Layered Assurance Workshop 13, 14 August 2008, BWI Hilton
based on Open Group, 23 July 2008, Chicago

MILS Policy Architecture And Layered Assurance

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Compositional Assurance

- We build systems from components
- And we'd like critical properties and assurance to compose
  - That is, assurance for the whole is built on assurance for the components
- Seldom happens: assurance dives into everything
- The system assurance argument may not decompose on architectural lines
  - So what is architecture?
  - A good one simplifies the assurance case

The MILS Idea

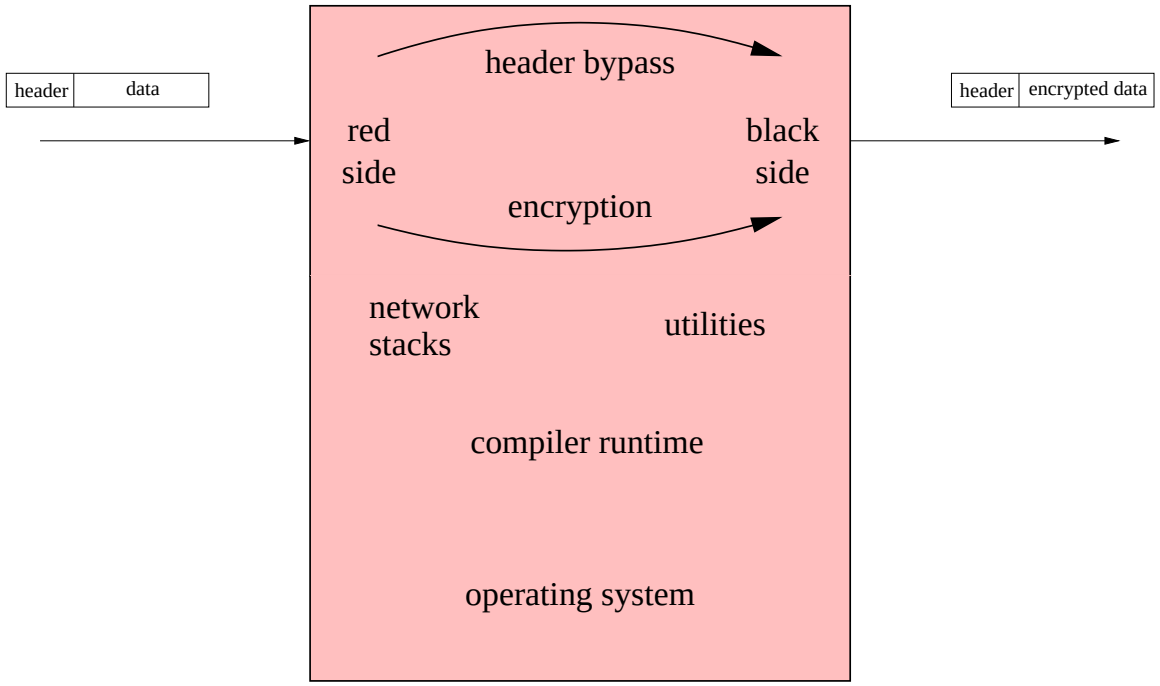
- Construct an architecture so that assurance does decompose along structural lines
- Two issues in security:
  - Enforce the security policy
  - Manage shared resources securely
- The MILS idea is to handle these separately
- Focus the system architecture on simplifying the argument for policy enforcement
  - Hence policy architecture
- The policy architecture becomes the interface between the the two issues

Policy Architecture

- Intuitively, a boxes and arrows diagram
  - There is a formal model for this
- Boxes encapsulate data, information, control
  - Access only local state, incoming communications
  - i.e., they are state machines
- Arrows are channels for information flow
  - Strictly unidirectional
  - Absence of arrows is often crucial
- Some boxes are trusted to enforce local security policies
- Want the trusted boxes to be as simple as possible
- Decompose the policy architecture to achieve this
- Assume boxes and arrows are free

Crypto Controller Example: Step 1

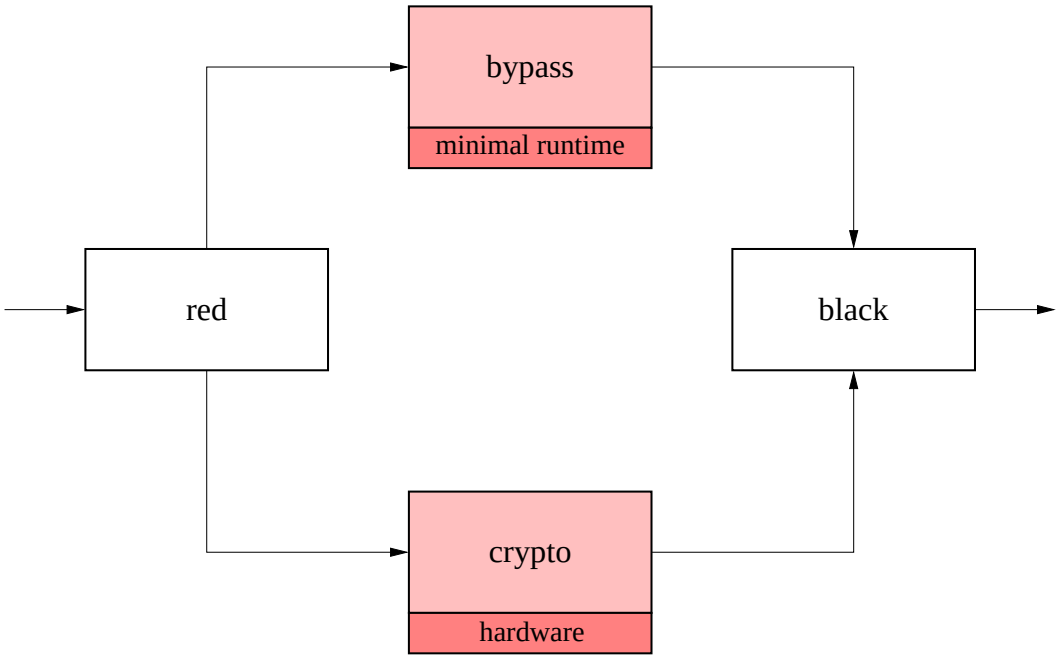
Policy: no plaintext on black network



No architecture, everything trusted

Crypto Controller Example: Step 2

Good policy architecture: fewer things trusted



Local policies:

Header bypass: low bandwidth, data looks like headers

Crypto: all output encrypted

Policy Architecture: Compositional Assurance

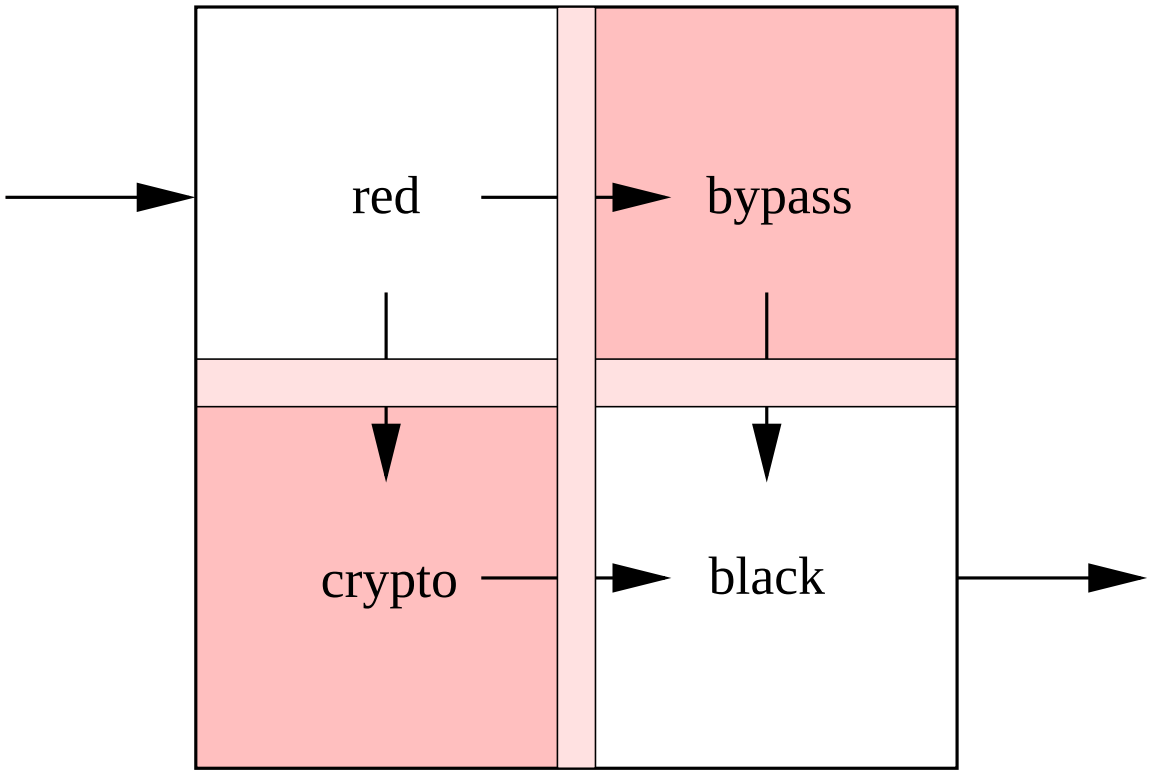
- Construct assurance for each trusted component individually
  - i.e., each component enforces its local policy
- Then provide an argument that the local policies
  - In the context of the policy architecture

  Combine to achieve the overall system policy
- Medium robustness: this is done informally
- High robustness: this is done formally (compositional verification)
- Cf. layered assurance

Resource Sharing

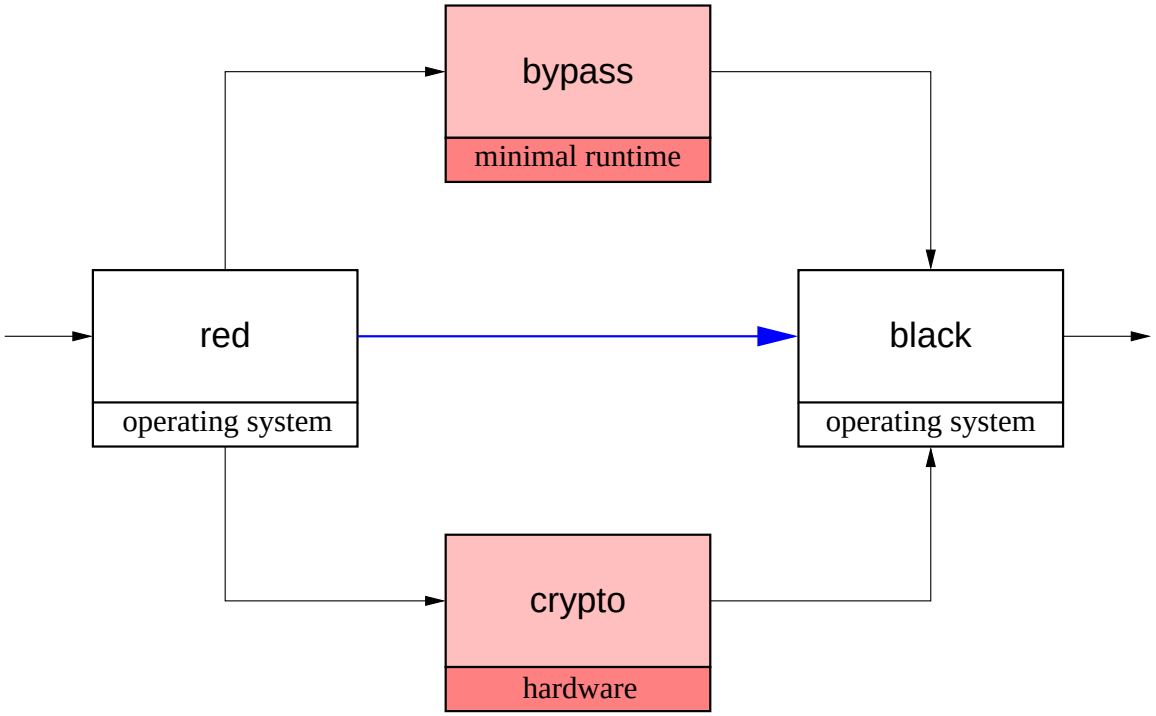
- Next, we need to implement the logical components and the communications of the policy architecture in an affordable manner
- Allow different components and communications to share resources
- Need to be sure the sharing does not violate the policy architecture
  - Flaws might add new communications paths
  - Might blur the separation between components

Uncontrolled Resource Sharing

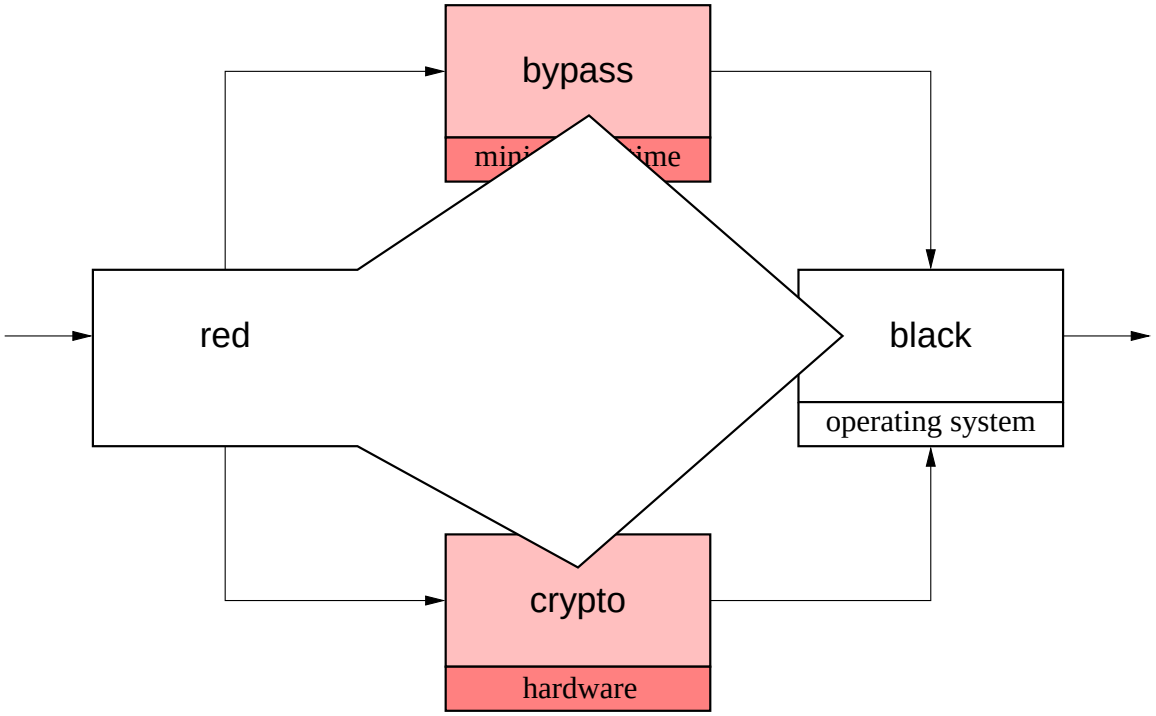


Naive sharing could allow direct red to black information flow, or could blur the integrity of the components

Unintended Communications Paths



Blurred Separation Between Components

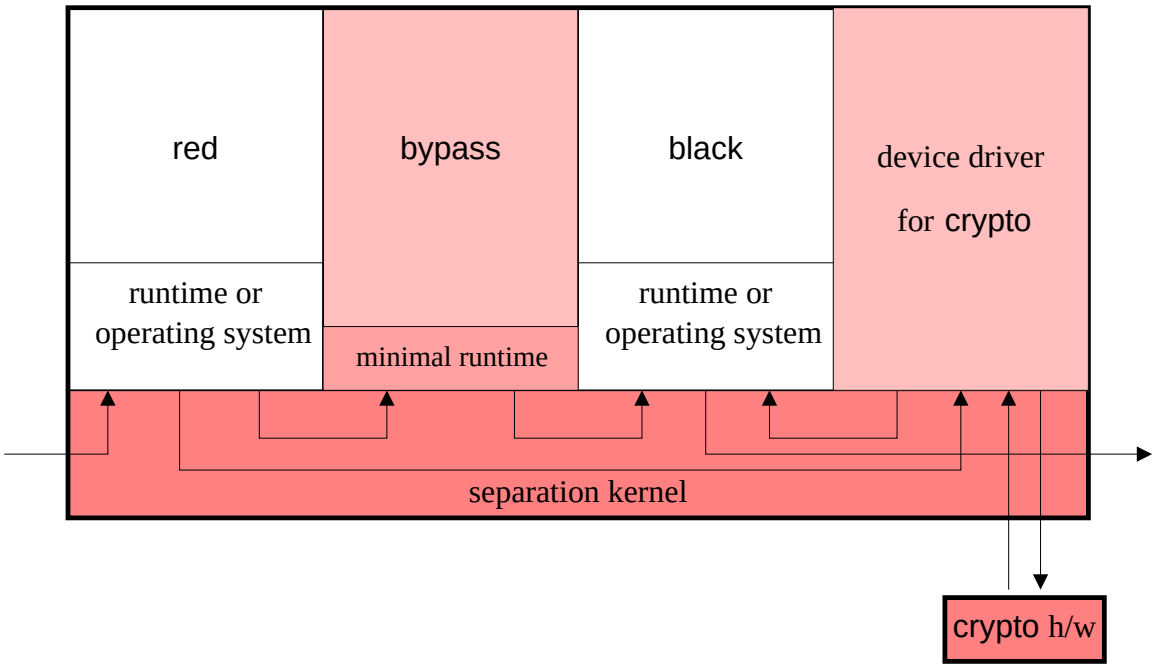


Secure Resource Sharing

- For broadly useful classes of resources
  - e.g., file systems, networks, consoles, processors
- Provide implementations that can be shared securely
- Start by defining what it means to partition specific kinds of resource into separate logical components
- Definition in the form of a protection profile (PP)
  - e.g. separation kernel protection profile (SKPP)
  - or network subsystem PP, filesystem PP, etc.

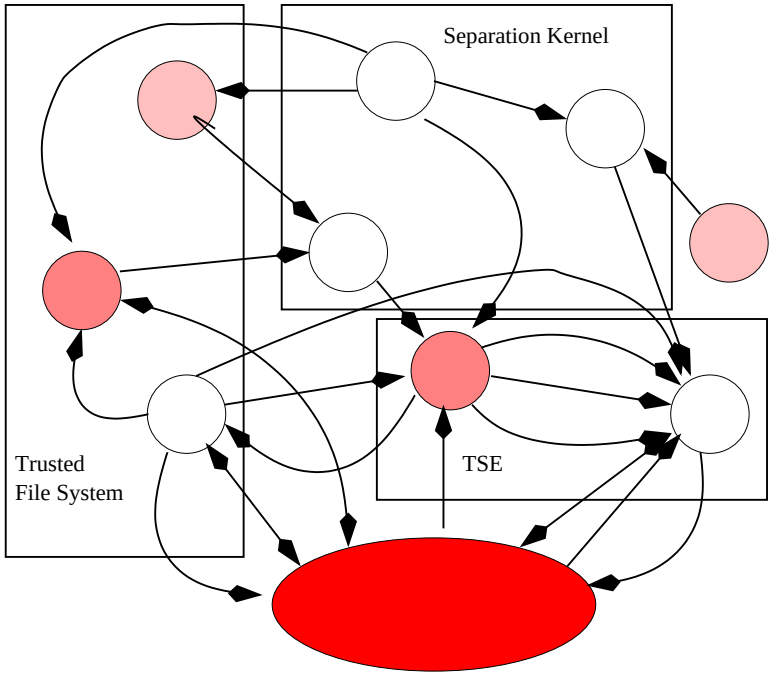
Crypto Controller Example: Step 3

Separation kernel securely partitions the processor resource



The integrity of the policy architecture is preserved

A Generic MILS System



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

Resource Sharing: Compositional Assurance

- Construct assurance for each resource sharing component individually
  - i.e., each component enforces separation
- Then provide an argument that the individual components
  - Are additively compositional

  And threfore combine to create the policy architecture
- Medium robustness: this is done informally
- High robustness: this is done formally (compositional verification)
- Cf. layered assurance

MILS Business Model

- DoD moves things forward by supporting development of protection profiles
  - Separation kernels, partitioning communications systems, TCP/IP network stacks, file systems, consoles, publish-subscribe
- Then vendors create a COTS marketplace of compliant components
- Currently they are all resource sharing components; should be some policy components, too
  - e.g., filters, downgraders for CDS

MILS In The Enterprise

- Separation kernels are like minimal hypervisors (cf. Xen)
  - MILS separation kernel (4 KSLOC), EAL7
  - Avionics partitioning kernel (20 KSLOC), DO-178B Level A ($\approx$ EAL4)
  - Hypervisor (60 KSLOC), EAL?
- Can expect some convergence of APIs (cf. ARINC 653)
- Different vendors will offer different functionality/assurance tradeoffs
- Can extend use of hypervisors from providing isolated virtual hosts to supporting the policy architecture of a secure service

Recent Progress

- Initial development of mathematical theory for compositional assurance of MILS systems
- Initial development (by Rance DeLong) of a Common Criteria Authoring Environment to assist construction of coherent PPs
- PPs for several MILS components at different levels of completion
  - SKPP done, PCSPP nearly done
  - Console, network, filesystem, under way
- High and medium robustness separation kernels from several RTOS vendors

Summary

- Key idea of MILS is to align the architecture with the assurance case
- Enabler for this is separation of concerns
  - Enforcing policy
  - Sharing resources
- The policy architecture is the interface between these
- Efficient and secure resource sharing allows the policy architecture to have many logically separate components and communications
  - Use this to simplify the trusted components
  - Which eases their assurance
- Assured resource sharing components are COTS
- Assurance for the system is composed from that of the components

Thanks

- To Carolyn Boettcher of Raytheon and Wilmar Sifre of AFRL
- And to Rance DeLong, Joe Bergmann and members of RTES