CC Compositional Certification for MILS Virtualization Platforms


*DFKI, **EADS Innovation Works, ++Fraunhofer IESE, +SYSGO
September 18, 2012
Outline

1. From Components to Compositionality
2. Multiple Independent Levels of Security
3. Secure Gateway
4. Summary
Components

- Structuring systems into components is the main means for
  • the reuse of certain parts,
  • an efficient change management, and
  • the reduction of complexity (avoid possible exponential growth of complexity).
Components (ctd.)

- Boundaries of components.
- Interfaces between components and to the environment.
- How to exploit structure in design, analysis, and evaluation?
Compositionality

- Frege’s principle of compositionality for the semantics of language constructs: \[ str(c_0, c_1) \] = \( \text{comp}_{str}([c_0], [c_1]) \)
- In software development one needs to associate import/export descriptions (specifications) with components.
- Allows for an independent treatment of components.
Composition

- Reasoning about composition *only in terms of export/import descriptions* (as an additional step).
- Required imports may be pruned if guaranteed by other components or remain for further (iterated) compositions.

- *When is this (ideal) approach possible in security engineering?*
Compositional Evaluation

- Two approaches:
  1. CAP
  2. CCDB-2007-09-001
- Both consider a base and a dependent system that uses "services" from the base system.
- First approach can be made to cover more general schemes and iteration (according to CC Part 3, B).
- Our interest: *Instantiation of these schemes for MILS architectures.*
- Two (independent) security targets and an additional document for composition.
- Level A, B, C: Rigor in analyzing the composition.
- Emphasis on interactions between dependent and base system to guarantee objectives of the composed system.
- Restriction:

  CC Part 3 Para 141: The CAPs only consider resistance against an attacker with an attack potential up to Enhanced-Basic. This is due to the level of design information that can be provided through the ACO_DEV, limiting some of the factors associated with attack potential (knowledge of the composed TOE) and subsequently affecting the rigour of vulnerability analysis that can be performed by the evaluator. Therefore, the level of assurance in the composed TOE is limited, although the assurance in the individual components within the composed TOE may be much higher.
- Two steps: Platform and dependent system on the platform.

- A more restricted generic interaction between base and dependent system.

  15 The hardware platform has no 'strictly functional' properties related to security. It provides mechanisms to protect the composite product assets, but the composite product behaviour depends widely on the software application having to use, to configure and activate these security mechanisms.

- Motivated by hardware platforms.
Compositionality for Security

- Assurance for security goals cannot always be decomposed.
  1. Conceptual (logical) limitations like for the composition of protocols and policies.
  2. Technical problems: Shared resources make it difficult to ground logical valid designs on existing platforms.

*Multiple Independent Levels of Security* (Rushby 1981) solves (2).

- Provides support for compositional evaluations.
- Layered architecture: (MILS-) *separation platform* as a base component other components depend on.
MILS Approach

- Global security policies (like MLS type policies) are difficult to enforce (analyze, verify, evaluate) and often inadequate for large monolithic systems.

- Separation of concerns: MILS separation kernels allow for the implementation of architectures with manageable local policies that interact (only) through well defined (controllable) channels.

- Formalizations by information flow analysis and related approaches: Rushby, Di Vito, Greve&Wilding&Vanfleet.
Integration and Security in Avionics

- Integration drives security concerns in avionics.
- Functional integration: Formerly autonomous components are combined to create new functionality (old functionality largely remains).
- Architectural integration (IMA): Common hardware platform.
- Security concerns (example): air gap.
- Look for an "evaluation friendly" approach: Ongoing work in SeSaM funded by the German Federal Ministry of Education and Research (BMBF) under funding ID 01BY112[0-3].
Security Gateway - Overview

- Purpose: Control the information flow between applications of different security domains.
  - outbound component: protection against information leakage (confidentiality)
  - inbound component: protection against compromising data (integrity)

- Conceptual structure:
PikeOS as a Component

- Started to work on compositional treatment: (CAP ACO_REL1.1/CCDB ASE_COMP.1-1, CAP ACO_REL.1.2/CCDB ASE_COMP.1-2).
- A closer look at the interaction between PikeOS and the gateway:
Observations - Lessons Learned

- Communication with PikeOS *always* uses the PikeOS API.
- PikeOS *always* offers the same generic security services to all applications running on it.
  - The only application-specific aspect of the interaction is (explicitly, declaratively) given by a PikeOS *channel-flow policy*.
  - Policies as export descriptions.
  - PikeOS (will be) evaluated for arbitrary policies based on the given (fixed) PikeOS API.
  - Formal model provides semantics of policies.
Observations - Lessons Learned (ctd.)

- Only PikeOS configuration has to be assessed from an application-specific point of view.
- Guarantees (semantics) of policies are part of PikeOS ST and require no further context-dependent analysis.
- An instance of the ideal case.
- Therefore in our view: Reasons for CAP restrictions do not apply here.
- Situation similar to (hardware) platforms treated by CCDB-2007-09-001.
PikeOS Security Services

- PikeOS flow policy (abstract view)

- Main service: Applications in a partition perceive (and influence) the environment (all other partitions) only through declared interfaces (to channels).

- Ongoing work on formalization.

- (Formal) export specification for PikeOS to be used for compositional reasoning at the application level.
Import to Gateway Scenario

- Integrity of gateway data (configuration) as far as access (dependencies) from outside are concerned.
- No direct communication between green and brown application (no bypass).
- Instantiation of PikeOS (for the gateway scenario) by configuration of a policy.
- Contribution of PikeOS can be expressed in a formal way.
Objectives of the Gateway

- Filtering mechanism: O.ADDRESS_FILTERING, O.CLEAN_PROTOCOL_HEADER, O.RESIDUAL_INFORMATION

- Implementation uses functionality of the platform.
  - no security services
  - application-independent
  - execution mechanism (similar to hardware platforms)

- In general several local security mechanisms may interact through PikeOS channels.
  - modular architectures to be supported by MILS
  - CCDB-2007-09-001 possibly not general enough
  - But role of PikeOS remains the same.
Hardware as a Component

- Open Question: Consider hardware platform as a separate component?
- Use of hardware might lead to illicit information flows.
- Need assumptions (import) for hardware.
- More complex configuration issues: Requirements from the application (for the final PikeOS flow policy) might force certain hardware constellations (to meet the assumptions).
Summary

- CAP scheme captures (many/all) essential aspects of compositional design.
- Flexible enough for MILS architectures as opposed to CCDB-2007-09-001.
- There are situations (as in case of PikeOS) where reasons for restrictions do not apply.
- Worthwhile to characterize these situations where composition can be fully analyzed in terms of export/import?
- May be useful for composition on top of the separation platform.