Towards Formal Evaluation of a High-Assurance Guard

Mark R. Heckman <mark.heckman@aesec.com>

Roger R. Schell <roger.schell@aesec.com>

Edwards E. Reed <ed.reed@aesec.com>

2012 Layered Assurance Workshop, Dec 3 – 4, Orlando, Florida

Transfer Guards

- Type of *Cross Domain Solution*
- Permits the movement of data between domains
- Blocks the data that shouldn't be released
  - Key word scans, to block release of high data to low
  - Creation of sanitized data out of sensitive data
  - Antivirus scan for transfers from low integrity to high

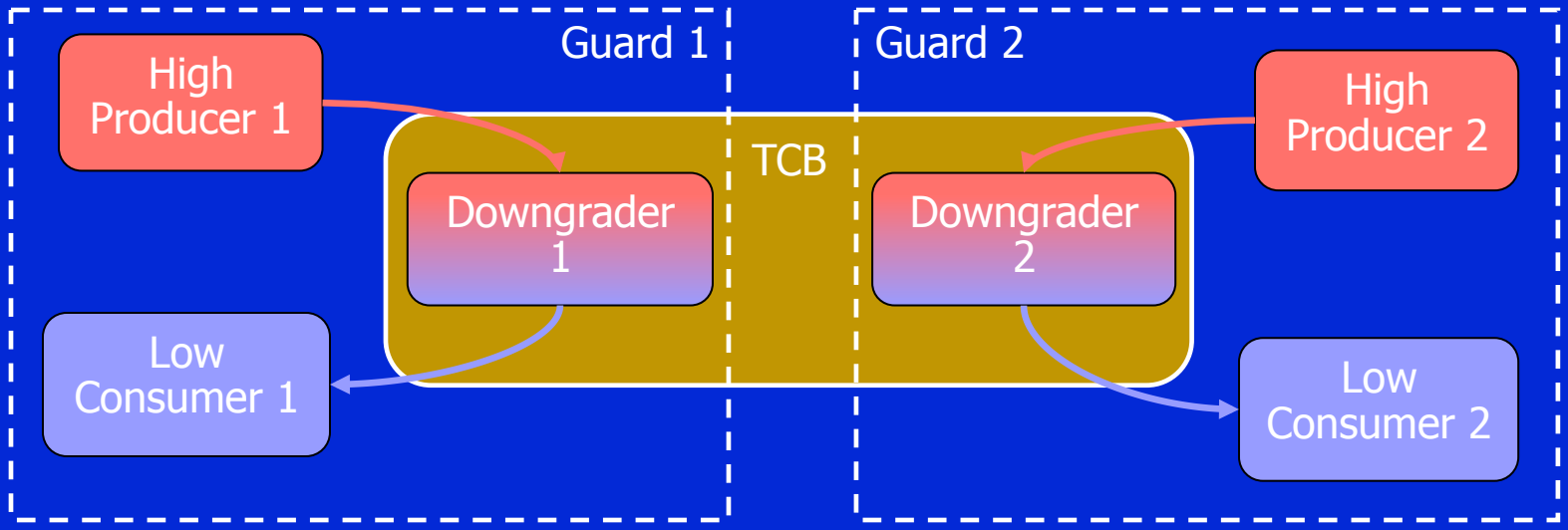
Current Transfer Guards

- Built on low-assurance base operating system
  - E.g., EAL 4+ (SELinux, Trusted Solaris)
- Can't limit range of trust
  - Need separate system for each guard/domain
  - Redundant systems: Guard server "farms"
- Susceptible to software supply-chain subversion
  - Trusted distribution? System integrity? Secure recovery?
- Lack sound system composition method
- Require repetitive, total system recertification
  - Due to changes to policy, HW/SW, configuration

High-assurance Transfer Guard

- Guard as application built on TCSEC Class A1 TCB
- A1 TCB base solves problems of
  - Susceptibility to supply-chain subversion
  - Resource redundancy due to range of trust
- Need sound composition method
  - Permits "divide and conquer" formal evaluation
  - Permits incremental (re)evaluation

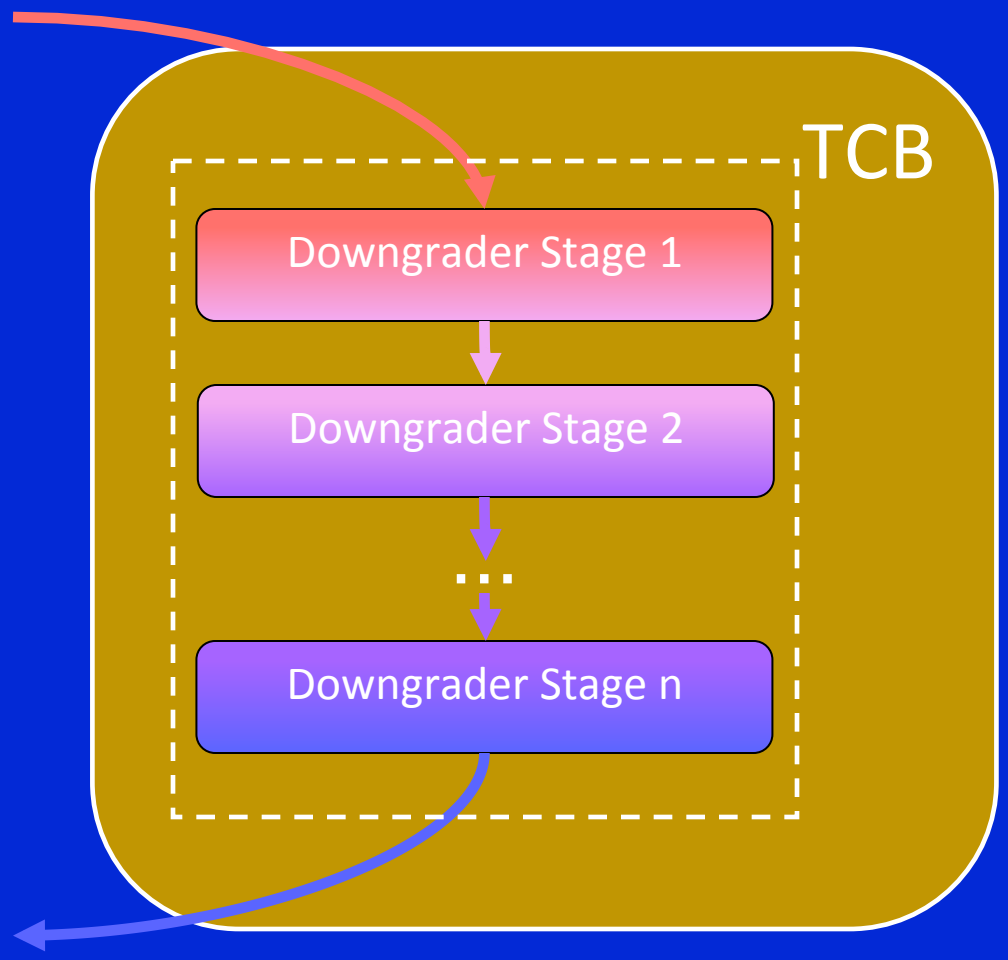
Abstract Transfer Guard



- Components:
  - High producer of information
  - Downgrader
  - Low consumer of information
- TCB permits multiple downgrade ranges

Abstract Downgrader

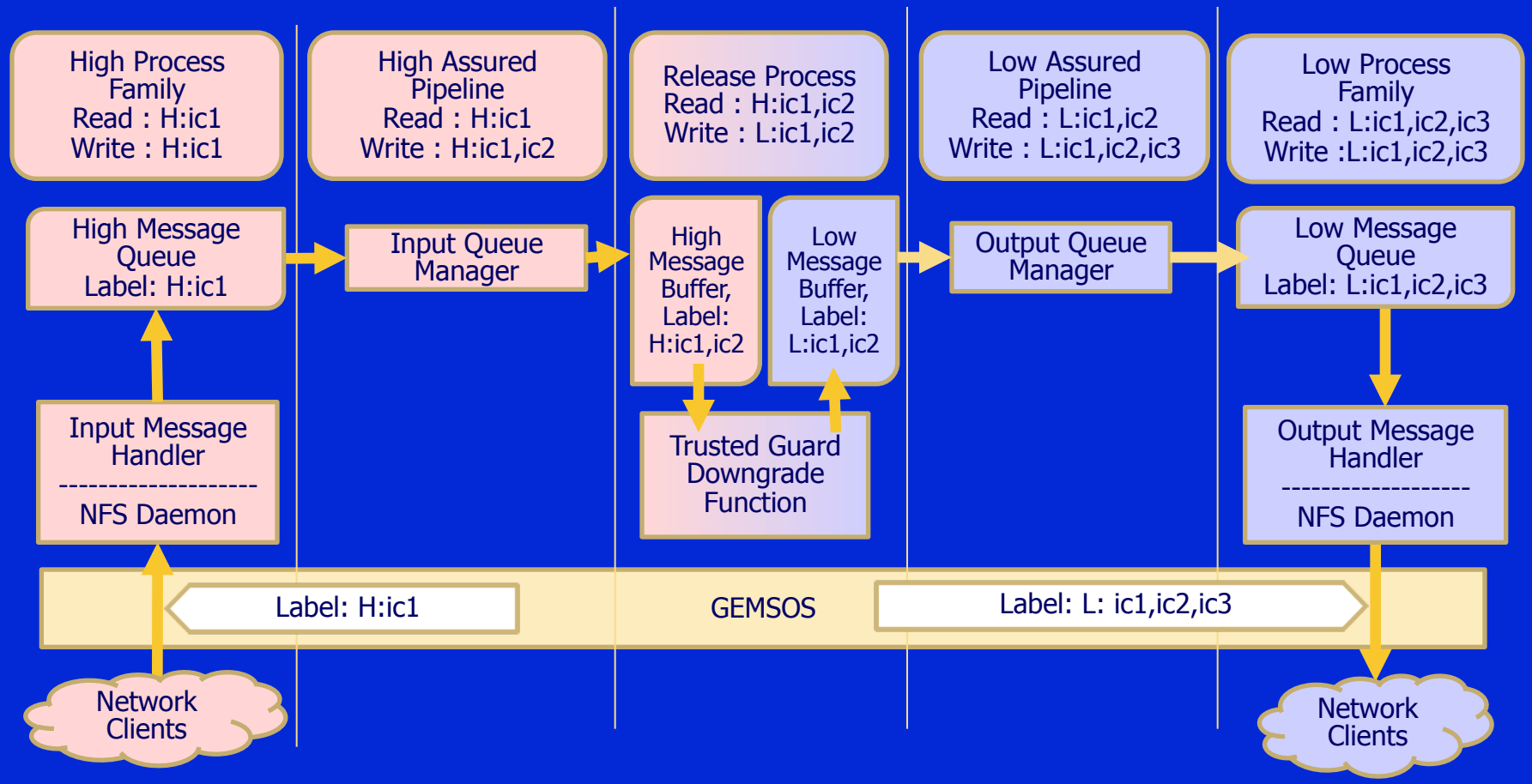
- Possibly multiple, distinct policies
- Pipeline or other arrangement



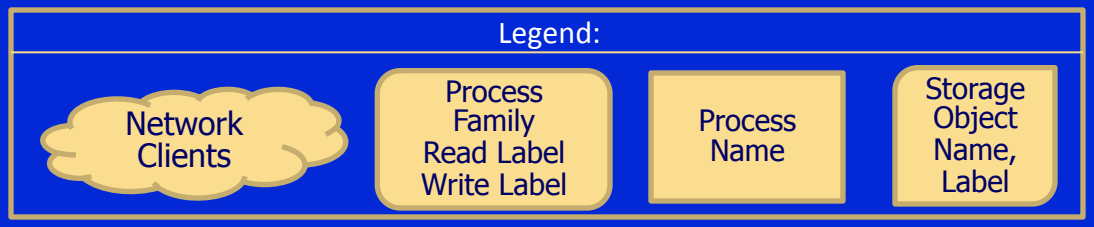
Aesec Virtual Guard (AVG)

- Implementation of Abstract Transfer Guard
- Designed for high-assurance base – GEMSOS
  - Gemini MLS Operating System implements
    - Bell-LaPadula access control
    - Biba mandatory integrity
  - Evaluated at TCSEC Class A1
- AVG takes advantage of GEMSOS mechanisms
  - Process isolation
  - Multi-level subjects with restricted range
  - Assured pipelines using GEMSOS MAC labels

AVG Architecture

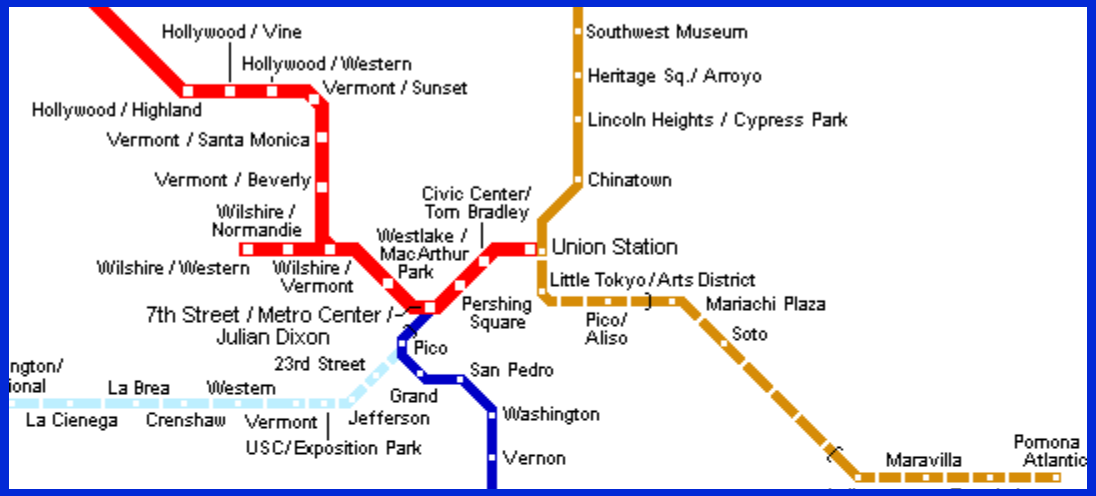


Labels use Secrecy Levels (H, L) and Integrity Categories (ic1, ic2, ic3)



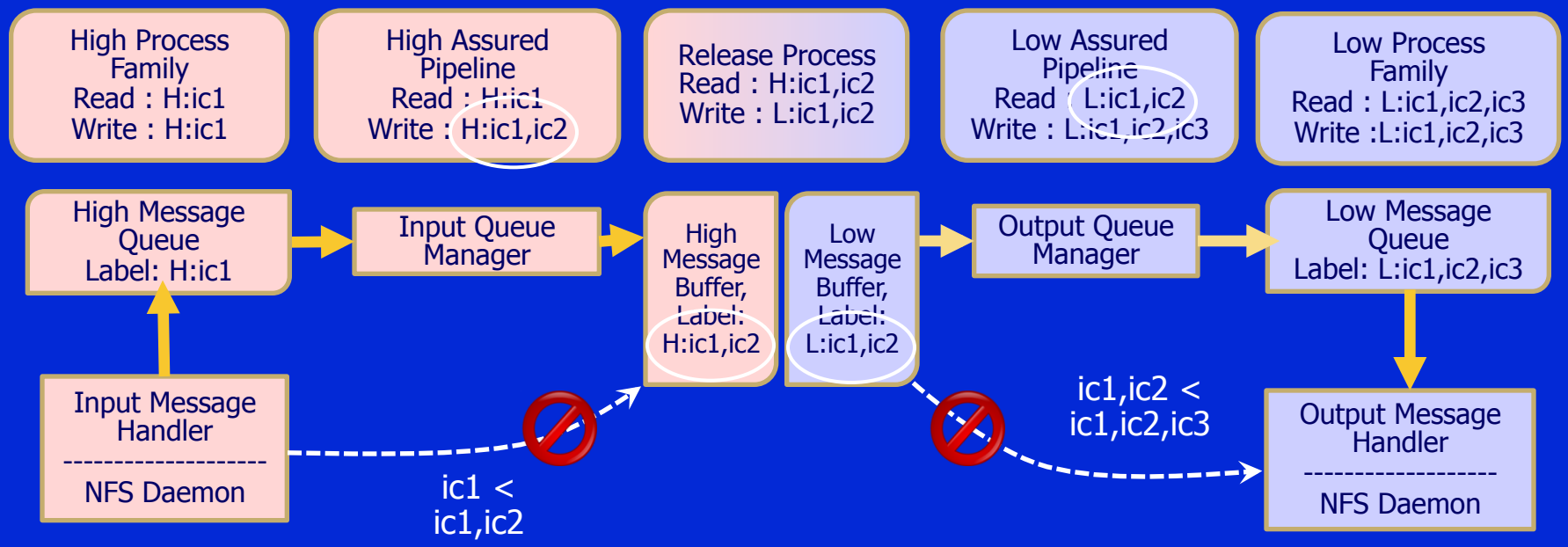
Assured Pipelines

- Sequences communication between processes
- Like a train, can't bypass a station
- Controls flow of data to/from downgrader
- Orders downgrader stages
- Implemented in AVG using Integrity categories



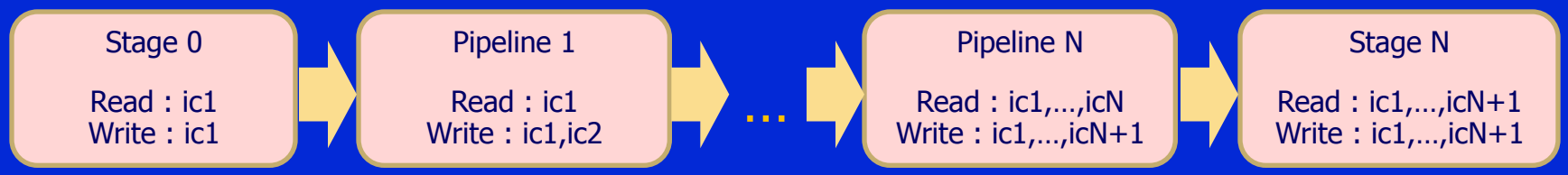
Assured Pipelines in AVG

- Biba Integrity:
  - The Simple Integrity Axiom: no read from a lower integrity level (no read down)
  - The * (star) Integrity Axiom: no write to a higher level of integrity (no write up)



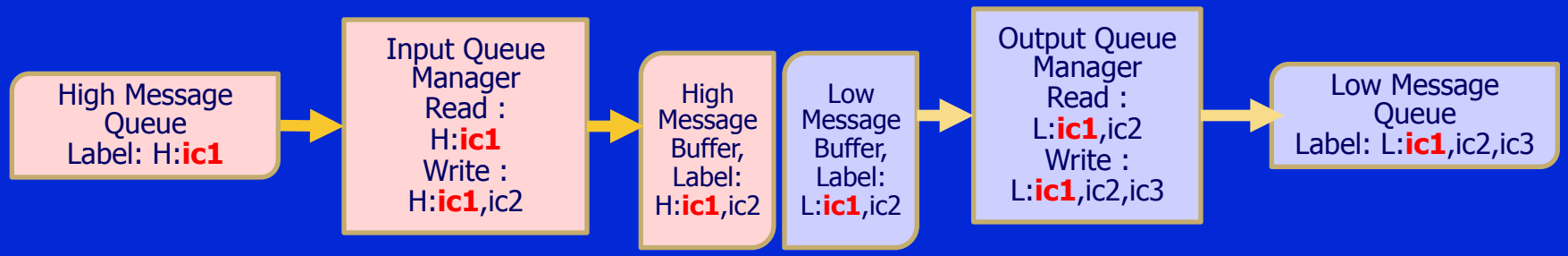
Adding Downgrader Stages

- Use additional integrity categories
- One additional category per stage
- Sequence of non-bypassable stages
- Modular, but confined, security policies



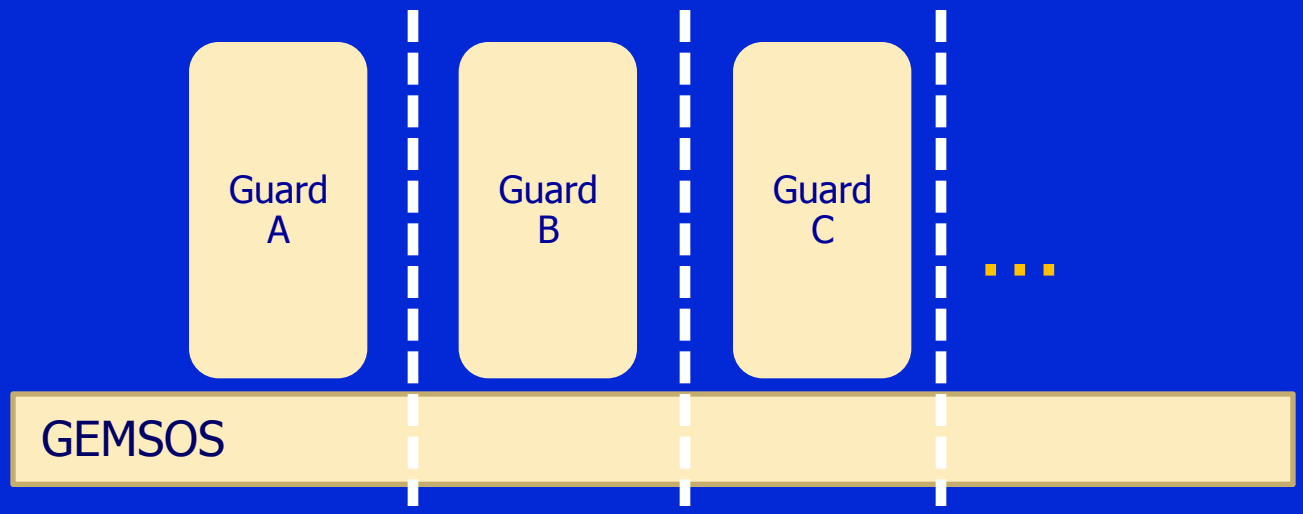
Guard Identifiers

- Start with basic process separation
- Add unique integrity category for each guard
  - "Guard identifier"
- "ic1" is guard identifier in figure
  - Assigned to every process and storage object



Multiple Virtual Guards

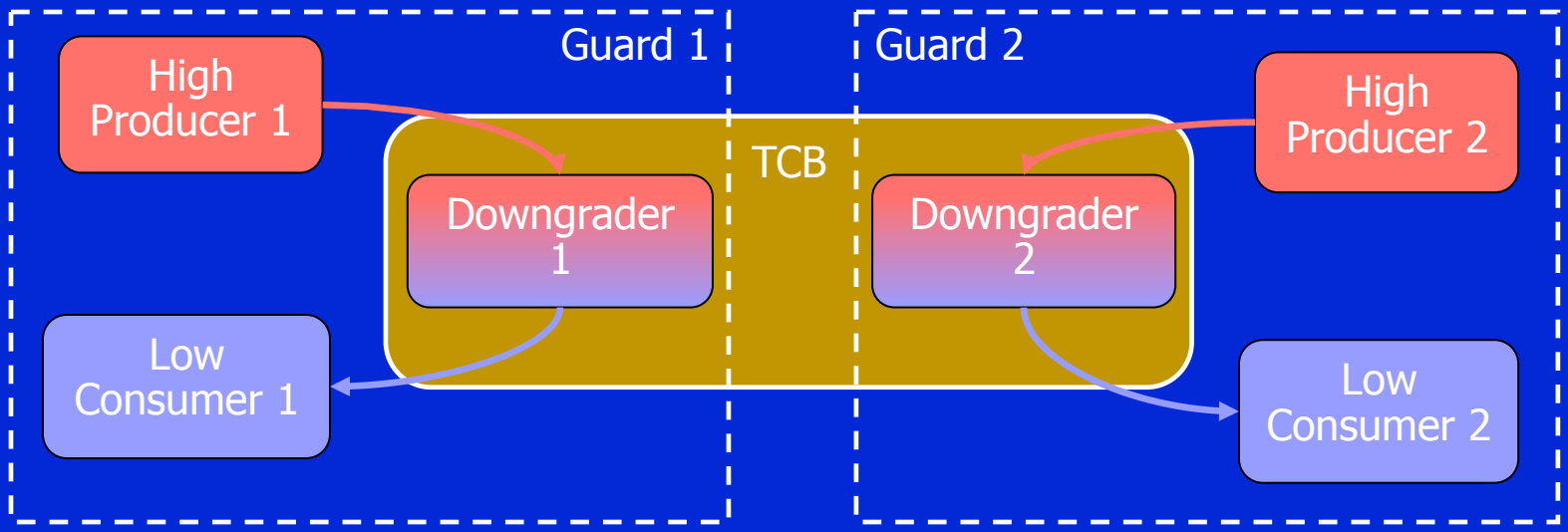
- Each guard has unique guard identifier
- Biba Integrity policy protects guard
  - Different integrity categories are non-comparable
  - No flow between guards on same server
- Can have different secrecy levels and ranges



Performance

- Prototype runs on single 550 MHz IA32 CPU
  - Did not use GEMSOS multiprocessing feature
- Single "dirty word" search downgrader
- Processing measured within server, independent of transfers
- 2500 msgs/sec
  - 4KB message size
- Primarily CPU bound

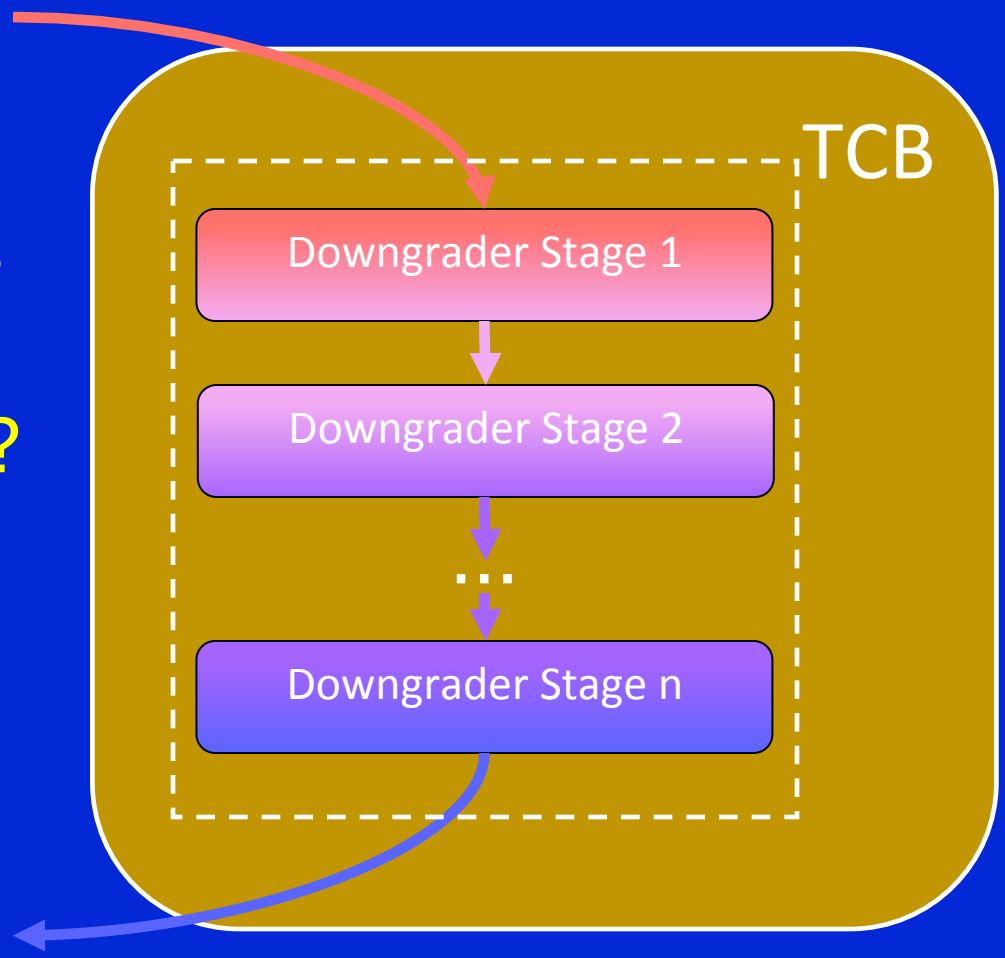
Abstract Transfer Guard Proofs



- Can Producer/downgrader/consumer be composed into a guard?
- Guards isolated?
- Downgrader evaluable separately from TCB?

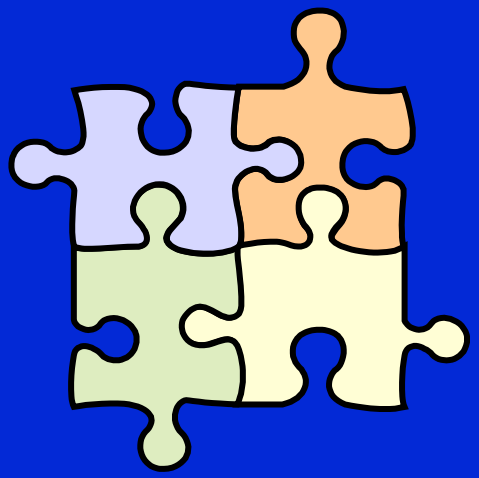
Abstract Downgrader Proofs

- Does each stage correctly enforce its policy?
- Stages composable?
- Ordering and isolation properties



Composition Problem

- Interconnected components
- Each has known security properties
- What are security properties of system?
- Want to use arguments about components to make arguments for entire system
  - Without having to reanalyze



Unconstrained vs. Constrained

- Constrained composition:
  - Specific properties
  - Specific interconnections
  - Practical solutions exist
  - E.g., TCB Partitions and TCB Subsets

- Unconstrained composition:
  - Arbitrary properties
  - Arbitrary interconnections
  - No general solution known (or possible?)

Classifying Proof Obligations

- Infrastructure Constrained composition
  - Guards isolated?
  - Ordering and isolation properties of downgrader stages
  - Can Producer/downgrader/consumer be <u>composed</u> into a guard?
  - Downgrader evaluable separately from TCB?

- Non-infrastructure Unconstrained composition
  - Does each stage correctly enforce downgrade policy?
  - Can stages be <u>composed</u> into the downgrader?

Guard Isolation/Stage Ordering

- GEMSOS-enforced process isolation
- GEMSOS mandatory integrity used for …
- Guard isolation:
  - Guard identifier integrity categories
- Downgrader stage isolation and ordering
  - Assured pipelines
  - Implemented using integrity categories

Producer/Downgrader/Consumer

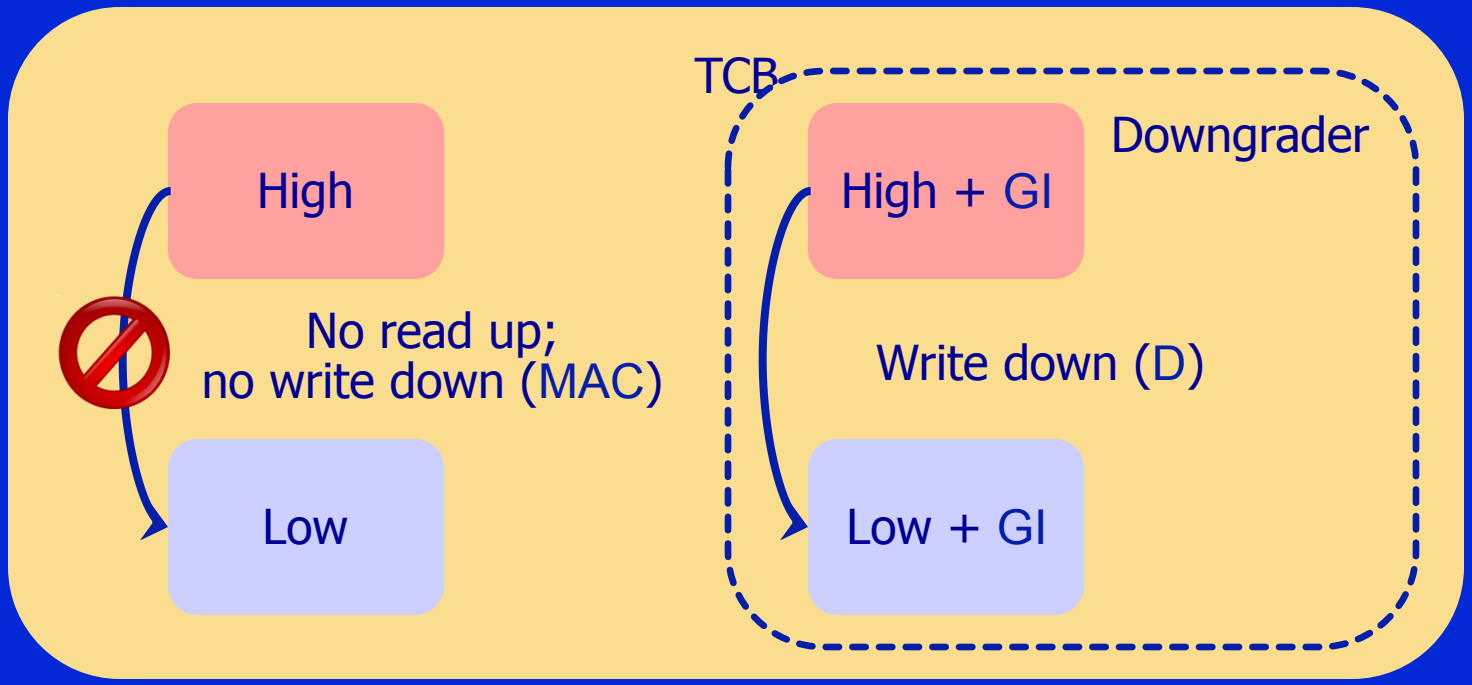
- Trusted Network Interpretation of TCSEC
  - Virtual machine is example of "network component"
- Downgraders are virtual machines
  - Due to guard identifiers
- Downgrader on TCB is NTCB partition
- Producer and consumer are also NTCB partitions
- Use TCB Partitions technique (TNI) to compose

Downgrader Separate from TCB

- Two policy-enforcing entities
- TCB policy
  - Includes notion of multilevel subject
  - Limits range of multilevel subject
  - Isolates trusted subject
- Downgrader policy
  - Defines what trusted subjects do within range

System Policy Specification Sketch

- Mandatory access control (MAC)
- Downgrader isolated by guard identifier (GI)
- Downgrader protected domain policy D



TCB Subsets (from TDI)

- Less-primitive subset (Downgrader)
- More primitive subset (TCB)
  - Strict MAC policy
  - Permits trusted subjects in less-primitive subset domain
- Tamperproof and mandatory system config.
  - Guard identifier creates isolated protection domain
  - Downgrader range
- Security properties of TCB unaffected by changes to the downgrader policy
- Hence, can use TCB Subsets technique

Balanced Assurance

- Less-primitive TCB subsets
  - Constrained by underlying, general-purpose TCB
  - Have correspondingly lower risk
  - Do not require all Class A1 assurance techniques
- Downgrader is such a less-primitive TCB subset
- But, downgrader is trusted
- Does it have lower risk?
- Yes, because does only one thing – downgrading
- TCB-enforced isolation prevents anything else

Downgrader Arguments

- Program verification needed
- But what is policy model?
- Multiple policies ➔ Unconstrained composition

- AVG provides necessary support
- Verified programs requires code integrity
  - A1 TCB specifically addresses subversion
- Composition of stages requires assured ordering
  - Assured pipelines and guard identifier

Conclusions

- Can evaluate downgrader separately from TCB
  - Permits "divide and conquer" formal evaluation
  - Permits incremental (re)evaluation
- Supports multiple downgraders on same system
  - Due to high-assurance base
- Separates constrained from unconstrained composition
  - Increase confidence in overall evaluation correctness

Questions?

Towards Formal Evaluation of a High-Assurance Guard

Mark R. Heckman <mark.heckman@aesec.com>

Roger R. Schell <roger.schell@aesec.com>

Edwards E. Reed <ed.reed@aesec.com>

2012 Layered Assurance Workshop, Dec 3 – 4, Orlando, Florida