Security-by-Contract for Pervasive Services

Nicola Dragoni, Fabio Massacci, Ida Siaahan
(University of Trento)
www.massacci.org
www.s3ms.org

Outline
• Motivation
• Security x Contract
  – Concepts
  – Workflow
• Policy/Contract Matching
  – Automata Modulo Theory
• Conclusions

Motivation
• Today’s smart phones/nomadic devices have more computing and communication power than PCs 20 years ago, but ...
• Not even remotely the amount of third party software available for PCs at that time, and
• A long term market growth cannot be based on selling ring-tones as the only “added-value” services.

Observations
• A validation infrastructure exists
  – A signature is checked on the device;
  – No semantics is attached to it.
• Some technologies exist
  – Static analysis to prove program properties
    [Leroy et al., and many others]
  – Monitor generation for complex properties
    [Havelund & Rosu, Erlingsson & Schneider, Krkow et al. Ligatti et al.]
• Security-by-Contract (SxC) puts them together
  – Use contracts as semantics for the signatures;
  – Use static analysis and monitors as basic
### Key Concepts

- **Contract carried by application;**
  - Claimed Security behavior of application;
  - (Security) interactions with its host platform;
  - Maybe with Proof that code satisfies contract.

- **Policy specified by a platform.**
  - Desired Security behavior of application;
  - Fine-grained resource control

- **But I trust noboy, I just need policy monitor**
  - Monitoring ONLY a part of the story…

---

### SxC Workflow – User’s View

1. **Start**
2. **Check Evidence**
   - Yes → **Enforce policies**
   - No → **Match contract & policies**
3. **Match contract & policies**
   - Yes → **Perform in-lining**
   - No → **Perform run-time monitoring**
4. **Perform in-lining**
   - Yes → **Execute application**
   - No → **Enforce policies**

**REMEMBER USERS WANT TO GET THERE!**
**What’s Automata Modulo Theory?**

- **Finite State Automata**
  - They represent the security behavior (claimed or desired)
  - You should know that…

- **With “Infinite” Edges**
  - URL starting with “https://” are not that few…
  - Battery Levels less than 30%

- **BUT Finitely represented with Expressions**
  - $m=\text{Java.IO.Connector} \&\&$
  - protocol(x)=https \&\& protocol(x)==http
  - appType(x)!=jpg || appType(x)=appType(y)

- **Decidable theory for satisfiability of expressions**

---

**Why Modulo Theory**

- **Matching = Language Containment**
  - Actions allowed by contract $\subseteq$ actions allowed by policy
  - Exists Classical nested DFS

- **Search for counterexamples**
  - Path allowed by contract but NOT allowed by policy
  - Path allowed by contract and allowed by NEG policy

- **Path allowed by contract and by neg policy**
  - At run-time: two sequence of actions
  - Symbolically: two sequences of expressions
  - IF conjunction of pair of expressions SAT (modulo theory)
  - THEN exists common action…
Summary

• **Security-by-Contract**
  – Ideas stolen from Design-by-Contract (Bertrand Meyer) and Model-Carrying-Code (Sekar et al.)

• **Security must account complete lifecycle**
  – Enforcement and Development & Matching

• **Matching Policy and Contract**
  – Mapped into FSA with expressions on edges
  – If theory for deciding edges polynomial (most cases) => Practical

• EuroPKI’06 & NordSec’07

To-Do-List

• **Not all properties currently captured**
  – Connect only to an url that you have seen at the beginning of the session (or in the jar manifest etc.)
  – Requires history-dependent automata

• **More faithful to Design-by-Contract**
  – Precondition = security properties platform must guarantee (missing but easy)
  – Invariant = security behavior
  – Postcondition = services that midlet delivers? or obligations left to the platform?

• Negotiate security vs services?