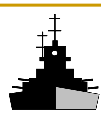
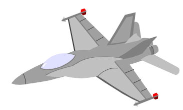
High-Assurance,
Real-time MILS Architecture

W. Mark Vanfleet
Senior NSA/IAD
Security Analyst
wvanflee@restarea.ncsc.mil

Bill Beckwith
Objective Interface Systems
CEO/CTO
bill.beckwith@ois.com



This presentation represents joint research between the
Air Force, Army, Navy, NSA, Boeing, Lockheed Martin, Objective Interface,
Green Hills, LynuxWorks, Wind River, GD, Rockwell Collins, MITRE, U of Idaho



Why Security in Embedded Systems?

Why Security in Embedded Systems?

- A Year in the Life of a Utility System
  - 100 - 150 hits/day on control network
  - 17 intrusions
  - 2 Denial of Service (DoS) events
  - 3 Loss of Control Events
    - Switchgear controller
    - Boiler Deaerator controls

Why Security in Embedded Systems? (cont.)

- To reduce the risk of
  - Damage to internal systems
    - Expense to repair
    - Operational blockage
  - Damage to customer systems
    - Loss of goodwill
    - Market recognition: irreparable damage to quality image
    - Liability for customer losses
  - Peace of mind
    - Spend attention on forward initiatives
    - Less monitoring required
    - Sleep better
- To reduce costs
  - Potentially lower insurance
  - Less monitoring costs

Why Security in Embedded Systems? (cont.)

- Australian Water Utility
  - Vitek Boden, 48, April 23rd, 2000, Queensland, Australia
    - disgruntled ex-employee of equipment supplier
    - Vehicle became command center for sewage treatment
    - Controlled 300 SCADA water and sewage nodes
    - "*was* the central control system" during intrusions
    - Released millions of liters of sewage
    - Killed marine life, blackened creek water, bad stench
  - Caught on 46th attempt
    - Was angling for a consulting job to "fix" the problems he caused
- Result of embedded systems without security
- SCADA: Supervisory Control And Data Acquisition

Why Security in Embedded Systems? (cont.)

"The problem is that programmable logic controllers, digital control systems, and supervisory control and data acquisition, or SCADA, systems were never designed with security in mind"

- "SCADA vs. the hackers", *Mechanical Engineering*, December 2002

- Existing SCADA systems lack authentication of administrators and operators
- Mechanical engineers recognize the problem

Security Evolution

Security Evolution
Fail-first Patch-later

- Most commercial computer security architectures
  - Reactive result of problems
    - Virus
    - Worms
    - Hacker
  - The result of systems software where security is an afterthought
    - Operating systems
    - Communications architectures
- Inappropriate approach
  - For the communications infrastructure
  - Or any other mission-critical system

Security Evolution
Fail-first Patch-later (cont.)

- Questions:
  - How many PC anti-virus programs can detect or repair handle malicious device drivers?
    - *None!*
  - What can an Active-X web download do to your PC?
    - *Anything!*

Security Evolution
Foundational Threats

- Software is as secure as its foundation
- If foundation can be successfully attacked
  - Then almost any form of system security is **useless**
- Foundational threats include
  - Bypass
  - Compromise
  - Tamper
  - Cascade
  - Covert Channel
  - Virus
  - Subversion

Security Evolution
Trusting the Foundation

- Alternative to the *fail-first patch-later* approach
- Use an approach designed to protect highly secure military systems
- **Mathematically verify** trusted components of
  - Operating system
  - Communications system
- Potential to fail security objectives is dramatically reduced

Relationship between System Modes and Assurance Levels

Common Criteria	*MILS/MLS Accreditation*
EAL3 (C-2 LoT)	*System High*
EAL4 (B-1 LoT)	*System High w/ Type Separation (SECRET NOFORN / SECRET NATO)*
EAL5 (B-2 LoT) *	*1 Level Separation (TS/S;S/C;C/U)*
EAL6 (B-3 LoT) *	*2 Level Separation (TS/S/C;S/C/U)*
EAL7 (A-1 LoT) *	*3 Level Separation (TS/S/C/U)*

* - RM: Reference Monitor

Assuming a Secure Development Environment

Security Evolution
Trusting the Foundation (cont.)

- Lower levels of Orange Book security enjoyed a wide success
  - In particular level C2
  - Wide commercial success
  - C2 certification is a common requirement for banking, insurance, and other security conscious commercial systems

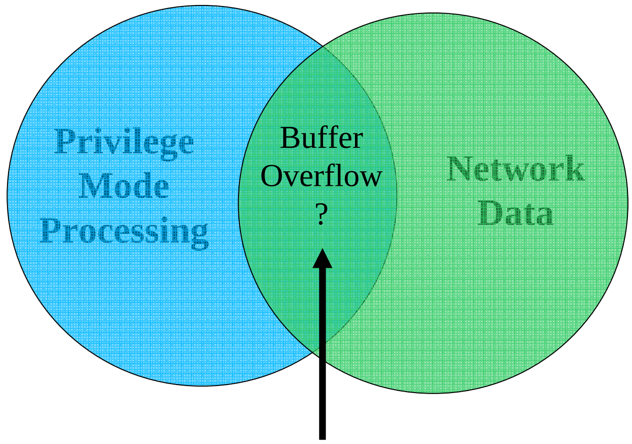
- Mathematical verification of general purpose software has a tarnished past
  - Littered with commercial and financial failures
  - Efforts were all focused on higher assurance levels in the U.S. DoD Orange Book

Security Evolution
Trusting the Foundation (cont.)

- Orange Book high assurance fell short in two areas
  1. Higher assurance levels (B3 and A1) required both
     - Mathematical verification of trusted system components
     - Those trusted systems components must contain significant security functionality
       - MAC, DAC, auditing, et al
       - Code size made mathematical verification almost impossible.

  2. Intersystem communication was not addressed
     - By core security architecture of the Orange Book
     - Trusted components (and device drivers) ran in privilege node for performance
       - Security critical application code also ran in privilege mode.
       - This was a nightmare to evaluate.
       - Such evaluations typically cost $100M.

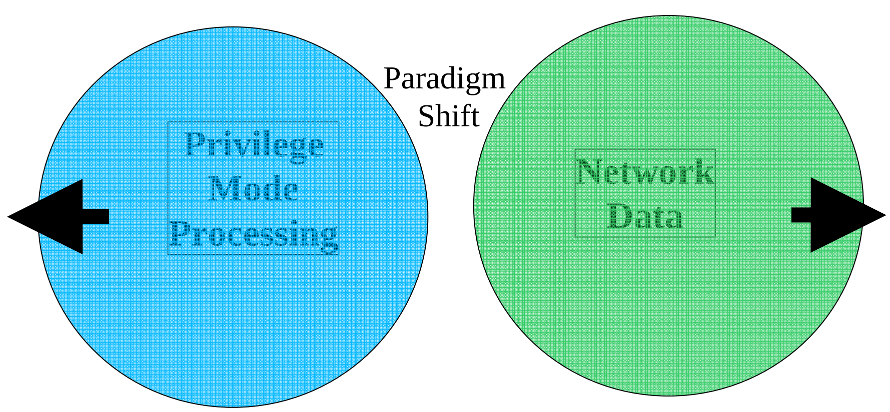
Foundational Threats



Wild Creatures of the Net, Worms, Virus, . . .

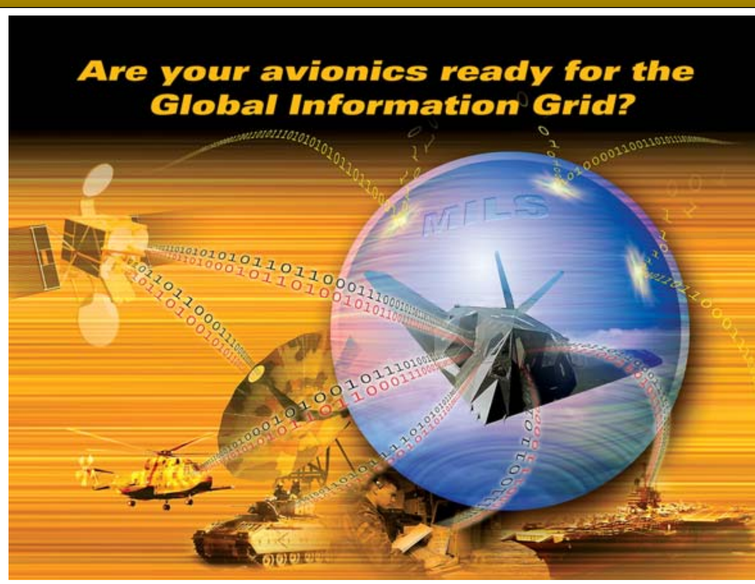
Foundational Threats

(That MILS Protects Against)



Under MILS Network Data and Privilege Mode Processing is Separated

GIG's Wide Scope Creates Wide Threats



MILS Overview

The Whole Point of MILS

Really simple:

- Dramatically **increase the scrutiny** of *security critical code*
- Dramatically **reduce the amount** of *security critical code*

Executive Overview
MILS Architecture Objectives

What does MILS do?

Enable the Application Layer Entities to
Enforce, Manage, and Control

***Application Level* Security Policies**

in such a manner that the Application Level Security Policies are

***N**on-bypassable*
***E**valuatable*
***A**lways-Invoked*
***T**amper-proof*

Reference Monitor Concept

MILS = Multiple Independent Levels of Security/Safety

MILS Architecture Objectives

How does MILS achieve its objectives?

Enforce an

Information Flow,
Data Isolation,
Periods Processing, and
Damage Limitation

Security Policy

between multiple address spaces:

First, in a **Microprocessor Centric Manner**, i.e., MILS RTOS Kernel,
Second, in a **Network Centric Manner**, i.e., MILS Middleware,

in such a manner that the layered Security Policies are

N E A T

Executive Overview
MILS Three Layer Architecture

Three distinct layers (John Rushby, PhD)

Partitioning Kernel
- Trusted to guarantee separation of time and space
  - Separate process spaces (partitions)
  - Time partitioning
- Secure transfer of control between partitions
- Really small: 4K lines of code

1. Middleware
   - Secure application component creation
   - Secure end-to-end inter-object message flow
   - Most of the traditional operating system functionality
     - Device drivers, file systems, etc.
   - Partitioning Communications System
     - Extends the policies of Partitioning Kernel to communication
     - Facilitates traditional middleware
       - Real-time CORBA, DDS, web services, etc.

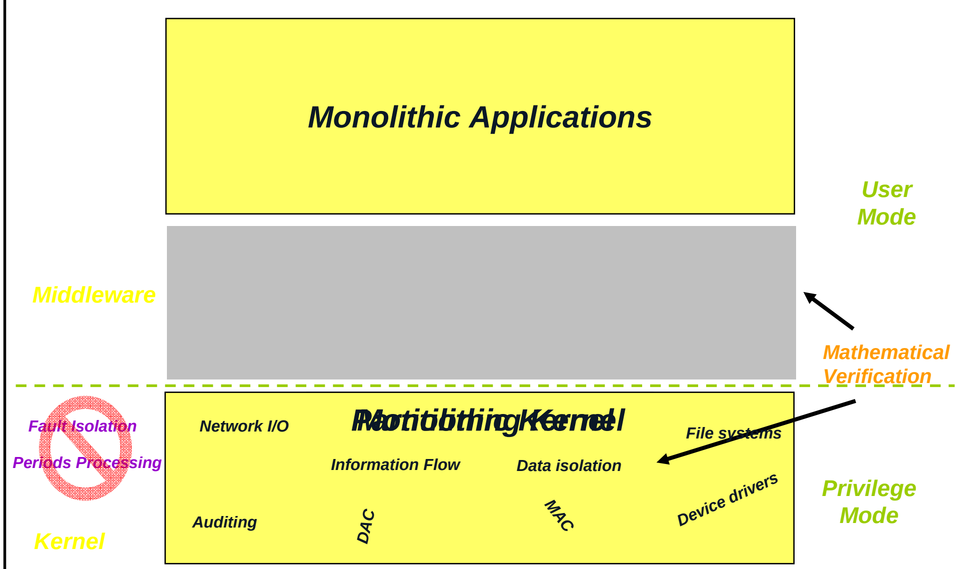
2. Applications
   - *Can* enforce application-specific security functions
   - e.g., firewalls, crypto services, guards

Partitioning Kernel

- Where should PK reside?
  - To be tamper-proof
    - Must be in a separate address space from any untrusted application code
  - To be non-bypassable
    - Must be part of every input or output service request issued by an application
    - The PK must be the sole proprietor of privileged mode processing

- Why is putting security functions in kernel bad?
  - Security functions are often application-specific
  - Any code co-resident with security function could interfere with kernel's security enforcement
  - Entire kernel must be analyzed for weaknesses and malicious code

Orange Book vs. MILS Architecture



Layer Responsibilities

Partitioning Kernel Functionality

- Time and Space Partitioning
- Data Isolation
- Inter-partition Communication
- Periods Processing
- Minimum Interrupt Servicing
- Semaphores
- Timers
- Instrumentation

And nothing else!

MILS Middleware Functionality

- **RTOS Services**
  - Device Drivers
  - CORBA
  - File System
  - …
- **Partitioned Communication System**
  - Inter-processor communication

MILS

- MILS makes **mathematical verification** possible
  - Of the core systems and communications software
  - By reducing the security functionality
  - To four key security policies

1. ***Information Flow***
2. ***Data Isolation***
3. ***Periods Processing***
4. ***Damage Limitation***

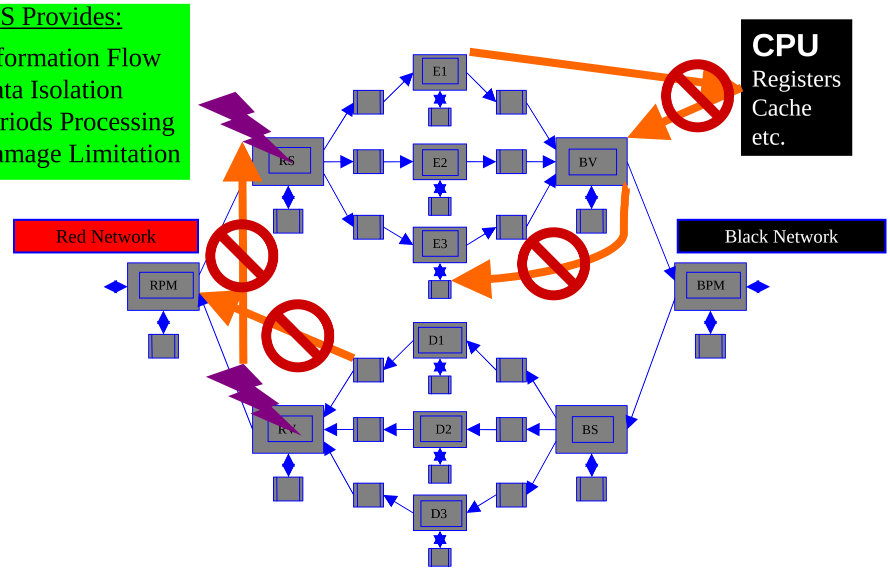
MILS Security Policies

- Information Flow
  - Information originates only from authorized sources
  - And is delivered only to intended recipients
  - Source of Information is authenticated to recipient
- Data Isolation
  - Information in a partition is accessible only by that partition
  - Private data remains private
  - May require encryption for end-to-end protection
- Periods Processing
  - The microprocessor itself will not leak information from one partition to another as the processor switches from partition to partition
- Damage Limitation
  - A failure in one partition will not cascade to another partition
  - Failures will be detected, contained, & recovered from locally

Executive Overview
MILS Security Policy Example

MILS Provides:

- Information Flow
- Data Isolation
- Periods Processing
- Damage Limitation



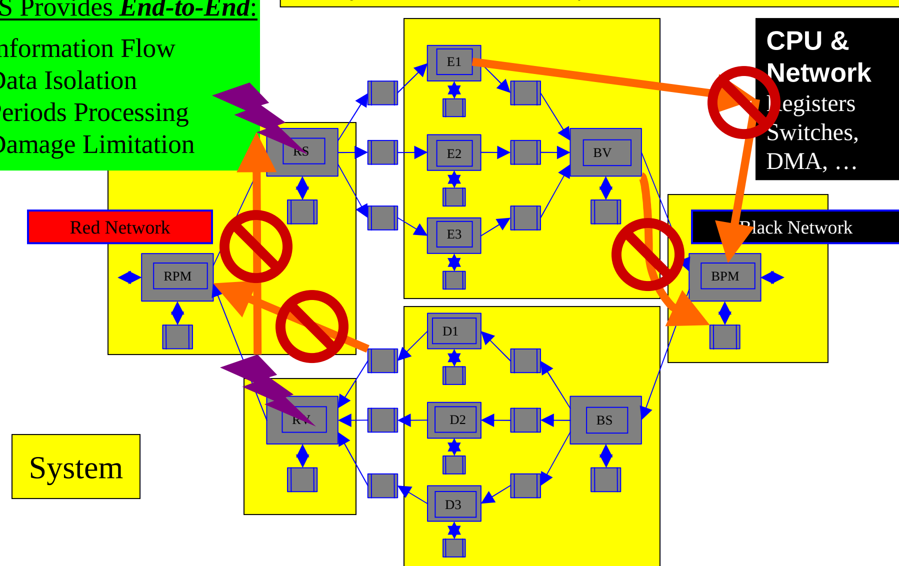
6/28/2004 High-Assurance, Real-Time MILS Architecture Vanfleet, Beckwith - 29

Executive Overview
MILS Network Security Policy Example

Policy Enforcement Independent of Node Boundaries

PCS Provides ***End-to-End***:

- Information Flow
- Data Isolation
- Periods Processing
- Damage Limitation

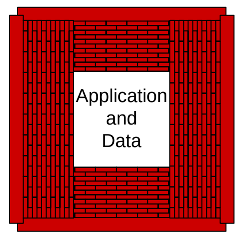


6/28/2004 High-Assurance, Real-Time MILS Architecture Vanfleet, Beckwith - 30

Executive Overview
MILS: Like a JVM for All Applications

- The Java Virtual Machine contains
  - Internet Java
    - To a confined set of operations
    - In each JVM
  - Result:
    - Potential for damage is limited
- The MILS architecture contains
  - All executable code
    - To a confined set of operations
    - In each ***partition***
  - Result:
    - Potential for damage is bounded
    - + Information flow is bounded

MILS Partition



Required Characteristics of
MILS Reference Monitors

- Partitioning Kernel & trusted Middleware must be:

N E A T

Non-bypassable
- Security functions cannot be circumvented

Evaluatable
- Security functions are small enough and simple enough for mathematically verification

Always Invoked
- Security functions are invoked each and every time

Tamperproof
- Subversive code cannot alter the security functions
  - By exhausting resources, over-running buffers, or other ways of making the security software fail

MILS *NEAT* Requirements

- With most operating system architectures it is very difficult to prove that *NEAT* requirements are met
- MILS architecture makes this much easier to prove *NEAT*-ness
- MILS is based on a micro-kernel that:
  - partitions the computer into separate address spaces and scheduling intervals
  - guarantees isolation of the partitions
  - supports controlled communications among partitions

Safety and Security

- **Safety and Security**
  - Military market alone not large enough to justify investing in expensive RTOS evaluations
  - Commercial avionics market has attracted their investment dollars
  - Especially given the imminent adoption of Global Air Traffic Management (GATM) rules
- **ARINC-653**
  - Written specifically for avionics computing (especially flight safety)
  - Specifies an RTOS design like the partitioning kernel architecture
  - Same design goal:
    - Allow two or more programs to share a computer
    - Guarantee that they cannot interfere with each other
  - Both memory and processing time are statically allocated to partitions using configuration tables
    - Static network of communication channels between partitions
    - All input and output for a partition go through these channels
      - Except a few kernel services (e.g. reading the real time clock)
  - ARINC-653 specifies generic framework for enforcing an application-specific information flow control security policy
- Partitioning kernel guarantees information can flow from one partition to another only in the ways specified in the static configuration tables

Layer Responsibilities

Partitioning Kernel Functionality

- Time and Space Partitioning
- Data Isolation
- Inter-partition Communication
- Periods Processing
- Minimum Interrupt Servicing
- Semaphores
- Timers
- Instrumentation

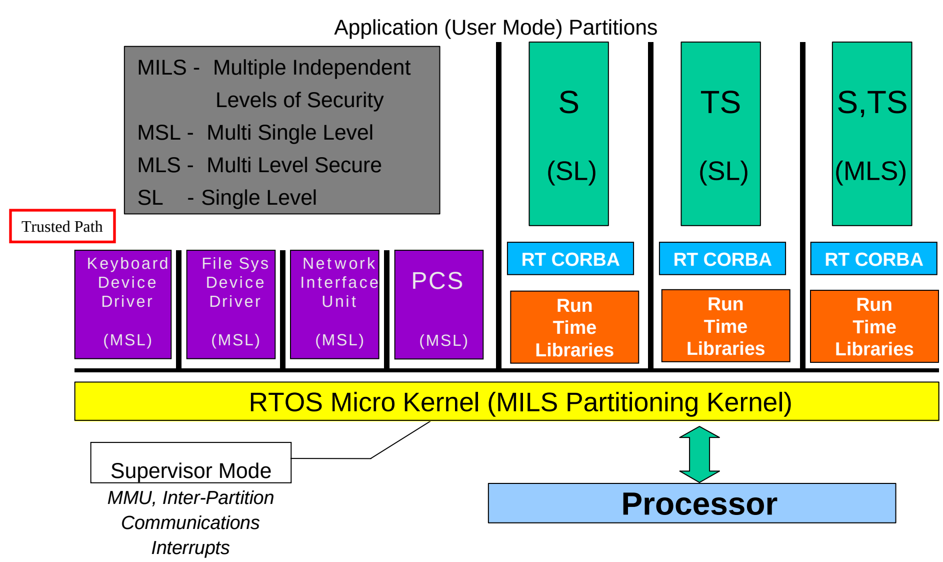
And nothing else!

MILS Middleware Functionality

- **RTOS Services**
  - Device Drivers
  - CORBA
  - File System
  - …
- **Partitioned Communication System**
  - Inter-processor communication

6/28/2004 High-Assurance, Real-Time MILS Architecture Vanfleet, Beckwith - 35

Executive Overview
MILS Architecture – High Assurance



6/28/2004 High-Assurance, Real-Time MILS Architecture Vanfleet, Beckwith - 36

Partitioning Kernel: Just a Start …

- Partitioning Kernel provides
  - Secure foundation for secure middleware
- Secure Middleware provides
  - Most of traditional O/S capabilities
    - File system
    - Device drivers (*not* in the kernel, not special privileges)
    - Etc.
  - Secure intersystem communication (PCS)
  - Secure foundation for building secure applications
- Secure Applications can
  - Be built!
  - Be trusted to enforce application-level security policies!!!

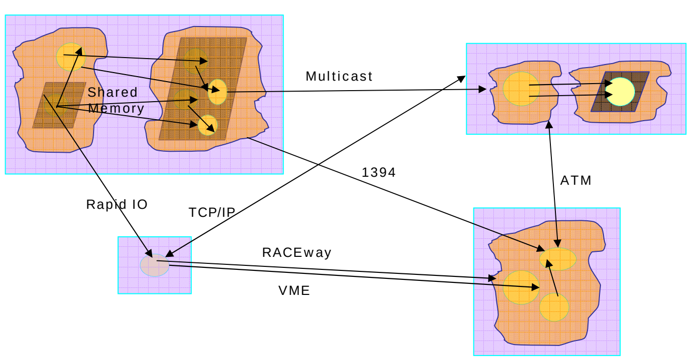
Distributed Security

Distributed Security Requirements

- Rely upon partitioning kernel to enforce middleware security policies on a given node
  - Information Flow
  - Data Isolation
  - Periods Processing
  - Damage Limitation
- Application-specific security requirements
  - must not creep down into the middleware (or kernel)
  - ensure the system remains supportable and evaluatable
- Optimal inter-partition communication
  - Minimizing added latency (first byte)
  - Minimizing bandwidth reduction (per byte)
- Fault tolerance
  - Security infrastructure must have no single point of failure
  - Security infrastructure must support fault tolerant applications

Distributed Object Communication

- Partition Local – same address space, same machine
- Machine Local – different address space, same machine
- Remote – different address space, on a different machine



Partitioned Communication System

Partitioned Communication System

- Partitioned Communication System
  - Part of MILS Middleware
  - Responsible for all communication between MILS nodes
- Purpose
  - Extend MILS partitioning kernel protection to multiple nodes
- Similar philosophy to MILS Partitioning Kernel
  - Minimalist: only what is needed to enforce end-to-end versions of policies
    - *End-to-end* Information Flow
    - *End-to-end* Data Isolation
    - *End-to-end* Periods Processing
    - *End-to-end* Damage Limitation
  - Designed for EAL level 7 evaluation

PCS Objective

- Just like MILS Partitioning Kernel:
  - Enable the Application Layer Entities to
    - Enforce, Manage, and Control
  - Application Level
    - Security Policies
  - in such a manner that the Application Level Security Policies are
    - **N**on-Bypassable,
    - **E**valuatable,
    - **A**lways-Invoked, and
    - **T**amper-proof.
  - An architecture that allows the Security Kernel and PCS to share the RESPONSIBILITY of Security with the Application.
- Extended:
  - To all inter-partition communication within a group of MILS nodes (*enclave*)

PCS Requirements

- Strong Identity
  - Nodes within enclave
- Separation of Levels/Communities of Interest
  - Need cryptographic separation
- Secure Configuration of all Nodes in Enclave
  - Federated information
  - Distributed (compared) vs. Centralized (signed)
- Secure Clock Synchronization
- Secure Loading: signed partition images
- Elimination of Covert Channels
  - Bandwidth provisioning & partitioning
  - Network resources: bandwidth, hardware resources, buffers

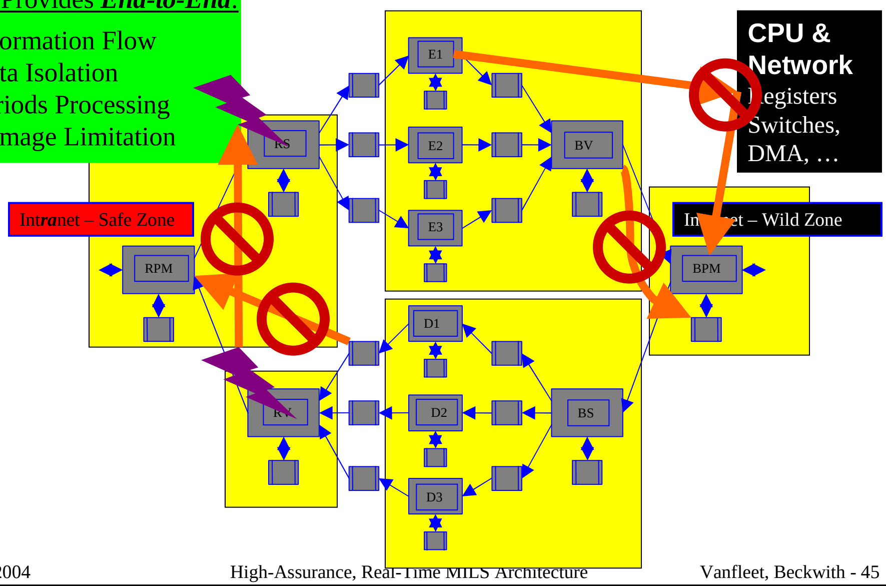
MILS Security Policy Example: Distributed Internet Firewall

Policy Enforcement Independent of Node Boundaries

PCS Provides ***End-to-End***:

- Information Flow
- Data Isolation
- Periods Processing
- Damage Limitation

CPU & Network
Registers
Switches,
DMA, …



Real-time MILS CORBA

Real-Time MILS CORBA

- Real-time CORBA can take advantage of PCS capabilities
  - Real-time CORBA + PCS = Real-time MILS CORBA
  - Additional application-level security policies are enforceable because of MILS PK and PCS foundation
- Real-time MILS CORBA represents a single enabling application infrastructure

Real-time MILS CORBA (cont.)

- Can address key cross-cutting system requirements
- MILS-based distributed security
  - High-assurance
  - High-integrity (safety critical systems)
- Real-time
  - Fixed priority
  - Dynamic scheduling
- Distributed object communications
  - Predictable
  - Low latency
  - High bandwidth

Real-time CORBA + MILS Synergy

- Synthesis yields an unexpected benefit
  - Flexibility of Real-time CORBA allows realization of MILS protection
  - MILS is all about **location awareness**
    - Well designed MILS system separates functions into separate partitions
    - Takes advantage of the MILS partitioning protection
  - Real-time CORBA is all about **location transparency**
    - The application code of a properly designed distributed system built with Real-time CORBA will not be aware of the location of the different parts of the system.
    - CORBA flexibility allows performance optimizations by rearranging what partitions each system object executes in.
    - System layout can be corrected late in the development cycle
  - Combination of MILS and Real-time CORBA allows
    - Rearrange system functions to take advantage of protection

Optional Topics

- Foundational Threats
- MILS Architecture Application
- MILS Example
- MILS Intelligent I/O Devices
- MILS Middleware Realization

Foundational Threats

Security Evolution
Foundational Threats

- Software is as secure as its foundation
- If foundation can be successfully attacked
  – Then almost any form of system security is **useless**
- Foundational threats include
  – Bypass
  – Compromise
  – Tamper
  – Cascade
  – Covert Channel
  – Virus
  – Subversion

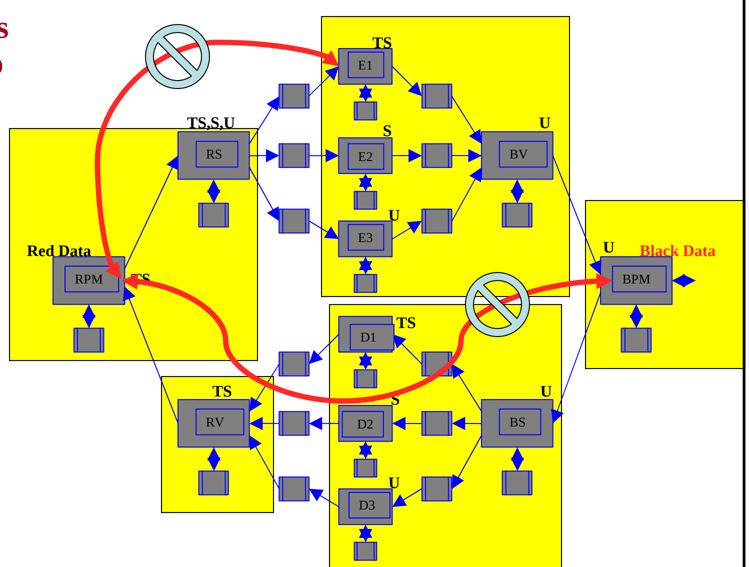
Security Evolution
Foundational Threats (cont.)

- Bypass
  - Malicious/flawed software circumvents the system's protection
  - Safety/security critical protocol not invoked
  - If critical software can be bypassed there is no assurance that application programs using critical services are safe
  - Bypass can occur at multiple layers
    - O/S
    - Communications
    - Application

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ **Bypass**
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion



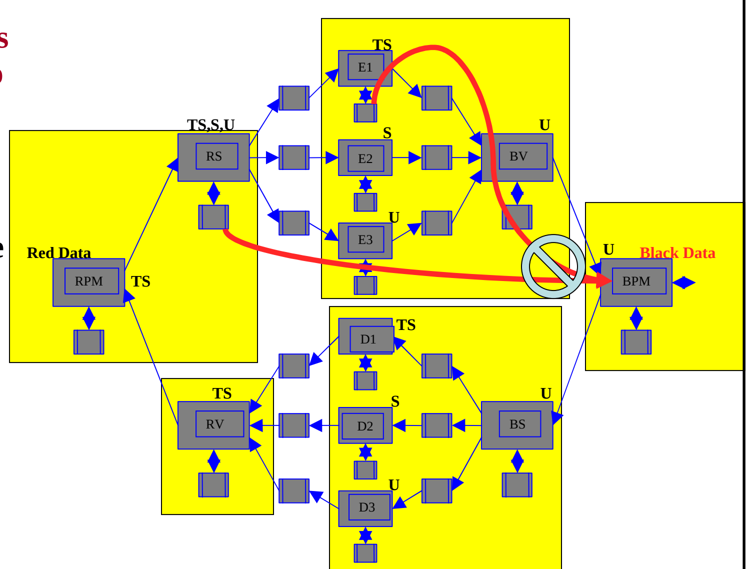
Security Evolution
Foundational Threats (cont.)

- Compromise
  - Malicious/flawed software can read private data of other programs
  - If invasive software can monitor the data of other applications then entire system security is suspect
    - Like spyware so common in today's Internet environment

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ Bypass
- ✓ **Compromise**
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion



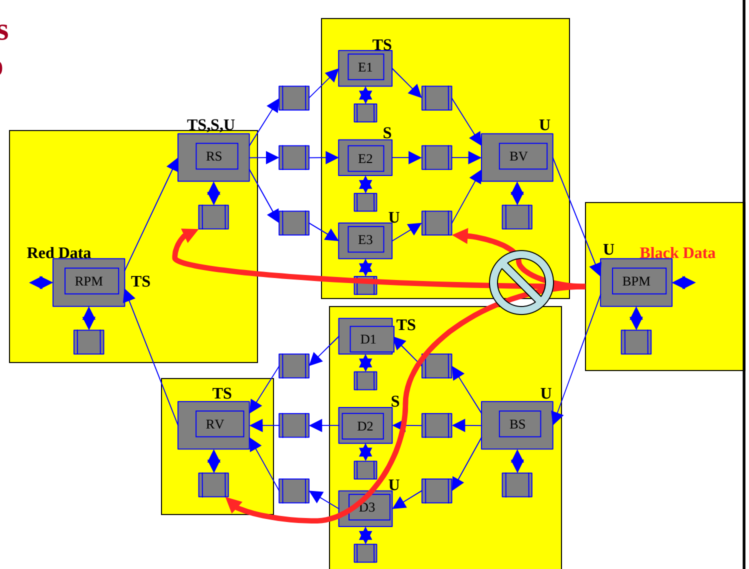
Security Evolution
Foundational Threats (cont.)

- Tamper
  - Malicious/flawed software modifies the sensitive data of other programs
  - If tamper is possible then no application is safe from viruses, worms, hackers, spyware, etc.

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ Bypass
- ✓ Compromise
- ✓ **Tamper**
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion



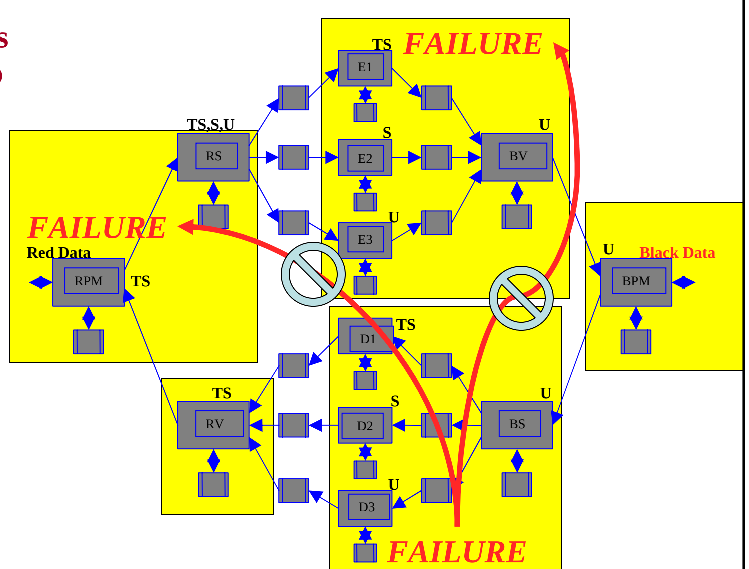
Security Evolution
Foundational Threats (cont.)

- Cascade
  - Malicious/flawed software allows failures to cascade from one system component to another
  - If a failure of one application can cause failure of another application it may be possible for much larger system failure
    - A notable example of unintentional failure cascade is a Navy cook who entered zero into a window that asked for a non-zero number
    - The application divided by zero
    - This caused other applications failed
    - Eventually the O/S failed
    - The hard drive got screwed up
    - The system would not reboot
    - The ship was towed to shore

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ **Cascade**
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion



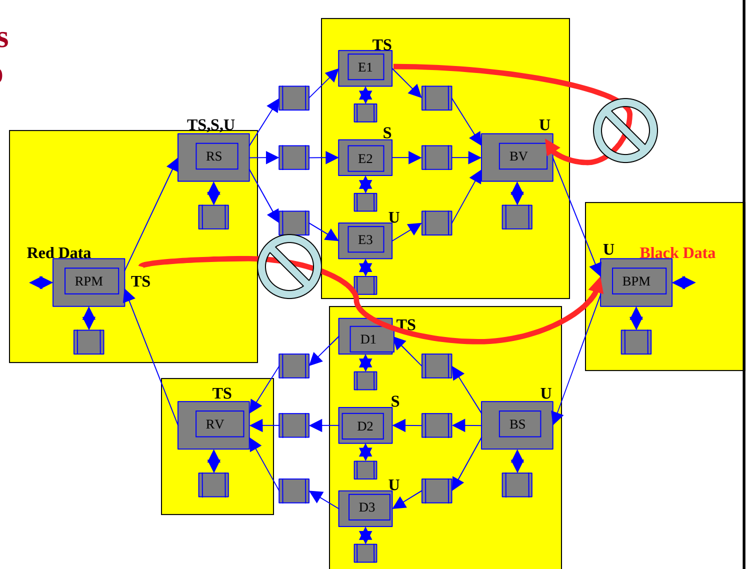
Security Evolution
Foundational Threats (cont.)

- Covert Channel
  - Malicious/flawed software that can leak information through a communication channel that is a side effect
  - Example:
    - By detecting the presence or absence of a message an observer can derive information as to the activity of the communicating parties
    - Can use morse code to signal information right through a typical hardware VPN
  - If there are covert timing channels available a malicious communicating party can leak any information to the observing party by creating intentional timing messages in an arranged pattern

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ **Covert Channel**
- ✓ Virus
- ✓ Subversion



Covert Channel File System

- Serious problem for software that implements any kind of
- information flow policy between applications
  - Covert channels are only exploitable in the presence of Trojan horses
  - Side channel analysis of traditional smart cards (with all applications in a single protection domain – ie: mutually trusting) is NOT covert channel analysis
- Covert channel analysis is required at EAL5 by Common Criteria
  - If an information flow policy exists
- Covert channel analysis of software requires an intimate knowledge of the hardware implementation – cannot be transferred between two different models of same processor

Disk Arm Covert Channel

- **IBM 370 discovered covert channel in the software implementation of the elevator algorithm in disk drivers**
  - State variable of the direction of disk arm motion is exploitable

- **Solved problem by eliminating elevator algorithm from disk driver**
- **More modern disk controllers implement the elevator algorithm in hardware!**
  - Not mentioned in hardware interface specification
  - Hardware designer and evaluator may have no idea that there could be a problem from this, yet O/S designer needs to know if this is done and whether it can be turned off
- **DEC's VMM Security Kernel for the VAX had to deal with this problem in its A1 Orange Book evaluation**
  - Had to have in-depth information about the hardware implementation
  - Elevator algorithm could NOT be turned off
  - Required major re-design of VMM disk drivers to batch all disk requests to conceal the effects of the elevator algorithm – patented result
  - ETR-lite on disk controller would likely have not revealed the problem!

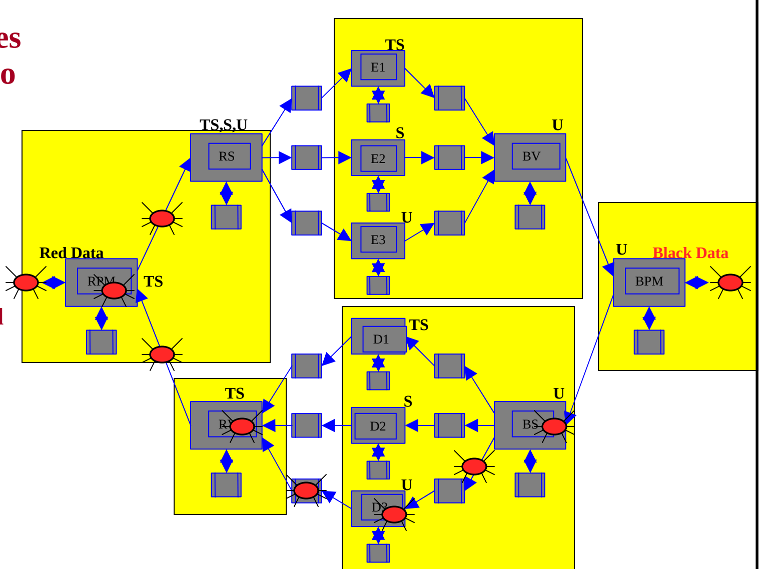
Security Evolution
Foundational Threats (cont.)

- Virus
  - Malicious/flawed software that runs at privileged levels so that it can infect all parts of the system and other systems.
  - What is necessary is an architecture that enforces and manages the concept of **least privilege**.
    - Then when a compromise occurs
      - Its damage is local
      - Its damage can be detected
      - Its damage recovered from
    - A big part of countering the computer virus problem is kicking device drivers and applications out of privilege mode.

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ **Virus**
- ✓ Subversion



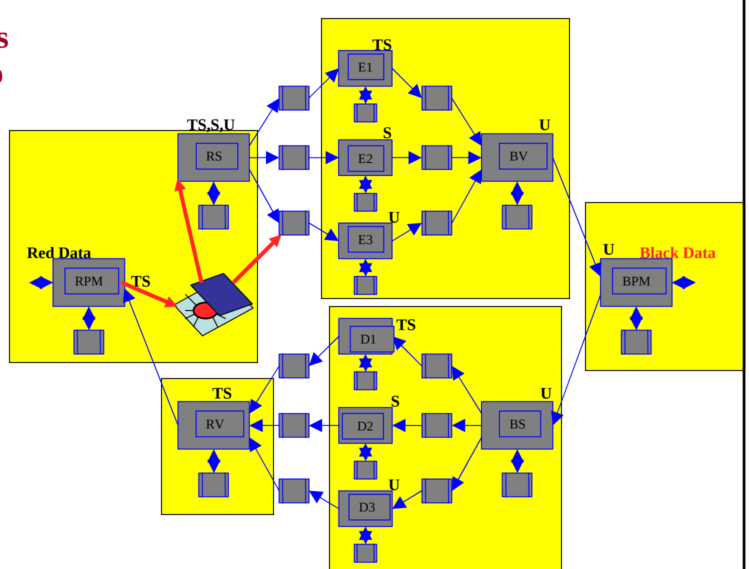
Security Evolution
Foundational Threats (cont.)

- Subversion
  - Malicious/flawed software loaded by user who thinks software is legitimate
  - All code needs to be signed or it does not even load
  - The source of all software must be traceable to the original author
  - Software authors should follow good software engineering practices
  - Preventing subversion is everyone's responsibility.

Foundational Threats
(That MILS Protects Against)

MILS provides mechanisms to counter Foundational Threats

- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ **Subversion**

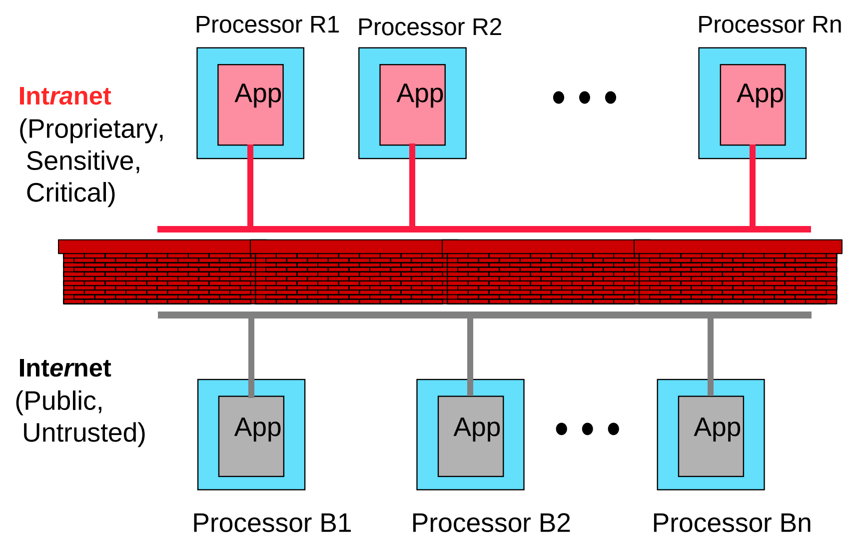


MILS Architecture Application

MILS Replaces Physical Separation

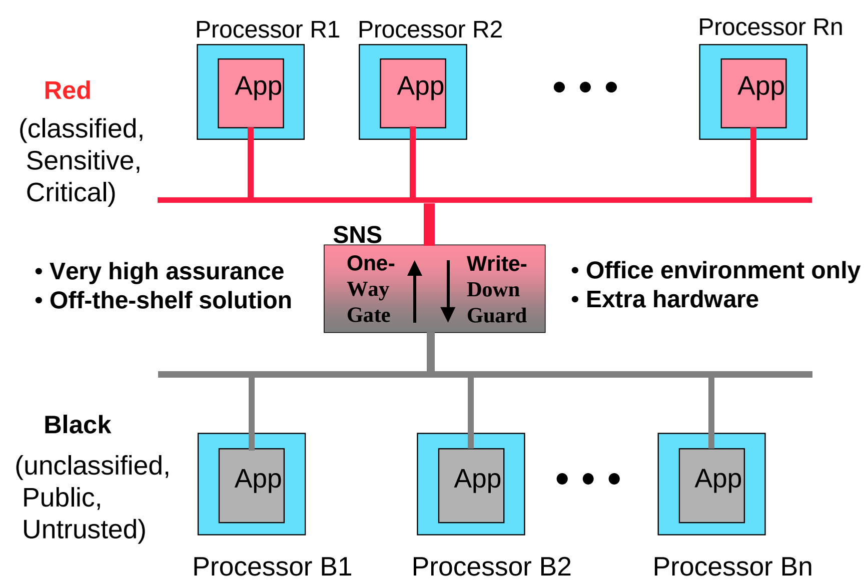
- MILS architecture allows computer security measures to achieve the assurance levels as "physically isolated" systems
  - All O/S code not necessary for performing Partitioning Kernel functions moved out of privileged mode
  - O/S service code moved to middleware layer
    - e.g. device drivers, file system, POSIX
  - Prevents software and network attacks from elevating a partition privilege to an unauthorized level

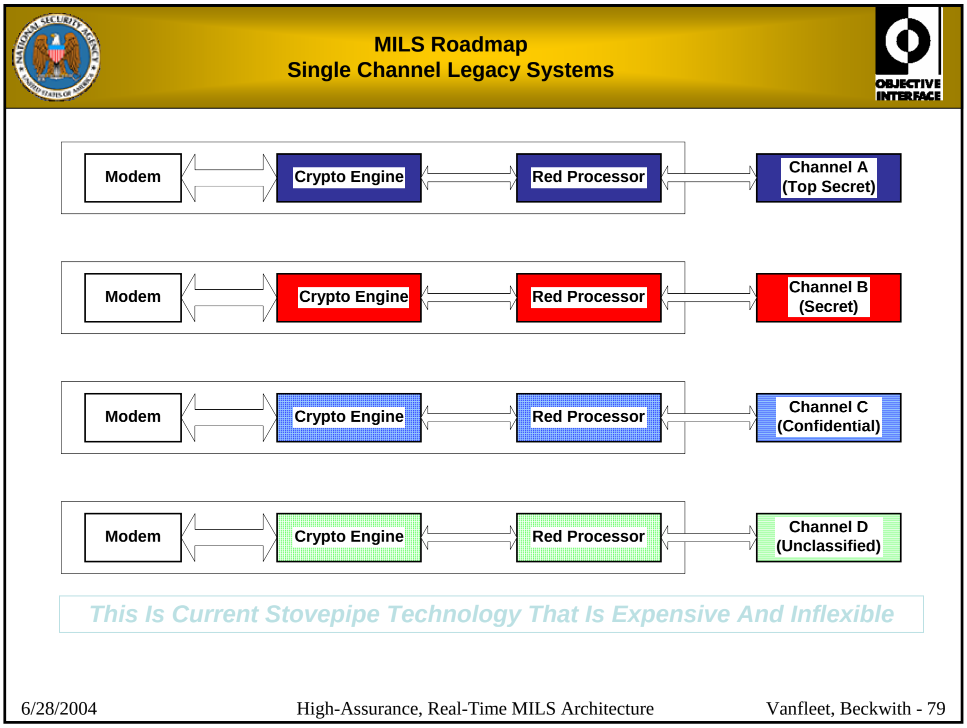
Best Security/Safety is Physical



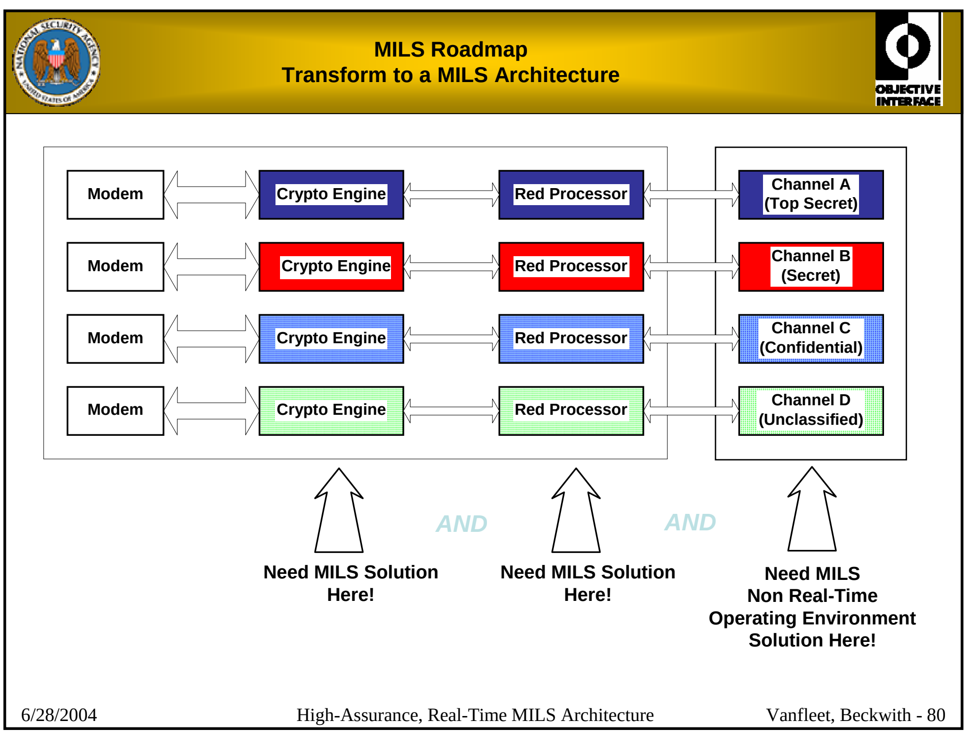
Secure Network Server

(Good, Expensive, Physical Solutions Exist)



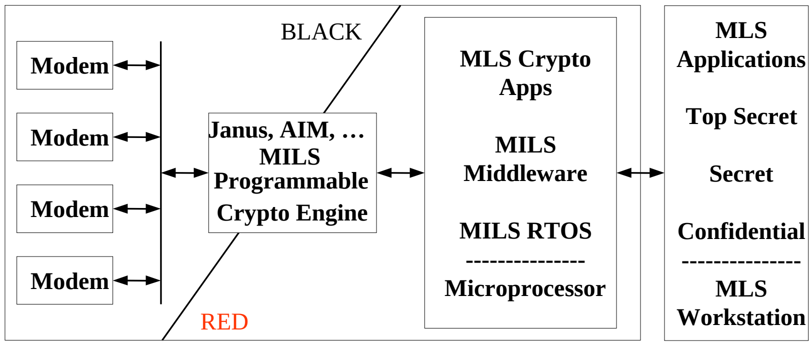


MILS Roadmap
Single Channel Legacy Systems
OBJECTIVE INTERFACE
Modem
Crypto Engine
Red Processor
Channel A (Top Secret)
Modem
Crypto Engine
Red Processor
Channel B (Secret)
Modem
Crypto Engine
Red Processor
Channel C (Confidential)
Modem
Crypto Engine
Red Processor
Channel D (Unclassified)
This Is Current Stovepipe Technology That Is Expensive And Inflexible
6/28/2004
High-Assurance, Real-Time MILS Architecture
Vanfleet, Beckwith - 79



MILS Roadmap
Transform to a MILS Architecture
OBJECTIVE INTERFACE
Modem
Crypto Engine
Red Processor
Channel A (Top Secret)
Modem
Crypto Engine
Red Processor
Channel B (Secret)
Modem
Crypto Engine
Red Processor
Channel C (Confidential)
Modem
Crypto Engine
Red Processor
Channel D (Unclassified)
Need MILS Solution Here!
AND
Need MILS Solution Here!
AND
Need MILS Non Real-Time Operating Environment Solution Here!
6/28/2004
High-Assurance, Real-Time MILS Architecture
Vanfleet, Beckwith - 80

MILS Roadmap
MILS Architecture



- Example MILS Applications
  - Communications Systems (JTRS, TTNT, TCS SATCOM)
    - Multi-channel, low power, low cost, flexible, open systems architectures
  - Precision Guidance/Navigation (GPS/SASSM, MUE)
    - Highly integrated, low power, low cost (system on a chip)
  - System & Platform Integration (FCS, Flight2, E6)
    - Integrated data management/fusion w/ information assurance

Introduction – MLS/MSLS

Multi-Level Secure/Safe (MLS): Processes data of differing classifications/sensitivities securely/safely
- down graders
- data fusion
- guards
- firewalls
- data bases

Multi-Single Level Secure/Safe (MSLS): Separates data of differing classifications/sensitivities securely/safely simultaneously
- communications platforms
- infrastructures

MILS Can Handle MLS

- A PK is ignorant of traditional Multi-Level Security (MLS)
  - Requirement for military and intelligence systems
- However, MILS is quite capable of supporting MLS systems
- MILS can be used to construct MLS systems because of
  - Strong separation guarantees
  - Certification process

Application Layer

- MILS empowers the application layer to protect itself
- Application layer is responsible for enforcing its own security policies
- This layer provides for application-specific security policies
- A partition that processes data from more than one secure application realm must be considered a privileged partition

MILS Example

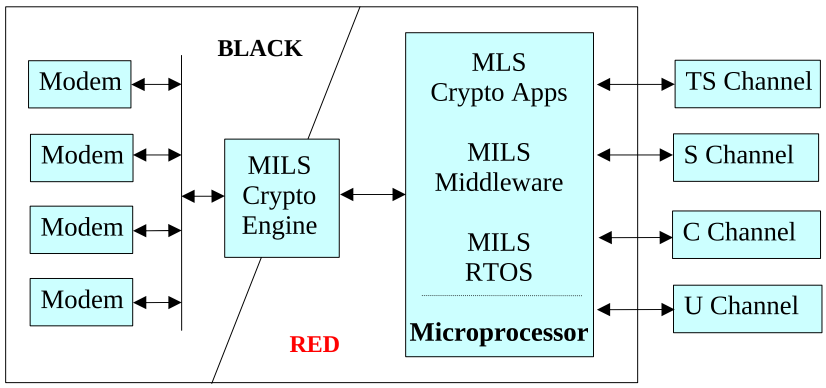
Example – JTRS
Joint Tactical Radio System

- Family of software programmable radios
- Design around Software Communications Architecture
- JTRS provides reliable multichannel voice, data, imagery, and video communications
- Eliminates communications problems of "stovepipe" legacy systems
- JTRS is:
  - Modular, enabling additional capabilities and features to be added to JTR sets
  - Scaleable, enabling additional capacity (bandwidth and channels) to be added to JTR sets
  - Backwards-compatible, communicates with legacy radios
  - Allowing dynamic intra-network and inter-network routing for data transport that is transparent to the radio operator

JTRS

Image Copyright © Rockwell Collins, Inc. All rights reserved

MILS Roadmap
MILS Crypto Engine & Emb OE



Software MLS Component

- JTRS is an example of a hardware / software middleware component
- The concept can be mapped to a similar software component
- Ex: Trusted network interface unit
  - Encrypts messages based on security label
  - Decrypts and labels messages appropriately

The MILS Approach to Designing an MLS Component

Classified network (Red), labeled messages

Unclassified Network (Black)

Image Copyright © Rockwell Collins, Inc. All rights reserved

Designing an MLS Component

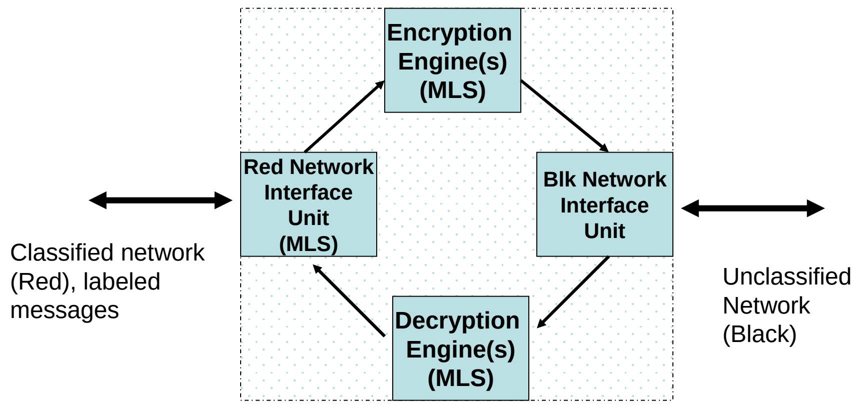
Classified network (Red), labeled messages

MLS Middleware Component

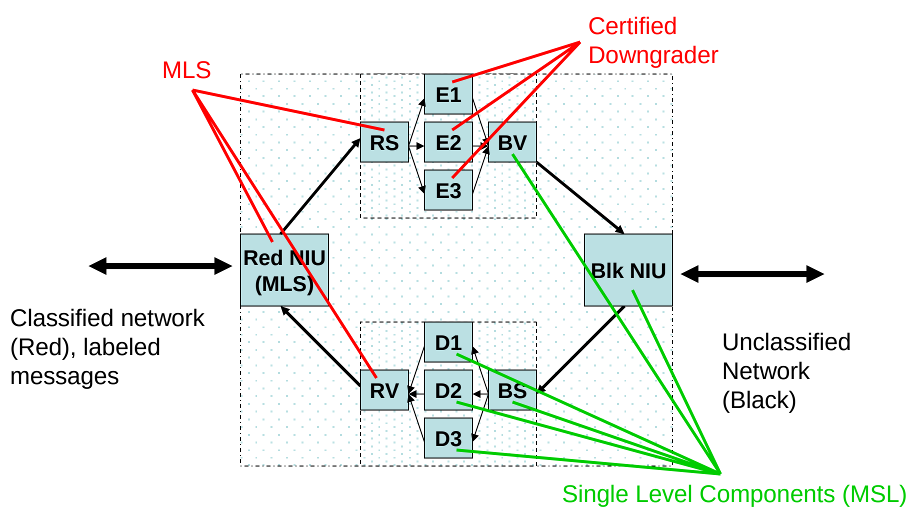
Ex: Cryptographic downgrader, such as JTRS or trusted network interface unit

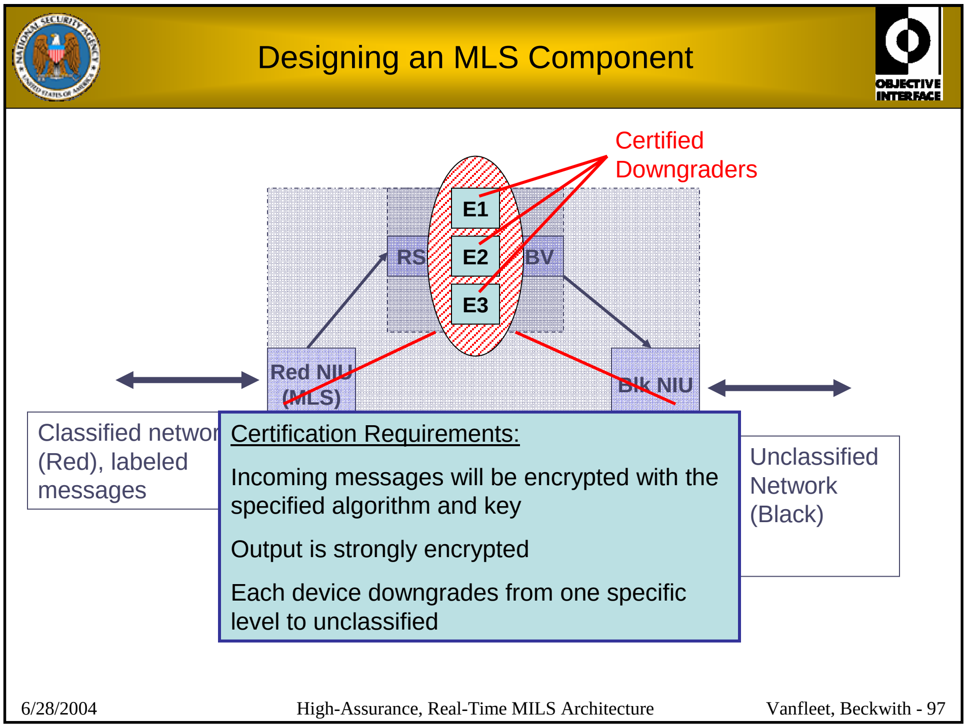
Unclassified Network (Black)

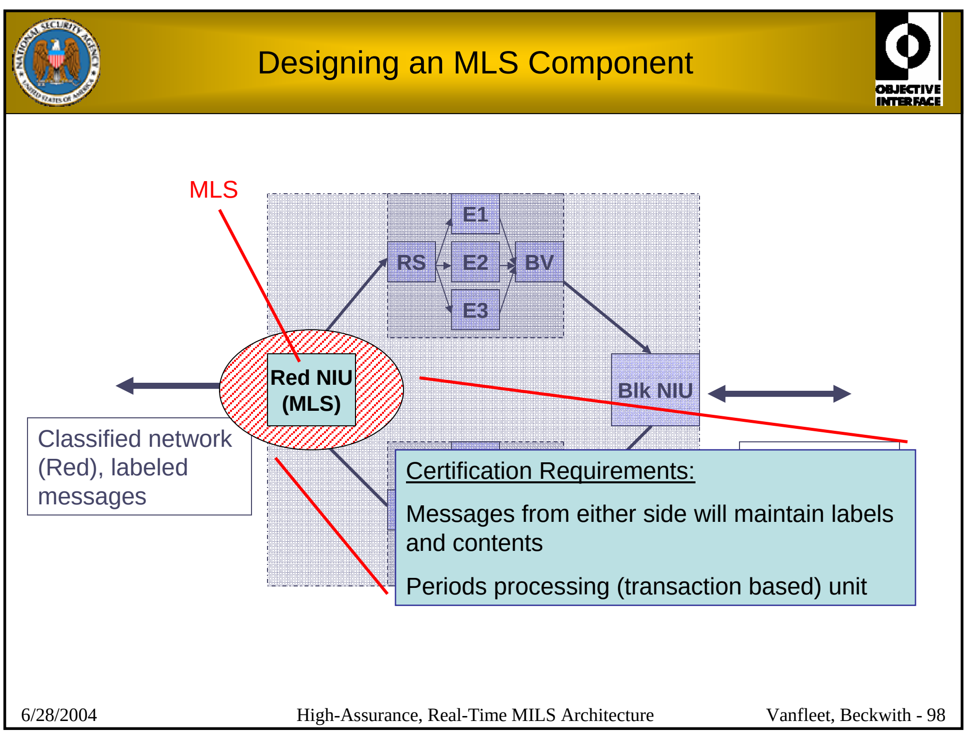
Designing an MLS Component



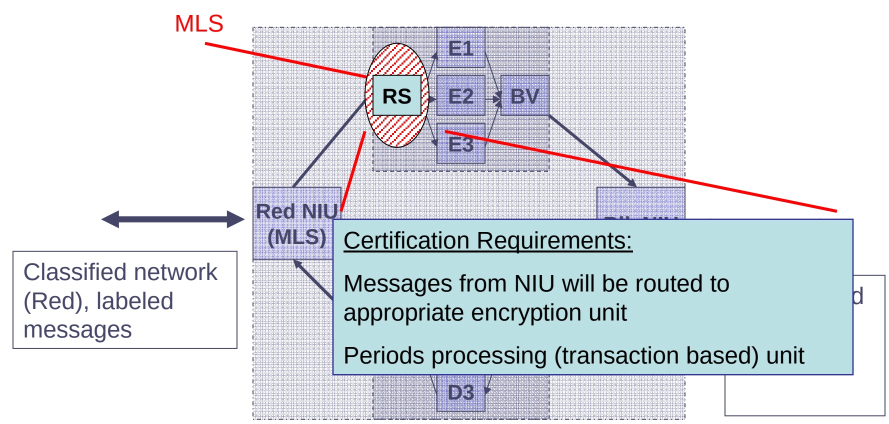
Designing an MLS Component




Designing an MLS Component
Certified
Downgraders
E1
E2
E3
RS
BV
Red NIU
(MLS)
Blk NIU
Classified network
(Red), labeled
messages
Unclassified
Network
(Black)
Certification Requirements:
Incoming messages will be encrypted with the specified algorithm and key
Output is strongly encrypted
Each device downgrades from one specific level to unclassified
6/28/2004
High-Assurance, Real-Time MILS Architecture
Vanfleet, Beckwith - 97

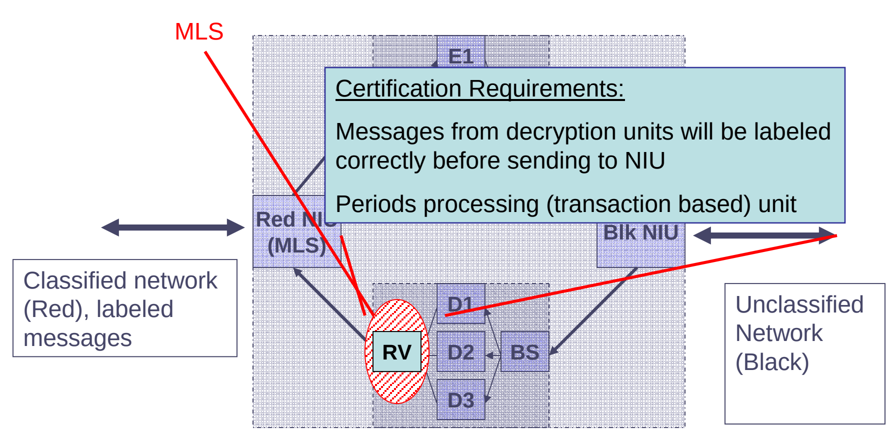

Designing an MLS Component
MLS
E1
E2
E3
RS
BV
Red NIU
(MLS)
Blk NIU
Classified network
(Red), labeled
messages
Certification Requirements:
Messages from either side will maintain labels and contents
Periods processing (transaction based) unit
6/28/2004
High-Assurance, Real-Time MILS Architecture
Vanfleet, Beckwith - 98

Designing an MLS Component



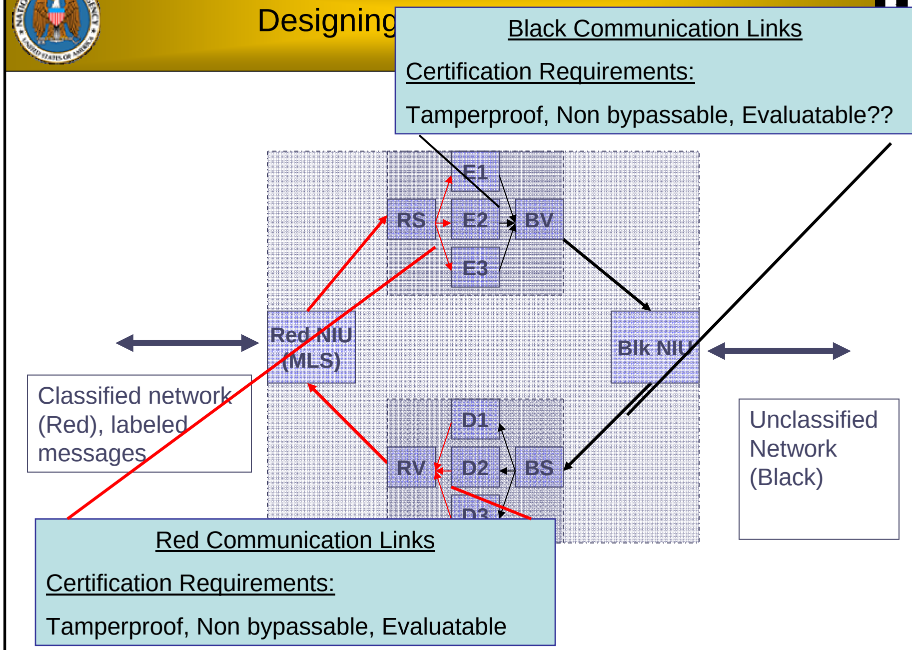
6/28/2004 High-Assurance, Real-Time MILS Architecture Vanfleet, Beckwith - 99

Designing an MLS Component



6/28/2004 High-Assurance, Real-Time MILS Architecture Vanfleet, Beckwith - 100

Designing



The MILS Architecture Approach

- Describe the system in terms of communicating components
  - Designate the clearance of each component and label as MLS or MSL
  - Determine the flow between components with respect to policy
  - Install “boundary firewalls” that manage information up-flow and down-flow
    - these are MLS components

The MILS Architecture Approach

- For each MLS device, determine its type
  - Downgrader – will take data from one security level and send data at a lower level
  - Transaction processor – will process data one message at a time; stateless, may filter data or perform operation on single message
  - Collator – will combine data from many inputs
- Verification of each device may involve additional MILS componentization

Implementation

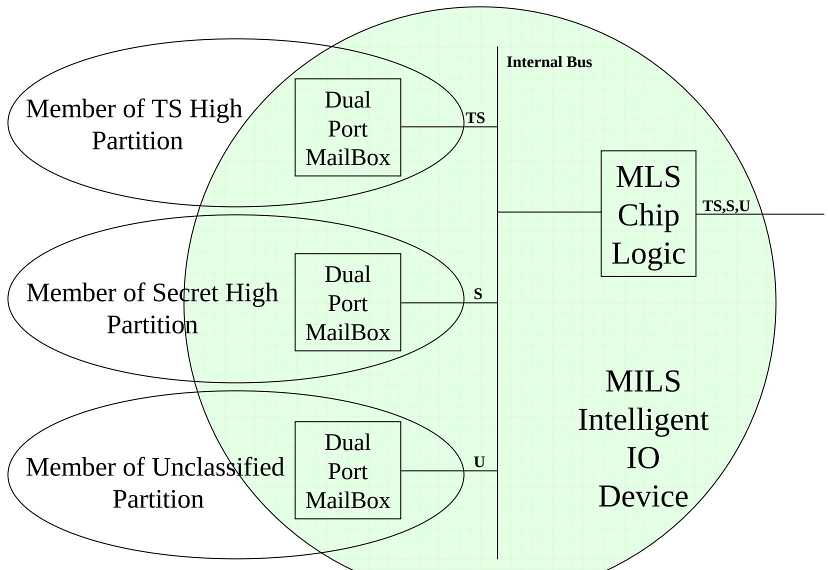
- Hierarchical Approach
  - Lowest level is separation kernel – enforces isolation, information flow, periods process, damage limitation on a single processor
  - Next level is middleware, to coordinate end-to-end separation
    - Need to create “trusted” components.
      - Verification of the components utilizes architectural support of lower layer
  - Next Level is application specific

MILS Intelligent I/O Devices

MILS Intelligent I/O Devices

- I/O Device Interface via User Mode Partitions ONLY
- I/O Device Supports Multiple User Mode Partitions
  - Each User Mode Partition has own Clearance
- I/O Device manages Clearance of User Mode Partitions
  - User Mode Partitions not trusted to report Clearance
- I/O Device Imports / Exports Security Label
  - Will not allow Write Down nor Read Up
  - Network Interface Unit (NIU) and Rapid-IO Examples

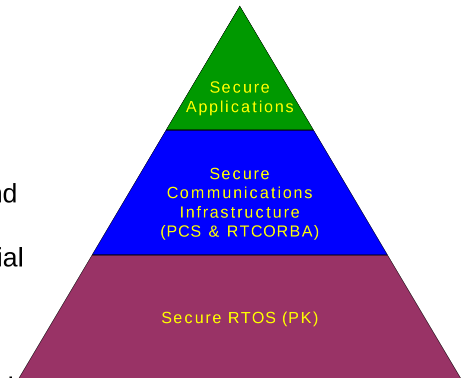
MILS Intelligent IO Device



MILS Middleware Realization

Business Dependencies

- Success depends on
  - Strong business and technical commitment by RTOS vendors
  - Customer need
    - There's a difference between *wanting* security and *buying* security
    - Performance, size, and predictability are *key*
    - Ease of use is essential
  - Logistics of standards groups
    - Reconciliation of business and technical perspectives



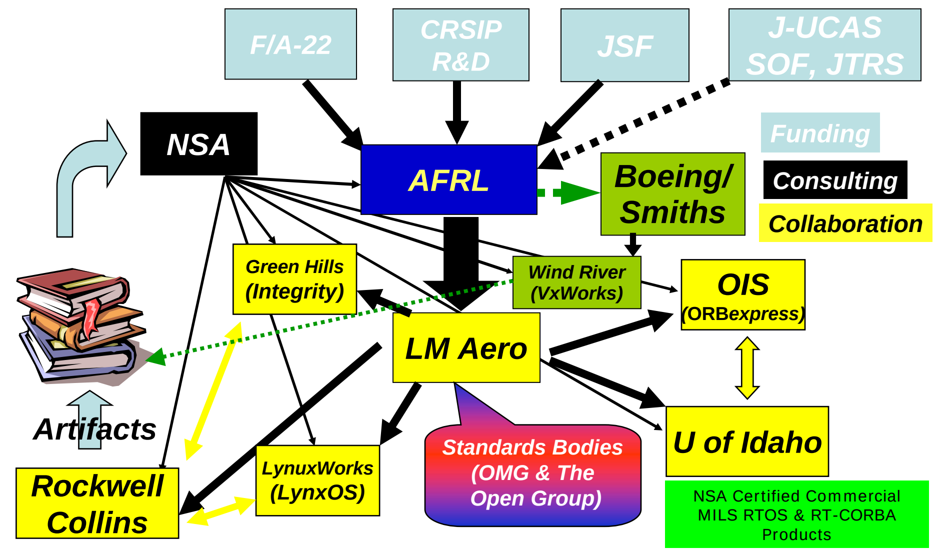
Industry Support

- At least three commercial RTOS vendors either have built, or are in the process of building, MILS-compliant operating systems:
  - Green Hills Software, Inc.
  - LynuxWorks, Inc.
  - Wind River Systems, Inc.

- Partners in the effort to integrate several MILS security Partitioning Kernels with a Real-time CORBA middleware implementation
  - National Security Agency
  - Air Force
  - Lockheed-Martin
  - Boeing
  - Objective Interface Systems, Inc.
  - Rockwell Collins
  - University of Idaho

- Effort should support an OMG standardization effort for high assurance Real-time MILS CORBA.

CRSIP R&D Led MILS Effort (2002 – Present)



Summary

- High Assurance, Deeply Embedded, Real Time, MILS/MSLS/MLS Systems are needed by the
  - War-Fighter,
  - Home Land Defense,
  - Telecommunications Systems,
  - Data Communications Systems,
  - Safety Critical Systems,
  - Process Control Systems,
  - Financial Systems,
  - Medical Systems,
  - et al
- The MILS/MSLS/MLS Partitioning Kernel architecture provides the lowest risk, quickest development time to provide high assurance systems

Acronyms

- MILS Multiple Independent Levels of Security/Safety
- MSLS Multiple Single Level Security/Safety
- MLS Multi-Level Secure/Safe
- PCS Partition Communication System
- CORBA Common Object Request Broker Architecture
- GIG Global Information Grid
- NEAT Non-bypassable, Evaluatable, Always-invoked,Tamper-proof
- NIU Network Interface Unit
- AIM Advanced INFOSEC Module
- ORB Object Request Broker
- O/S Operating System
- CC Common Criteria
- EAL Evaluation Assurance Level
- ARINC 653 Safety Community Standard for Time and Space Partitioning
- DMA Direct Management Access
- MMU Memory Management Unit

Partners

MILS Hardware Based Partitioning Kernel
AAMP7 **Rockwell Collins**

MILS Software Based Partitioning Kernel
Integrity-178 **Green Hills Software**
LynxOS-178 **LynuxWorks**
VxWorks AE Secure **Wind River**

MILS Middleware
ORB*express* **Objective Interface Systems, Inc.**
MILS TestBed **University of Idaho**
MILS TestBed **Naval Post Graduate School**