Insights into Composability from Lablet Research

Jonathan Aldrich
with William Scherlis, Anupam Datta, David Garlan, Bradley Schmerl, Joshua Sunshine, Marwan Abi-Antoun, Sam Malek, Juergen Pfeffer, Christian Kaestner, Andre Platzer, Limin Jia, Robert Harper, Travis Breaux, Witawas Srisa-an, and Arbob Ahman

Science of Security
Quarterly Lablet PI Meeting
University of Maryland
October 28-29, 2014

Five Hard Problems in the Science of Security

1. Scalability and composability
2. Policy-governed secure collaboration
3. Predictive security metrics
4. Resilient architectures
5. Human behavior

H.P. Talk

Our selection criteria for the problems

- High level of technical challenge
- Significant operational value
- Likelihood of benefiting from emphasis on scientific research methods and improved measurement capabilities

Hard Problem: Composability

Challenge

- Develop methods to enable the construction of secure systems with known security properties.
  - Construct from **components** each of which has known quality and security properties
  - **Avoid** full reanalysis of the constituent components.

Motivation

- Need composition to manage
  - Increasing scale, complexity, dynamism
  - Socio-technical ecosystems, rich supply chains
  - Direct evaluation of artifacts as they are produced/evolved

The SoS Lablet Approach

(1) Advance the state of cybersecurity research

- Focus on the hardest technical problems, emphasizing (at CMU)
  - HP 1: composability of modeling and reasoning as a key to scale and incrementality
  - HP 5: human behavior and usability for developers, evaluators, operators, and end users
- Support advances in the other three HPs also:
  - Policy, Metrics, Resiliency

(2) Advance the scientific coherence of the multidisciplinary body of cybersecurity technical results

- Advance most effective scientific processes
- Acknowledge the unavoidable multidisciplinary nature of cybersecurity
- Enhance the coherence of the relevant body of technical results
- Enhance productivity, validity, and translation into practice

(3) Engage and broaden the cybersecurity technical community

- Facilitate community and educational engagement
- Subcontractor partners, workshops, and conference events

Initial Workshop on Composability

- Held September 26, 2013 at CMU

- Crosscutting principles (excerpt)
  - Assume-guarantee reasoning
  - Game theory
  - Families of systems

- Open questions (excerpt)
  - New kinds of refinement needed to preserve security properties
  - How to reason under uncertainty
  - Level of abstraction (of programming, of assurance, and relating these)
  - Managing imprecise specifications

- Impact on practice
  - Adoption barriers and incentives – making the ROI case

Priming the Discussion Pump

- Our initial meetings have been focused on work at CMU
  - But we want to gather community input
  - Composability is subtle, and this is a work in progress!

- Question for Lablet researchers: What have you learned about composability that could generalize beyond your particular research project?
  - Consider methods, results, and patterns of approach. Examples are helpful!

- We'll come back to discuss these at the end!

Composability

What is Composability?

A Software Engineering view of Composability

- Construction C is compositional w.r.t. abstraction α if
  - there is an abstract construction C^{α},
  - operating on the abstract domains A_j
  - satisfying, for arbitrary parameter values p_i

$$\alpha\big(C(p_1, \dots, p_n)\big) = C^{\alpha}\big(\alpha(p_1), \dots \alpha(p_n)\big)$$

  - whenever the left hand side is defined

Definition by Arend Rensink, presented at Dagstuhl in 2012, suggested by Christian Kaestner. Taken from http://www.dagstuhl.de/mat/Files/12/12511/12511.RensinkArend.Slides.pptx

- What does this mean in the SoS setting?

Example: Sequential Composition of Information Flow Properties

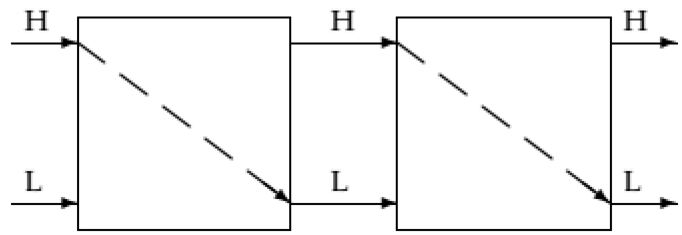


Figure 1: Sequential composition of non-interfering programs

- Confidentiality: high-security inputs do not flow to low-security outputs

- Sequential compositionality [Ahmad, Harper]
  - If two components preserve confidentiality
  - And we compose them in sequence
  - Then the result preserves confidentiality

Concurrent Composition of Information Flow Properties

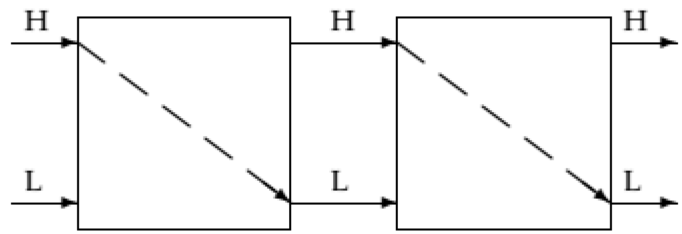


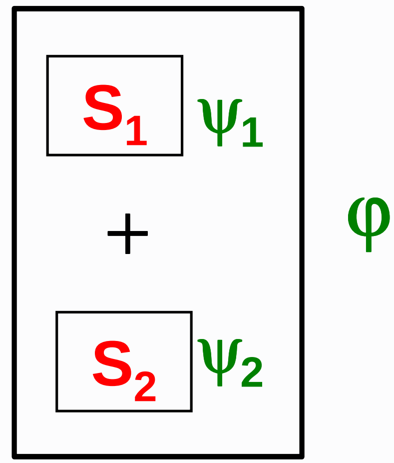
Figure 1: Sequential composition of non-interfering programs

- Confidentiality: high-security inputs do not flow to low-security outputs

- **Concurrent** compositionality
  - Presence/absence of input can be used to leak secure content!
  - Solution: the presence/absence of input also has a security level [Rafnsson *et al.* 2012, 2013]

- Current lablet research: compositional reasoning about declassification [Ahmad, Harper]

Background: Assume-Guarantee Reasoning



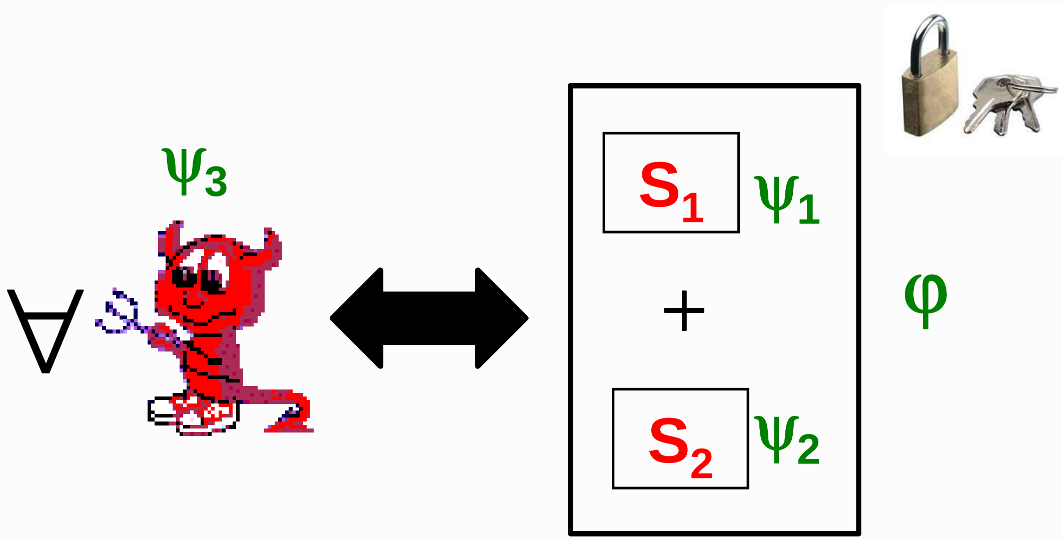
Do S_1 + S_2 satisfy a global property φ based on

- Local properties ψ_1 of S_1 and ψ_2 of S_2 that are **checkable** separately

Assume-Guarantee is a **general** technique for **composability**

Can we use it to unify approaches to **composable reasoning** about **security**?

Secure Assume-Guarantee Engineering



SAGE

Do $S_1 + S_2$ satisfy a global **security** property φ based on

- Local properties ψ_1 of S_1 and ψ_2 of S_2 that are **checkable** separately; and
- Invariant property ψ_3 of all adversaries of a certain class that is **enforceable**

[Garg, Jia, Datta et al. Logic of Secure Systems and System M]

Application Domain: Security Protocols

$\forall$ 

Security Protocols

Example: SSL/TLS

- Global property: authentication
- Local property: only send secrets encrypted with specific keys

PPT programs

Adversary invariant: Cannot forge signatures

SAGE vs traditional AG

Adversary invariants **enforced** by design of signature schemes for PPT environments

[Datta et al. Protocol Composition Logic]

Application Domain: Systems Software

$\forall$ 

Systems Software

Example: XMHF hypervisor

Global property: Integrity of hypervisor

Local property: each component updates memory protection bits safely

Interface-confined programs

Adversary invariant: Guest OS preserves safe memory protection bits

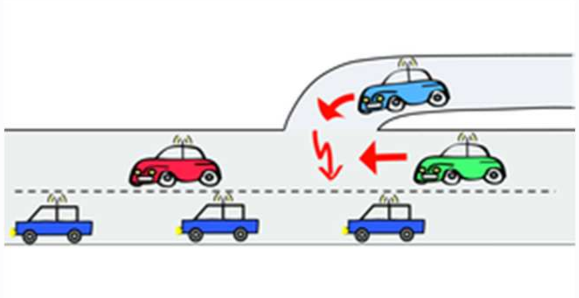
SAGE vs traditional AG

Interface-confinement of adversary code **enforced** using hardware-based interface confinement (HIC)

[Vasudevan, Chaki, Jia, Datta et al. Compositional XMHF]

Assume-Guarantee in Cyber-Physical Systems

- Known compositionality results in a software engineering setting
- New challenges adapting to secure assume-guarantee engineering
  - e.g. DARPA HACMS

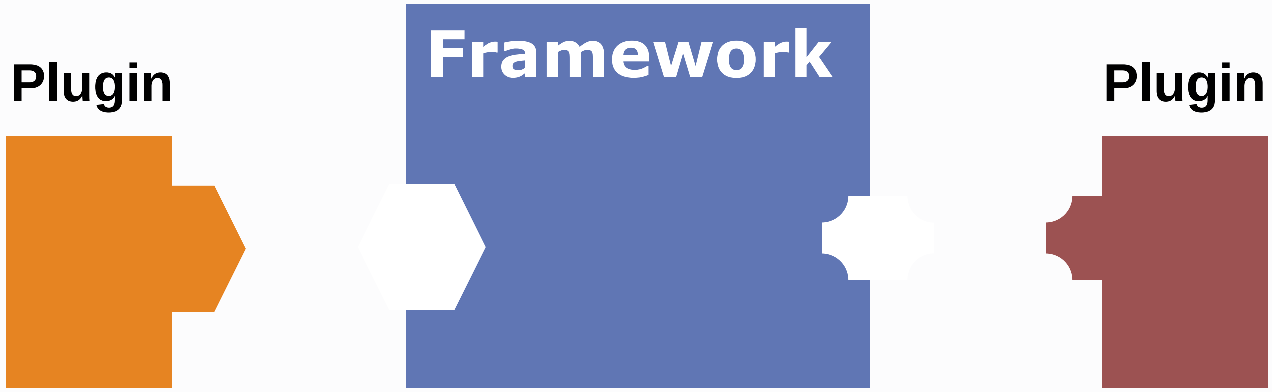


- Key question: what can we safely assume?
  - Communication happens in finite time?
    ✗ Not if an adversary can interfere
  - Our code will be run every N seconds?
    ✗ Not if an adversary refuses to yield the processor
  - No communication out of thin air?
    - This one is OK!

Potential to explore enforcement of these assumptions

Andre Platzer and Dexter Kozen, Security Reasoning for Distributed Systems with Uncertainties lablet project

Assume-Guarantee in a Framework Context



- Frameworks increasingly common form of reuse
  - Enterprise, web, mobile, etc.
- Assume-guarantee relationship with plugins
  - Framework assumes that plugins follow certain rules
    - e.g. don’t start your own thread, let framework manage network
  - As a result, framework can provide desired properties
    - Scalability, robustness, security, …
- Research challenge: enforcing rules on plugins
  - Looking at using *capabilities* to reason about what plugins can do
  - Link to interface enforcement in the Hypervisor lablet project

Garlan, Aldrich, Schmerl, Malek, Abi-Antoun: Science of Secure Frameworks lablet project

Research Challenges in Secure Assume-Guarantee Engineering

- The role of abstraction in security
  - The attacker can attempt to break our abstraction
    - e.g. timing in a CPS setting, totality in concurrent information flow
  - Research challenge to abstract the attacker
    - e.g. probabilistic polynomial time (PPT) attackers

- Interface specification and enforcement
  - Complete interface specification is an issue
    - must include anything the attacker may target, especially in open systems (e.g. Android)
  - Enforcing interface abstraction is a research challenge
    - cf. Hypervisor research, frameworks

- Diversity and dependencies between properties
  - Inter-compositionality: high-level properties build on lower-level properties

- New kinds of properties
  - Information flow is not a trace property, but a relation between traces

Discussion

- What have you learned about composability that could generalize beyond your particular research project?
  - Consider methods, results, and patterns of approach

- Do the themes above resonate with your own research?