Integration of Formal Methods and Testing for Model-Based Security Engineering

Achim D. Brucker

achim@brucker.ch http://www.brucker.ch/

NII Shonan Meeting Seminar 048

Integration of Formal Method and Testing for Model-Based Systems Engineering

December 1, 2014

Integration of Formal Methods and Testing for Model-Based Security Engineering

Abstract

We present a brief overview of various security testing works that range from applying off-the-shell tools (both dynamic tools as well as static program analysis) to theorem-prover based testing for ensuring the compliance of systems to high-level security policies.
Moreover, we report on the process of selecting the most appropriate (security) testing tools during product development derive open research questions based on our experience in developing, introducing, and applying (security) testing tools at SAP SE.

About ...

SAP SE

- Leader in Business Software
  - Cloud
  - Mobile
  - On premise
- Different technologies and platforms, e.g.,
  - In-memory database and application server (HANA)
  - Netweaver for ABAP and Java
- More than 25 industries
- 63% of the world's transaction revenue touches an SAP system
- More than 67 000 employees worldwide
- Headquartered in Walldorf, Germany

Myself

- *Senior Researcher* and *Program Lead* "Security Enablement - Strategy and Though Leadership"
- Foundational
  - Formal security models
  - Theorem prover-based testing
  - Testing and static analysis
- Applied:
  - Security Testing Strategy (more than 25 000 developers)
  - Evaluation/selection of security test tools (static, dynamic, and combined)
- Requirement-based modelling and verification

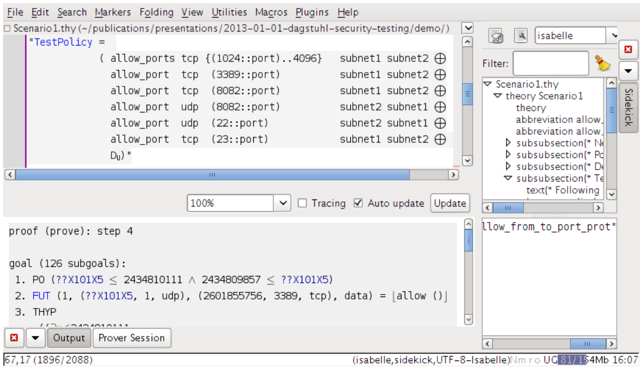
Model-based Security Testing: An Example

Overall idea:

1. Modeling security policies in HOL
   - Unified Policy Framework (UPF)
2. Optimizing models for testability
   - Verified policy transformations
3. Generating test cases
   - Partitioning using symbolic methods
4. Generating test data
   - Instantiating test cases
5. Validating test results
   - Execute concrete test data on system

Implementation:

- Add-on for HOL-TestGen
- Available as Free Software



```
"TestPolicy =
      ( allow_ports tcp {(1024::port)..4096}   subnet1 subnet2 ⊕
        allow_port  tcp  (3389::port)          subnet1 subnet2 ⊕
        allow_port  tcp  (8082::port)          subnet1 subnet2 ⊕
        allow_port  udp  (8082::port)          subnet2 subnet1 ⊕
        allow_port  udp  (22::port)            subnet2 subnet1 ⊕
        allow_port  tcp  (23::port)            subnet1 subnet2 ⊕
        D_U)"
```

```
proof (prove): step 4

goal (126 subgoals):
 1. PO (??X101X5 ≤ 2434810111 ∧ 2434809857 ≤ ??X101X5)
 2. FUT (1, (??X101X5, 1, udp), (2601855756, 3389, tcp), data) = ⌊allow ()⌋
 3. THYP
```

`http://www.brucker.ch/projects/hol-testgen/`

Technology Vector: Scope, Soundness, Coverage, Bug Finding Rate

“ Security testing methods should **be sound**. Really?

- **Observations:**
  - **Proving soundness** seems to be a prerequisite for getting an academic paper accepted.
  - Most “real-world” tools **are not sound** (the underlying theory might).
  - **Even worse:** most tools will not report anything, on our frameworks
- What I need in practice to provide the best “blend” to my developers:
  - A clear specification what it “in-scope”
  - A clear specification what it “out-scope”
  - Test cases that validate the expected behavior (e.g., similar to qualification kits for DO178C)
- **Claim:** We need more work on
  - clear specify the scope (soundness boundaries)
  - test sets for comparing security testing tools
  - extension/adaption points for testing tools

Technology Vector: Scope, Soundness, Coverage, Bug Finding Rate

> Security testing methods should **provide 100% coverage**. Really?

- **Observations:**
  - **Showing 100% coverage** seems to be a prerequisite for getting an academic paper accepted.
  - Most "real-world" tools **do not provide 100% coverage** (the underlying theory might).
  - **Even worse:** most tools will not report anything, on our frameworks
- What I need in practice to provide the best "blend" to my developers:
  - A clear specification what it "in-scope"
  - A clear specification what it "out-scope"
  - Test cases that validate the expected behavior (e.g., similar to qualification kits for DO178C)
- **Claim:** We need more work on
  - clear specify the scope (soundness boundaries)
  - test sets for comparing security testing tools
  - extension/adaption points for testing tools

Technology Vector: Scope, Soundness, Coverage, Bug Finding Rate

“ Security testing methods should **find real bugs**. Really?

- **Observations:**
  - **Finding real bugs** seems to be a prerequisite for getting an academic paper accepted.
  - Most “real-world” tools **find not all bugs** (the underlying theory might).
  - **Even worse:** most tools will not report anything, on our frameworks
- What I need in practice to provide the best “blend” to my developers:
  - A clear specification what it “in-scope”
  - A clear specification what it “out-scope”
  - Test cases that validate the expected behavior (e.g., similar to qualification kits for DO178C)
- **Claim:** We need more work on
  - clear specify the scope (soundness boundaries)
  - test sets for comparing security testing tools
  - extension/adaption points for testing tools

Technology Vector: Software Supply Chain Support

Might be Covered by “compositional Approaches to Testing”

> Testing is done by the developer of a software component. Really?

- Observations:
  - Software evolves over time (both, on-premise and Cloud): small changes are the norm
  - Software is build using
    - Free and Open Source Software
    - third party libraries (closed source)
    - assets of acquired companies As vendor, you are responsible for all code you ship to customers
- **Claim:** We need more research in
  - composable testing techniques, e.g.,
    - impact/change analysis for selecting test cases
    - automated inference of specifications of software components
  - in pushing testing across the whole software supply chain
    - techniques that generate “certificates”
    - formats and guidelines for exchanging “test tool configurations”

Thank you for your attention!

Any questions or remarks?

Related Publications

Ruediger Bachmann and Achim D. Brucker.
Developing secure software: A holistic approach to security testing.
Datenschutz und Datensicherheit (DuD), 38(4):257–261, April 2014.
http://www.brucker.ch/bibliography/abstract/bachmann.ea-security-testing-2014.

Achim D. Brucker, Lukas Brügger, and Burkhart Wolff.
Formal firewall conformance testing: An application of test and proof techniques.
stvr, 2014.
http://www.brucker.ch/bibliography/abstract/brucker.ea-formal-fw-testing-2014.

Achim D. Brucker and Uwe Sodan.
Deploying static application security testing on a large scale.
In Stefan Katzenbeisser, Volkmar Lotz, and Edgar Weippl, editors, gi *Sicherheit 2014*, volume 228, pages 91–101. gi, March 2014.
ISBN 978-3-88579-622-0.
http://www.brucker.ch/bibliography/abstract/brucker.ea-sast-expierences-2014.

Achim D. Brucker and Burkhart Wolff.
On theorem prover-based testing.
Formal Aspects of Computing, 25(5):683–721, 2013.
ISSN 0934-5043.
http://www.brucker.ch/bibliography/abstract/brucker.ea-theorem-prover-2012.