Compositional Security Evaluation: The MILS approach

John Rushby and Rance DeLong[†]

Computer Science Laboratory
SRI International
Menlo Park CA USA

[†]Primary affiliation: LynuxWorks

Systems, Components, and Security

- Security is a system property
- But there is a compelling case to establish a marketplace for security-relevant components
  - Secure file systems, communications subsystems, operating system kernels
  - Filters, downgraders, authentication services
- Want the security of these components to be evaluated
- In such a way that security evaluation for a system built on these is largely based on prior evaluations of the components
- This is an example of compositional assurance

Component-Based Design and Compositional Assurance

- Component-based design
  - Take some off the shelf components
  - Build some bespoke components
  - Connect them all together with glue (components)

  To achieve the required functionality
  - We understand the functionality of the system by understanding the functions of its components
- Compositional assurance
  - The idea that we can provide assurance for properties of a component-based system based on preconstructed assurance for properties of its components

Why Is Compositional Assurance Hard?

- Assurance must consider properties, not just function
  - Properties depend on component interactions as much as on individual component behavior
  - And must consider what must not happen
- Assurance must consider faults and malice
  - Including those that subvert the design
  - In particular, those that vitiate the separation into components and bypass the interfaces between them
  - i.e., those that create unintended interactions
- So assurance for components must anticipate this and provide very strong guarantees, and must consider interactions as well as local behavior

State of Practice in Compositional Assurance

- Not endorsed by any stringent certification regime we are familiar with
  - Because of the interaction issue: the current way to deal with this is to look at the whole system and inside every component
- E.g., the FAA certifies only airplanes, engines, propellers
  - Some weak mechanisms for components
    - ⋆ Reusable Software Components (AC 20-148)
  - And for incremental construction of certification
    - ⋆ Integrated Modular Avionics (DO-297/ED-124)
  - But the initial certification is always whole system, not compositional, and they reserve the right to look inside components
- Doing security evaluations compositionally would be a first!

Compositional Security Evaluation

Two topics:

Procedural: how to do this within the CC framework?

Technical: the approach we're developing for MILS

Compositional Evaluation Within The CC Framework

- A community effort is needed to explore this
  - Our goal here is to stimulate debate
- For MILS, we are using an ad-hoc approach
  - Developing a MILS Integration Protection Profile (MIPP)
- Use a PP because that is an established mechanism
- But it's really a meta-PP
  - Constraints on component PPs so that they compose to yield system-level assurance

The MILS Approach

- Historically, MILS stood for
  Multiple Independent Levels of Security
- Now, its best thought of as a name for . . .
- . . . a class of security architectures based on two levels

  Subject Interaction Level: the security attributes of encapsulated subjects, and their interactions

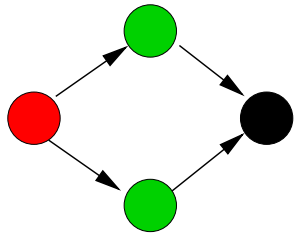
  Resource Sharing Level: the security attributes that arise from subjects sharing resources
- Each level has its own kind of component

  Subject Interaction Level: operational components

  Resource Sharing Level: foundational components

  And these each have their own kind of PP

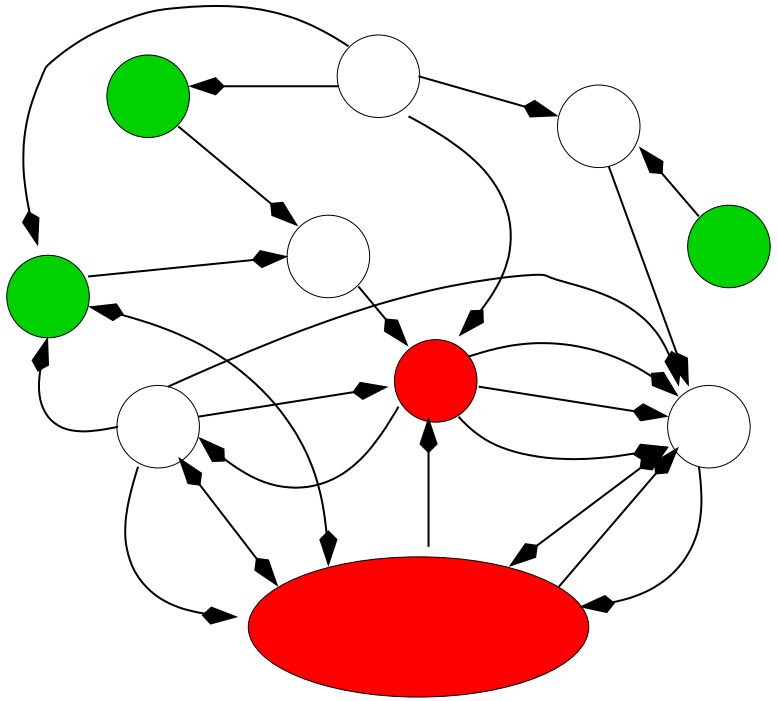
Intuitive Security Architecture



- Almost all system designs are portrayed in diagrams using circles and arrows
- But in security, these have a particular (often unconscious) force and interpretation
- Circles indicate subjects,
  - Encapsulated data, information, control,
- Arrows indicate interactions (and interfaces)
  - Absence of an arrow means absence of interaction
- The only things that happen inside a circle are consequences of things in that circle and the incoming arrows, and the only things that change are the internal state of the circle and its outgoing arrows (i.e., interacting state machines)

Good Intuitive Security Architecture

- Try to arrange the circles and arrows so that
  security depends on only a few trusted circles
- And those are trusted to do only relatively simple things
- Split big circles up if necessary to achieve these

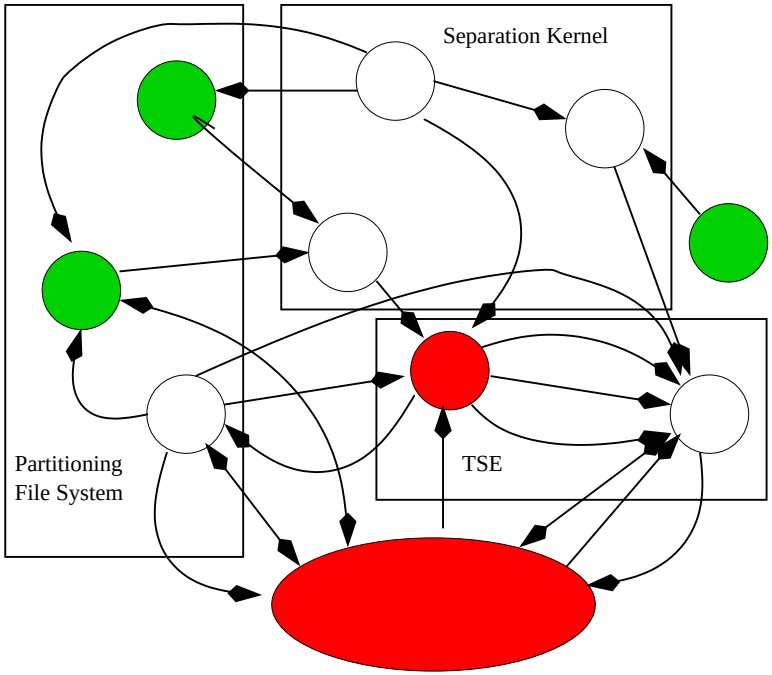


The MILS Architecture, Upper Level

- The system structure should directly reflect the circles and arrows picture
  - i.e., the implementation directly follows the logical design
  - Circles are subjects, arrows are interactions
- We can afford to have lots of circles and arrows, and should use this to reduce and simplify the trusted circles/subjects
  - Trusted subjects implement policy

The MILS Architecture, Lower Level

- We can afford to have lots of circles and arrows because we can efficiently and securely share physical resources among separate logical circles and arrows



Secure sharing is ensured by foundational components, which enforce partitioning/separation

- Allocation of logical components to shared resources has impacts beyond security (faults, performance)—care needed

Two Kinds of Components, Two Kinds of PPs

The lower and upper levels of the MILS architecture have different concerns and are realized by different kinds of components having different kinds of PPs

Upper level: components that provide or enforce specific security policy functionality

- Examples: downgrading, authentication, MLS flow
- Their PPs are operational: concerned with the specific security policy that they provide

Lower level: components that partition/separate/securely share physical resources among logical entities

- Examples: separation kernel, partitioning communication system, console, file system, network stack
- Their PPs are foundational: concerned with partitioning/separation/secure sharing

The Essence of MILS

- The operational and foundational security concerns are kept separate
  - Separate kinds of components
  - Separate kinds of PPs
- Cf. traditional security kernels
  - One component partitioned many kinds of resources
    - ⋆ complex implementation
  - And either enforced a single operational security policy
    - ⋆ too rigid to be useful

    Or several
    - ⋆ too complicated to be credible
- MILS is feasible today because we know how to do fine grain partitioning (e.g., paravirtualization), have better hardware support, and can afford the overhead

Two Kinds of Components, Three Kinds of Composition

We need to consider three kinds of component compositions

operational/operational: need compositionality

foundational/operational: need composability

foundational/foundational: need additivity

Consider these in turn

Compositionality

Operational components combine in a way that ensures **compositionality**

- There's some way to calculate the properties of interacting operational components from the properties of the components (with no need to look inside), e.g.:
  - Component A guarantees P if environment ensures Q
  - Component B guarantees Q if environment ensures P
  - Conclude that $A \,||\, B$ guarantees P and Q

  This is called assume-guarantee reasoning
- Requires components interact only through explicit computational mechanisms (e.g., shared variables)
  And that is what the foundational components guarantee

Composability

Foundational components ensure **composability** of operational components

- Properties of a collection of interacting operational components are preserved when they are placed (suitably) in the environment provided by a collection of foundational components
  - Circles behave the same when put in a box
- Hence foundational components do not get in the way
- And the combination is itself composable
  - A box containing circles still behaves as a box
- Hence operational components cannot interfere with each other nor with the foundational ones

Partitioning/separation are ways to provide composability

Additivity

Foundational components compose with each other **additively**

- e.g., **partitioning(kernel)** + **partitioning(network)** provides **partitioning(kernel + network)**

There is an asymmetry: partitioning network stacks and file systems and so on run as clients of the partitioning kernel

Ensuring Compositionality, Composability, and Additivity

- How to achieve these properties?
- There are two aspects

  Theory: developing/applying the computer science to understand and achieve these

  Application: interpreting and formulating the science in a manner consistent, as far as possible, with the CC and existing PPs

- Next few slides sketch some of the issues

Operational PPs and Compositionality

- We might like to specify some policies of operational components in terms of information flow
- Well-known that many flow properties do not compose
  - e.g., noninterference

  They don't refine, either
- And compositional flow properties are nonintuitive
  - e.g., restrictiveness
- Much of this is because flow security is not a property
  - A property is a subset of possible traces (behaviors)
  - But we cannot tell if a given trace is flow secure without knowing what other traces there might be
- In practice, flow security is enforced by requiring something stronger that is a property (e.g., unwinding)

Operational PPs and Compositionality (ctd.)

- We suspect that if operational PPs specify claims that are properties then we can use CS-style compositional reasoning
  - E.g., assume-guarantee (seen earlier)
  - There may even be end-to-end flow interpretations of these (cf. work of Ron van der Meyden)
  - Gets trickier when there is physical plant involved (e.g., hybrid systems, like aircraft) because there can be interaction through the plant
- Impact on PPs: **metarequirement**
  - Operational policies must be properties

Foundational PPs

- All these deliver the claim of separation/partitioning
  - No interaction among subjects (circles) except through specified channels (arrows)
- Specialized to the kind of entities considered

  Processor Partitions: separation kernel (SKPP)

  Communications: partitioning communication system (PCSPP)

  Screen real estate: console subsystem (MCSPP)

  Files: file system (MFSPP)

  TCP/IP networks: protocol stack (MNSPP)

  etc.

Foundational PPs and Composability

- This is what separation (as in separation kernel) is about
- Separation must have as its essence the guarantee of composability for operational components
- We believe this follows when the foundational components guarantee the integrity of the interfaces of their clients
- It's not yet clear whether composability constrains operational PPs and their policies to have a certain form

Foundational PPs and Additivity

- We suspect additivity of foundational components follows if all their PPs subscribe to a common notion of separation/partitioning
- And a common security environment
  - Common set of foundational threats
  - Common assumptions and organizational policies
- But respecting (all and only) the essential differences among the different components
  - e.g., computation vs. storage vs. communication
- Impact on PPs: **harmonization**

Summary

- Compositional evaluation is needed to support a component-based approach secure systems design
  - Will facilitate COTS marketplace for secure components
  - Protection Profiles for components need to have a form that supports composition
- We have presented a technical approach to compositional evaluation being developed for MILS
  - Will it apply beyond MILS?
- We propose to develop this into a MILS Integration Protection Profile (MIPP)
  - Is a PP the right vehicle?
- We wish to encourage community involvement in the topic of criteria for compositional evaluation
- Could have larger impact beyond security