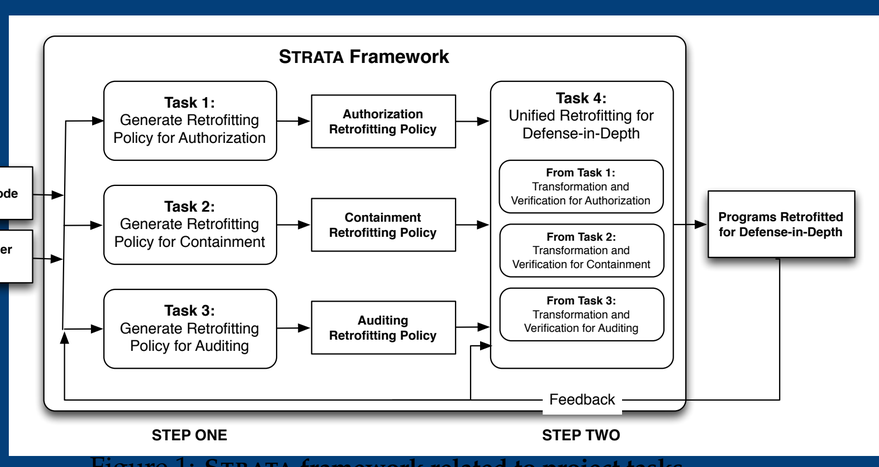

Figure 1: STRATA framework related to project tasks

The STRATA Framework aims to aid programmers in designing *retrofitting policies* that can then be *applied in an unified manner* to produce *optimized and validated security code*.

Approach

- Find security-sensitive operations: Each security control aims to mediate access to security-sensitive operations.
- Aid programmers in defining retrofitting policies: Security goals must be related to security-sensitive operations.
- Place controls for security goals: Given a retrofitting policy, transform the program to satisfy that policy while minimizing cost.
- Verify correct transformations: Leverage formal methods to assure correct transforms from our retrofitting framework.

Privilege Separation

Problem: Many modern frameworks offer support for least-privilege execution, but transforming legacy software to conform to these frameworks is tedious and error prone.

Approach: Developing an automated toolchain to refactor software for least-privilege frameworks. Implemented for the Mozilla JetPack framework.

Auditing

Problem: A lack of formal foundations supporting correctness conditions for specifying and retrofitting to enforce in-depth auditing policies.

Approach: Developing new formal semantics for auditing to support logical specifications of in-depth policies and soundness/completeness properties for retrofitting auditing instrumentation.

Authorization

Problem: Programmers have manually modified several programs to add authorization, but this task is time-consuming and error prone.

Approach: Use "choices" programs make using input to identify operations and leverage constraints on access policies to generate minimal placements.

Validation

Problem: Retrofitting transformations often implemented as compiler passes. Guarantees offered by inserted instrumentation may be undone by compiler optimizations.

Approach: Harnessing SMT solvers to prove equivalence of transformations across optimizations.

Interested in meeting the PIs? Attach post-it note below!

National Science Foundation
WHERE DISCOVERIES BEGIN

NSF Secure and Trustworthy Cyberspace Inaugural Principal Investigator Meeting
Nov. 27 -29th 2012
National Harbor, MD