Just in Time Assurance

Jim Alves-Foss, PhD, University of Idaho
Director Center for Secure and Dependable Computing

W. Mark Vanfleet, National Security Agency
INFOSEC Systems Security Analyst
Global Network Vulnerability Analyst

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Just in Time Assurance	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Idaho,College of Engineering,Moscow,ID,83844-9802**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 22nd Systems and Software Technology Conference (SSTC), 26-29 April 2010, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**22**	

Weinberg's Second Law

"If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization."

- Anyone can get a system certified with enough time, money, threats, and waivers
- It takes skill to design a system that is practical and affordable to certify
- It takes a virtuoso to design a system that is practical and affordable to recertify given unpredictable but inevitable obsolescence events
- ***This presentation discusses how practical and affordable recertification can become the norm instead of the rare exception***

What Does Just in Time Mean?

- Manufacturing:
  - Parts arrive only when needed because inventory is waste.
    - Requirement: Quick notice of stock depletion
- Assurance:
  - Only modified components are reevaluated because total system reevaluation is waste.
    - Requirement: **NEAT** system architecture

What is NEATness?

- ***N***on-bypassable:
  - The infrastructure guarantees critical reference monitors in information flow paths can't be circumvented
- ***E***valuatable:
  - Each critical reference monitor is small and simple enough to enable unambiguous specification and rigorous evaluation
- ***A***lways Invoked:
  - Critical reference monitors enforce their ***local*** security policy for every object they manage
- ***T***amper Proof:
  - The infrastructure guarantees subversive agents can't modify any critical reference monitor's security functions or data.

Local Security Policy Enforcement

"**Critical reference monitors enforce their *local* security policy for every object they manage**"

- Why the ***local***?
  - A reference monitor should not know anything about any other part of the system
  - Reference monitor scope is constrained to the objects it manages
- A local reference monitor can be maintained, updated, and replaced with minimal effect on the rest of the system
  - A firewall or controlled interface in an enterprise network should not have knowledge about anything other than the policy it must enforce
  - A reference monitor in a real-time embedded system should not have knowledge about the specific platform on which it has been deployed
- A system can be certified, deployed, updated, recertified, and redeployed with reevaluation required only for the new components
- **RESULT:** the cost spiral caused by obsolescence events can be controlled

NEATness Verification

- Provide assurance that the infrastructure has security properties that protect reference monitors from ***TIME*** events
- ***TIME*****:**
  - ***T***ype safety violation
  - ***I***nfiltration violation
  - ***M***ediation violation
  - ***E***xfiltration violation

TIME: Type Safety Violation

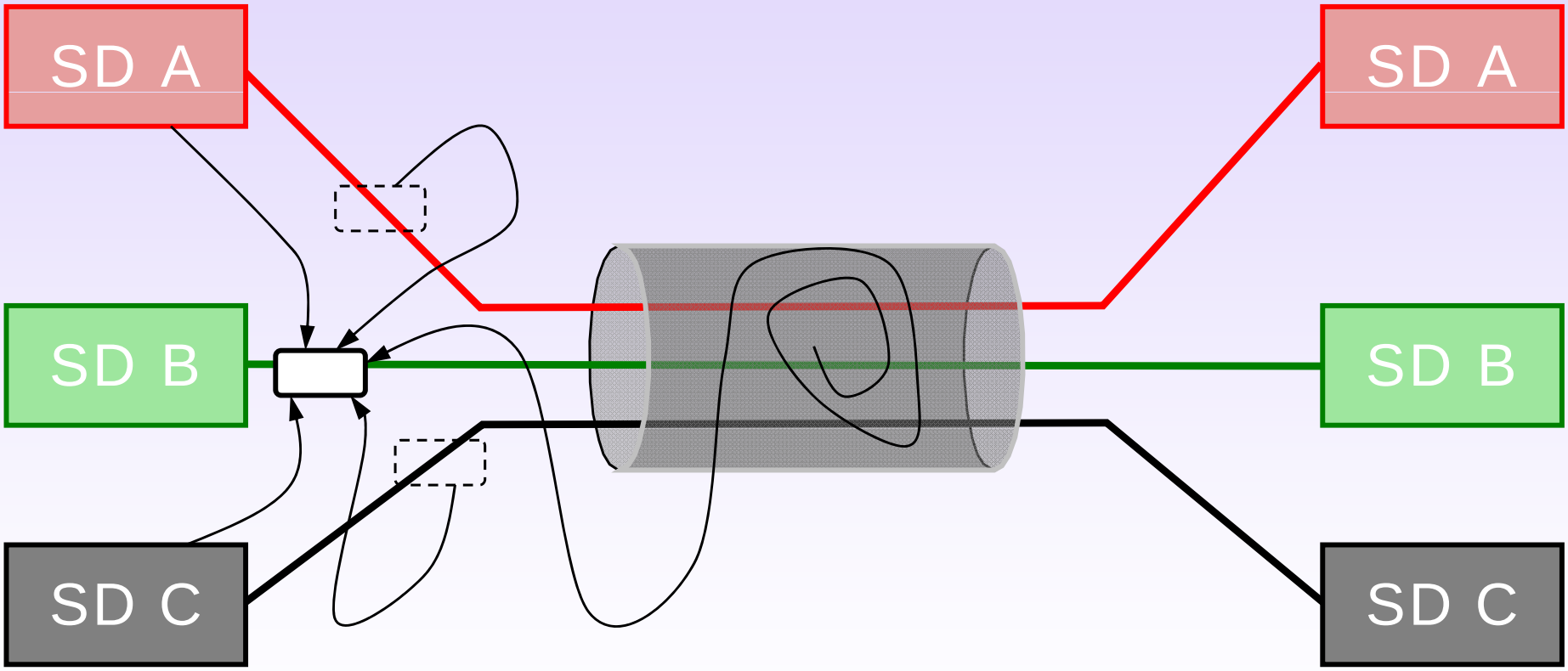
- When an object of one type is expected, a different type is delivered
- Consequences:
  - Buffer overflow
  - Address redirection
  - Unauthorized configuration modification
  - Activation of unintended code
    - Mission software turned into malware
    - Virus contagion
    - Root kit injection
    - Access control bypass

TIME: Infiltration

An unauthorized party may insert data into a channel, compromising integrity

- Party 1: An entity not authorized to send content on certain channels
- Party 2: Software, hardware, or systems that can attempt modification of traffic on certain channels but are not authorized to send content on those channels

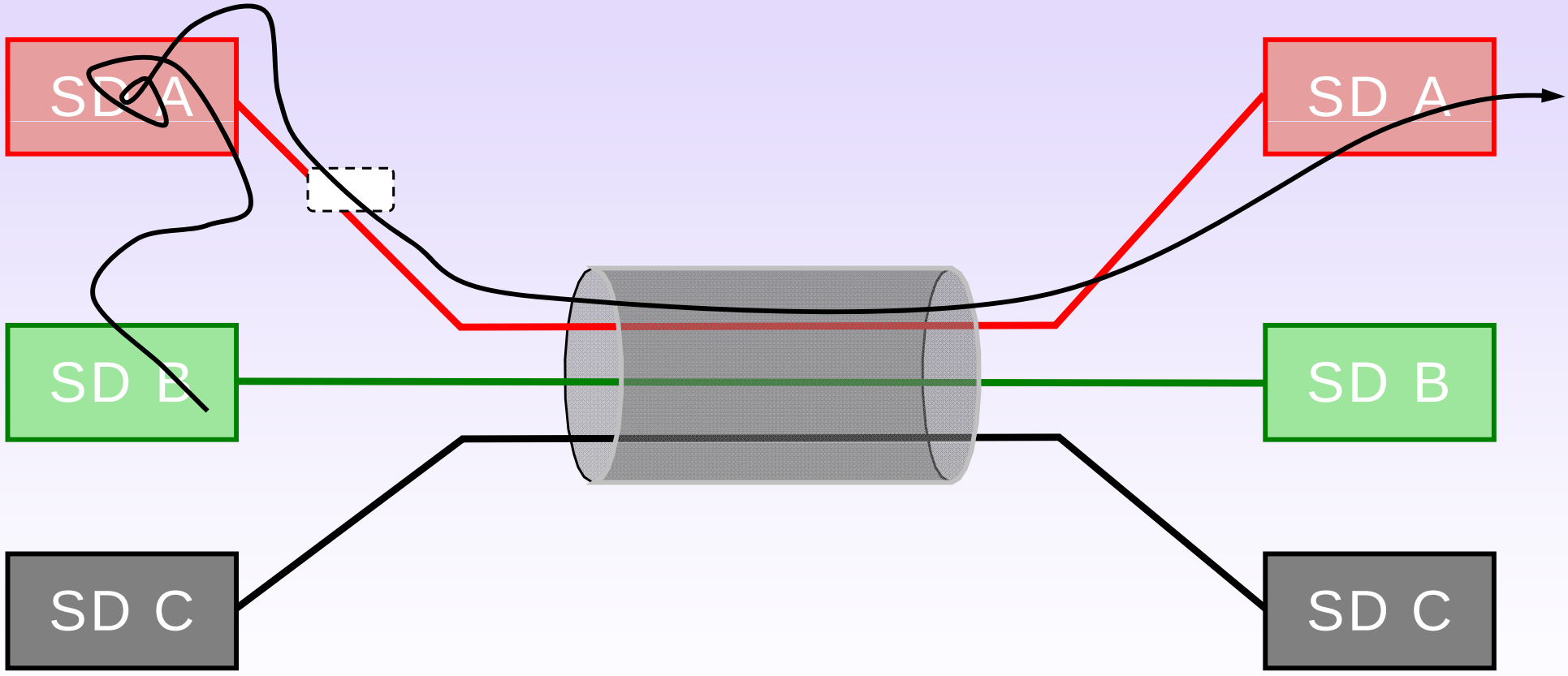
Infiltration



TIME: Mediation

- An unauthorized party may initiate or cause an information flow, compromising control
- Party 3: An entity that is not authorized to send content or cause content to be sent on certain channels

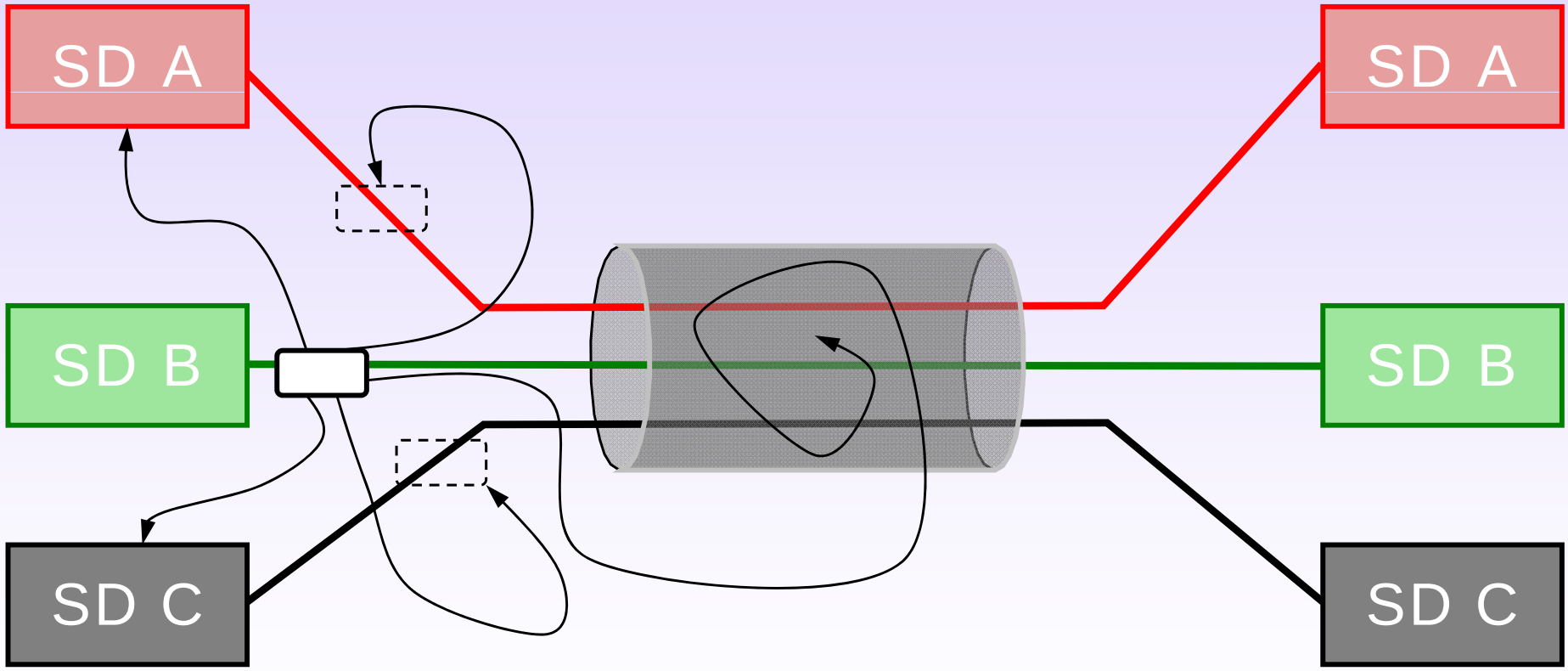
Mediation



TIME: Exfiltration

- An unauthorized party may perceive data in a channel, compromising confidentiality
- Party 4: An entity that is not authorized to receive content on certain channels
- Party 5: Software, hardware, or systems that can “see” all traffic of certain channels but are not authorized to receive the content of those channels

Exfiltration



Mitigating *TIME* Threats to *NEAT*ness

- Trustworthy separation enables security policy enforcement to be decomposed along structural lines
- Separation with respect to:
  - Space: Private data remains private
  - Time: Periods processing
  - Information Flow:
    - Only when initiated by authorized subjects,
    - Only delivered to authorized recipients
    - Sender authenticated to receiver

Benefits of Separation

- Specified interfaces are the only way that information may flow
  - **T**: Inappropriate data types can't be presented to an interface
  - **I:** There can be no infiltration (information pull violation)
  - M: There can be no mediation (control violation)
  - E: There can be no exfiltration (information push violation)
- Security Policy enforcement behaviors can be localized to each component reference monitor.
- Security policy architecture can then be decomposed as boxes and arrows

Boxes and Arrows Decomposition

- Boxes encapsulate objects
  - Access only local data and incoming communications
- Arrows are channels for information flow among boxes
  - Strictly unidirectional
  - Absence of arrows is just as crucial as their presence
- Draw enough boxes so that the ones that must be trustworthy are small and simple
  - Assume, for now, that boxes and arrows are “free”

Least Privilege Boxes

- A module trusted to enforce a system security policy in one layer can be untrusted in a different layer
- When a vulnerability in a reference monitor is found, it can be fixed without having to change anything else.
  - If we don’t change anything else, we don’t have to recertify the “anything else.”
- The architecture has done its job.

Security Policy Decomposition Benefits

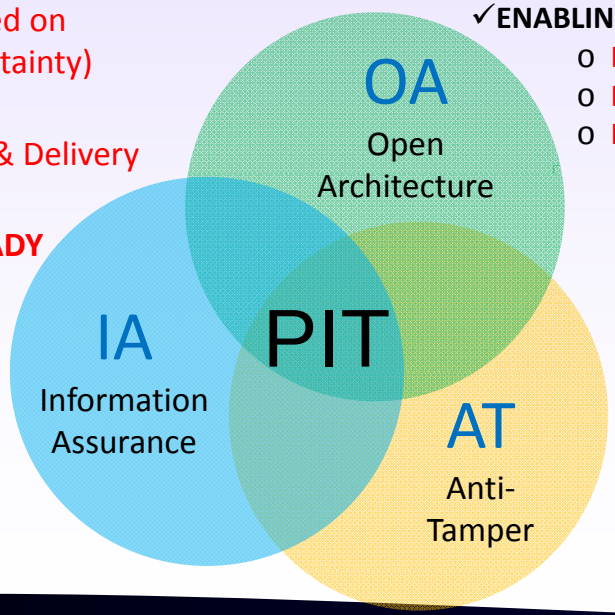
- Each least privilege security policy enforcement box is smaller, simpler, and more readily evaluatable
- Original security policy composition arguments remain unchanged despite obsolescence events
- Systems become more maintainable, adaptable, and extensible
- New threats from smarter and more experienced adversaries can be mitigated without redesigning the entire system

Compositional Assurance

- Compositional assurance is the path towards the goal of JIT Assurance
  - Construct individual assurance case for each trusted component
  - Provide argument that local policies combine to enforce the overall system policy
- ***Composability enables JIT Assurance***
  - A component can be patched, upgraded, refreshed, or replaced without affecting any other “parts”
  - Total system assurance is maintained at reasonable cost despite obsolescence events

Survivability

- ✓**CONFIDENTIALITY**
  - Critical Data **PROTECTED**
- ✓**INTEGRITY**
  - Free of Unauthorized Manipulation
- ✓**AUTHENTICATION**
  - Identity Confirmed
- ✓**AUTHORIZATION**
  - Privilege Confirmed
  - Mutual Suspicion
    (Reduced access based on authentication uncertainty)
- ✓**NON-REPUDIATION**
  - Proof of Data Origin & Delivery
- ✓**AVAILABILITY**
  - Critical functions **READY**



- ✓**DESIGNATE KEY INFORMATION EXCHANGES**
  - Standardize similar areas at Enterprise level across Primes
  - Blue force tracking, strike, mission planning , weather, …
- ✓**MODULARITY & VISIABILITY**
  - Enable affordable, safe and secure Technology Refreshes
  - Enable low cost rapid Technology Insertion
  - Enable recovery and adaptation against Zero-day Defects
- ✓**RE-USEABLE COMPONENTS**
  - Commercial based standards (POSIX, Open GL) - unmodified
  - Published standards (IEEE 1394, 802.11) - unmodified
  - Established proprietary standards (USB, Blue Ray) - unmodified
- ✓**INTEROPERABILITY & SECURITY (CJCSI 6212.01E)**
  - Global Network Information Enterprise Architecture
  - Support for Distributed degree of trust systems
- ✓**ENABLING ENVIRONMENTS**
  - Infrastructure and Enterprise API's Separable
  - Decouple data producers and consumers (cloud computing)
  - Register data grams and data streams within metadata registry

- ✓**DETERRENCE**
  - Undesirable Consequences
  - Strength of Mechanism
- ✓**PREVENTION**
  - Defense in Depth
  - Obfuscation
- ✓**DETECTION**
  - Visual, Alarm, Loss of Function, Attestation
  - Monitoring
- ✓**RESPONSE**
  - Destruction, Disabling, Zeroization
  - Adaption

System Life Total Cost of Ownership

Implementation

Certification / Accreditation

Deployment

Operations, Maintenance, and Administration

Technology Refresh

Growing Attack Surface over time

Obsolescence Events

Summary

- Separation enables JIT Assurance
- Networks of separated modules with proscribed frameworks to integrate them
- Trust of separation infrastructure to be verified.
  - Software implemented separation
    - Deployment of virtualization implementing isolation and redundancy
    - Requires validation of underlying hardware separation mechanisms (i.e., MMU, TPM, VT-d, etc.)
- Verification can be reused during all remaining steps in the system life cycle