MILS Integration PP, presentation to OG RTES, 26 October 2006, Lisbon.

MILS Integration Protection Profile

John Rushby, Rance DeLong[a]

Computer Science Laboratory

SRI International

Menlo Park CA USA

[a]Primary affiliation: LynuxWorks

Need for an Integration Protection Profile

- Security is a system property
- Existing MILS protection profiles (PPs) are for components
- How do we know that a system composed of evaluated components is secure?
    - And how is the evaluation for the system constructed from the evaluations of its components?
- This is what the MILS Integration PP (MIPP) must address
- It is an instance of compositional certification
- A bold vision that pushes the state of the art

MIPP First Draft

- Redirection of MNSPP activity
- Sponsored by AFRL through Raytheon
- Will provide an account of the MILS “idea” and architecture
- And and describe the path from Common Criteria (CC) through multiple PPs to a TOE
- And how MILS components and PPs compose to yield a secure and evaluable system
- First draft is a 10-page overview
- Available in a couple of weeks
- Closely related to the MILS PP Authoring Environment
- Funding to continue the effort is expected

Compositional Certification

- Because safety, security, etc. are system properties, traditional certification regimes consider only complete systems (or major components)
  - E.g., the FAA certifies only airplanes, engines, propellers
- Even when component already evaluated as part of another system, certifiers reserve right to look inside (cf. RSC)
- But modern business practices (outsourcing, COTS) make this increasingly untenable, even in first use of a component
  - System integrator, let alone system certifier, may have little visibility into the component
  - They merely define its requirements
- The component should be evaluated separately
  - Evaluation is in terms of properties delivered at interfaces
- System certification is then built on these interfaces and properties, with no looking inside

Compositional Certification for MILS

- Feasibility of compositional certification depends on the architecture
- Because compositional certification is all about properties delivered at interfaces, we need
  - Precise interfaces (the paths for component interaction)
    - There must be no paths for component interaction outside the known interfaces, even in the presence of faults, or of malice in untrusted components
  - Precise properties
    - Must be meaningful at interfaces
      - So they can be evaluated locally
    - Must be meaningful in combination
      - So they compose to yield evaluable system properties
- MILS is an architecture that promotes these characteristics

The MILS Architecture

- An architecture stands in relation to systems design as the US Constitution stands in relation to laws
- The constitution is not a law
  - It defines the allowable kinds of law
- An architecture is not a system
  - It defines the allowable kinds of system
- If the only purpose of the MILS architecture were to promote compositional certification, it might be easy
- But MILS must promote several goals

MILS Goals

These include

- **Security**
  - Security includes many notions, such as confidentiality, integrity, access control, authorized flow, authorized actions, and is often required in combination with other difficult properties, such as safety
- **Assurance**
  - i.e., compositional certification
- **Functionality**
  - The system must achieve its operational purpose, which is usually about something other than security
- **Affordability**

Previous approaches to computer security failed on one or more, or all of these

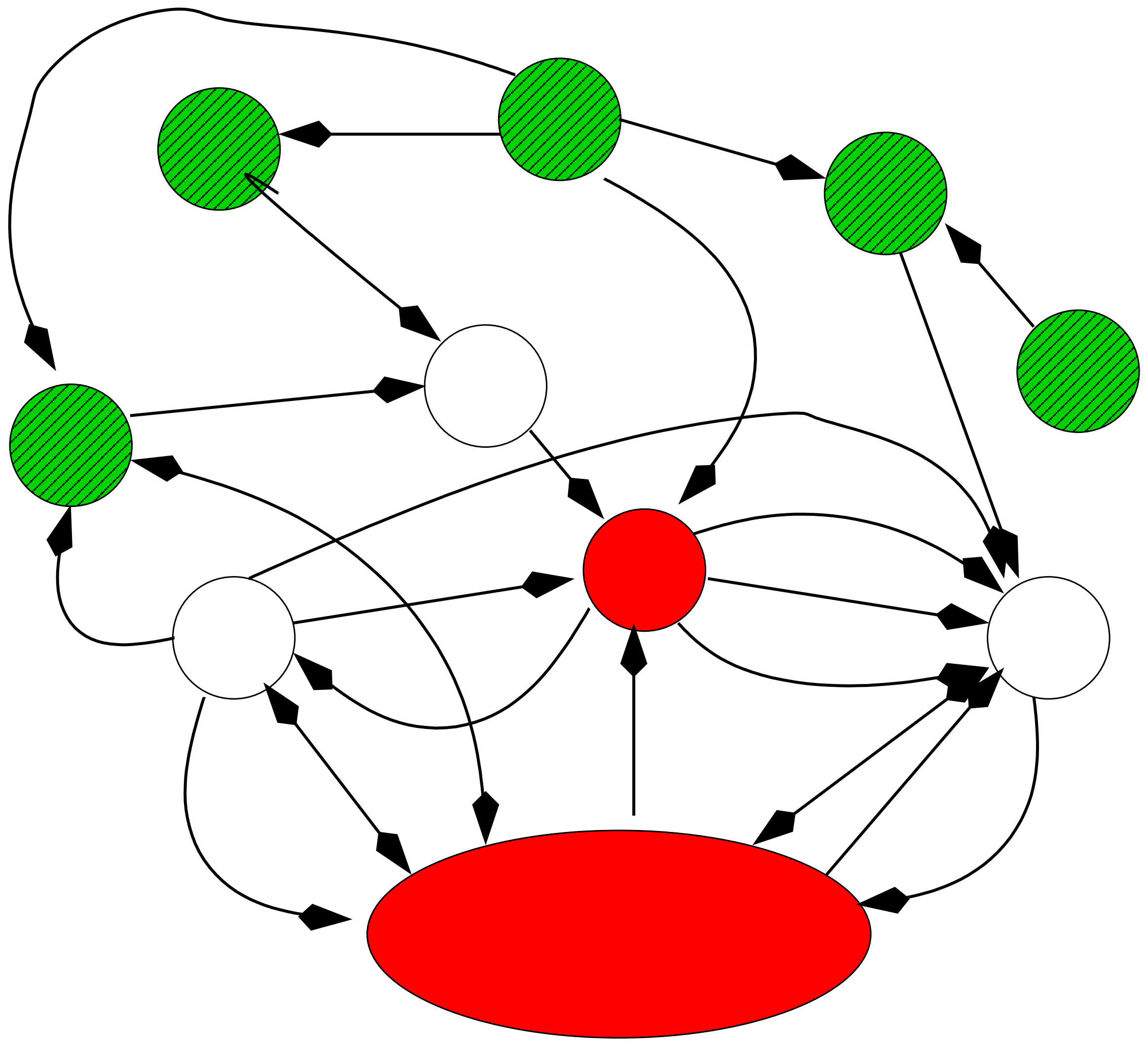
Affordability

- A reasonable expectation is that affordability will be promoted by a COTS competitive marketplace
- So we need open standards, large market, many suppliers
- The MILS component PPs (separation kernel, partitioning communication system, console, file system, network stack) are open standards intended to promote a COTS market
- Makes sense to develop these first so that suppliers have time to develop products
- But this bottom-up initiative must be complemented by a top-down one that helps systems integrators understand how to use these components
- That's the purpose of the MIPP

Top-Down Security

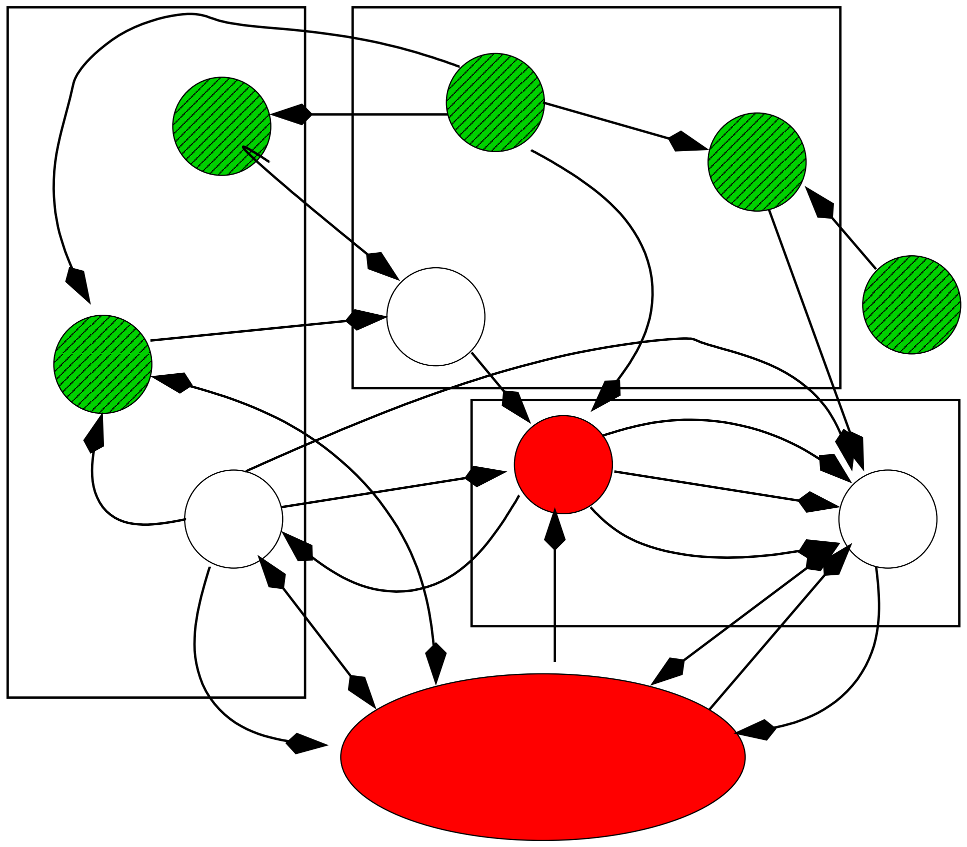
- Almost all system designs are portrayed in diagrams using circles (or boxes) and arrows
- But in security, these have a particular (often unconscious) force and interpretation
- Arrows indicate interfaces
  - Implicitly, absence of an arrow means absence of component interaction
- Circles indicate encapsulated data, information, control, etc.
  - 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

Picture



The Essence of Good Security Design

- 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



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
- We can afford to have lots of circle and arrows, and should use this to reduce and simplify the trusted circles

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
- Care and skill needed to determine which logical components share physical resources (performance, faults)
- Picture 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

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

Upper level: components that provide or enforce specific security functionality

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

The Essence of MILS

- The foundational and operational security concerns are kept separate
  - Separate kinds of components
  - Separate kinds of PPs
- Cf. traditional security kernels, where one component partitioned many kinds of resources (complex implementation), and either enforced a single operational security property (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

Composition of MILS Components

The foundational and the operational components and PPs compose differently

- Foundational components compose with each other **additively**
  - e.g., **partitioning(kernel)** + **partitioning(file system)** provides **partitioning(kernel + file system)**
- 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)

Composition of MILS Components (ctd)

- 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

Ensuring Additivity, Compositionality, and Composability

There are two aspects

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

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

The Essence of Certification

- All certification is based on **arguments** that purport to justify certain **claims**, based on documented **evidence**
- In some regimes (e.g., security), deployment decisions (i.e., judgments about the value of the claims) are separate from judgment of the credibility of the claims; in others (e.g., civil aircraft) they are combined
- Evidence may be measured facts about a system (e.g., static analysis, tests, peer review, operational history of similar systems), or claims about subsystems (supported by lower level certification)
- The evidence–arguments–claims structure is called an assurance case
- Two approaches to certification: implicit (standards based), and explicit

The Implicit (Standards Based) Approach to Certification

- Based on experience, recorded in standards and guidelines
  - e.g., DO-178B for airborne software in civil aircraft
- Follow a prescribed method (or prescribed processes)
- Deliver prescribed outputs (evidence)
  - e.g., documented requirements, designs, analyses, tests and outcomes, traceability among these
- Certification is primarily based on examination of evidence
- The argument that connects the method and the evidence to the claims is unstated (i.e., implicit)
  - Often the claims are implicit, too

So the actual assurance case is highly implicit

The Explicit (Goal Based) Approach to Certification

- Requires an explicit assurance case
  - e.g., UK air traffic management (CAP670 SW01)
- Outline form of the case may be specified by standards or regulation (e.g., MOD DefStan 00-56), but the precise form will be nominated by the applicant, who must
  - Make an explicit set of goals or claims
  - Provide supporting evidence for the claims
  - And arguments that link the evidence to the claims
    - ⋆ Make clear the underlying assumptions and judgments
    - ⋆ Should allow different viewpoints and levels of detail
- Certification, by independent assessors, considers all aspects of the case

Evidence and Arguments

Evidence can be facts, assumptions, or sub-claims (from a lower level case)

Arguments can be

Analytic: can be repeated and checked by others, and potentially by machine

- e.g., logical proofs, calculations, tests
- Probabilistic (quantitative statistical) reasoning is a special case

Reviews: based on human judgment and consensus

- e.g., code walkthroughs

Qualitative: have an indirect or implicit link to claims

- e.g., CMI levels, staff skills and experience

Critique of Standards-Based Approaches

- When the claims, arguments, and assumptions are reconstructed and made explicit, many of the arguments turn out to be qualitative
  - Requirements to follow certain design practices
  - Requirements for "safe subsets" of C, C++ and other coding standards (JSF standard is a 1 mbyte Word file)
- No evidence these are effective, some contrary evidence
- Compare implicit evidence from MISRA C vs. explicit evidence with SPARK ADA (using the Examiner)

Evaluation of Standards-Based Approaches

- Work well in fields that are stable or change slowly
  - Can institutionalize lessons learned, best practice
    - e.g. evolution of DO-178 from A to B to C
- But less suitable with novel problems, solutions, methods
- For MILS we should be considering explicit assurance cases

Further Critique of Standards-Based Approaches

- Even when analytic evidence and arguments are employed, their selection and degree of application are often based on qualitative judgments
    - MC/DC tests for DO-178B Level A but not Level B
    - Formal specifications (but not formal analysis) at some IEC 61508 SIL levels
- "Because we cannot demonstrate how well we've done, we'll show how hard we've tried"
    - And for really critical components, we'll try harder
    - This is the notion of software integrity levels (SILs)
    - Little evidence what works, nor that more is better

SILs and Qualitative Selection of Evidence

- Some components may be subjected to less severe threats than others
- Or the consequences of their failure may be less severe
- So they are less critical
- Should we provide weaker evidence for the same claims
- Or strong evidence for weaker claims?

Assurance Cases, The CC and PPs

- The CC predated the emergence of explicit assurance cases
- But has many of their characteristics
    - Tells you what to think about, not what to do
- PPs specialize the CC requirements toward specific classes of system and component so that the developer of a specific TOE has clear guidelines to follow
- Each PP should provide the framework for an explicit assurance case
    - Provides the claims
    - Specifies evidence to produce (and methods to follow)
    - Provides the argument linking evidence to claims

  It becomes a complete assurance case when the TOE developer supplies the evidence

Foundational PPs

- All these deliver the claim of separation/partitioning
  - No interaction among entities (circles) except through specified channels (arrows)
- But 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 Additivity

- I suspect we can get additivity of foundational components if all their PPs subscribe to a common notion of separation/partitioning
- And a common security environment
  - Common set of foundational threats (Mark Vanfleet)
  - 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**

Foundational PPs and Composability

- The notion of separation/partitioning must have as its essence the guarantee of composability for operational components
- This follows (I think) when the foundational components guarantee the integrity of the interfaces of their clients
- It's not yet clear to me whether this requires the operational PPs and their claims to have a certain form

Operational PPs and Compositionality

- Well-known that many security properties do not compose
  - e.g., noninterference

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

Operational PPs and Compositionality (ctd.)

- I suspect that if operational PPs specify claims that are properties then we can use CS-style compositional reasoning
  - I tried this some years ago applying McMillan's compositional method to the TTA architecture–it looks feasible
  - Gets trickier when there is physical plant involved (i.e., hybrid systems, like aircraft) because their can be interaction through the plant
- Impact on PPs: **metarequirement**

Summary

- This needs to be a community effort
- First draft (in a couple of weeks) is a basis for discussion
- A practical, effective, and community-endorsed approach is a prerequisite for MILS success
- Many areas want compositional certification, so MILS success would have large impact