Layered Attestation of a Cross Domain System

An Experiment in Runtime Attestation

Will Thomas, Logan Schmalz, Sarah Johnson, Adam Petz, Perry Alexander

Institute for Information Sciences, The University of Kansas

`{30willthomas,loganschmalz,sarahjohnson,ampetz,palexand}@ku.edu`

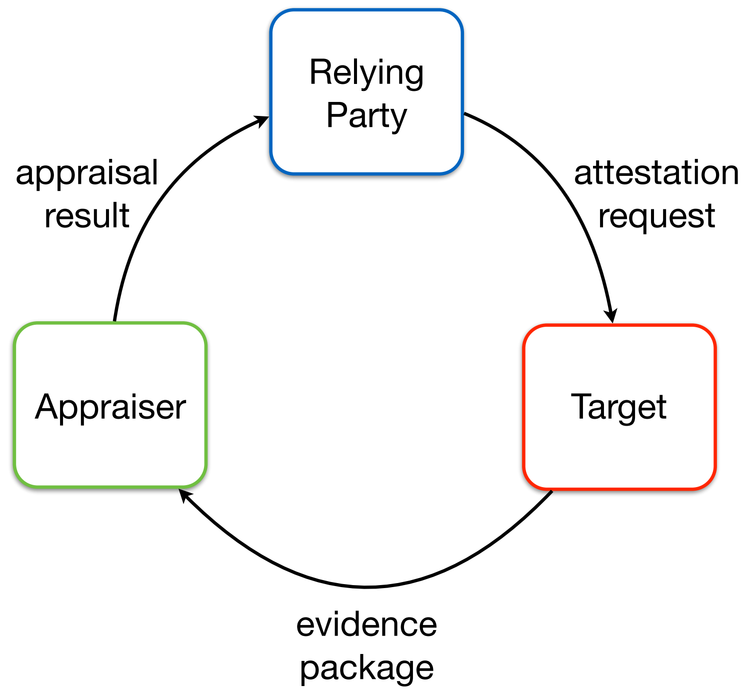
Joshua Guttman, Paul Rowe

The MITRE Corporation

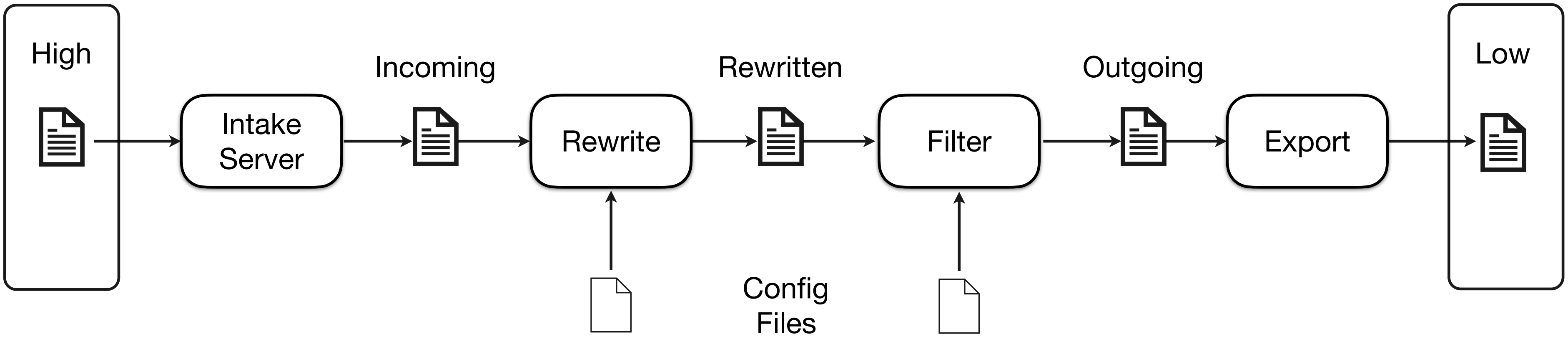
`{guttman, prowe}@mitre.org`

Semantic Remote Attestation

- Relying Party requests appraisal
  - specifies needed information
  - provides a fresh nonce
- Target gathers and generates evidence
  - measures OS & applications
  - generates cryptographic signatures
- Appraiser assesses evidence
  - good application behavior
  - infrastructure trustworthiness
  - good nonce



Attestation of a Cross Domain System

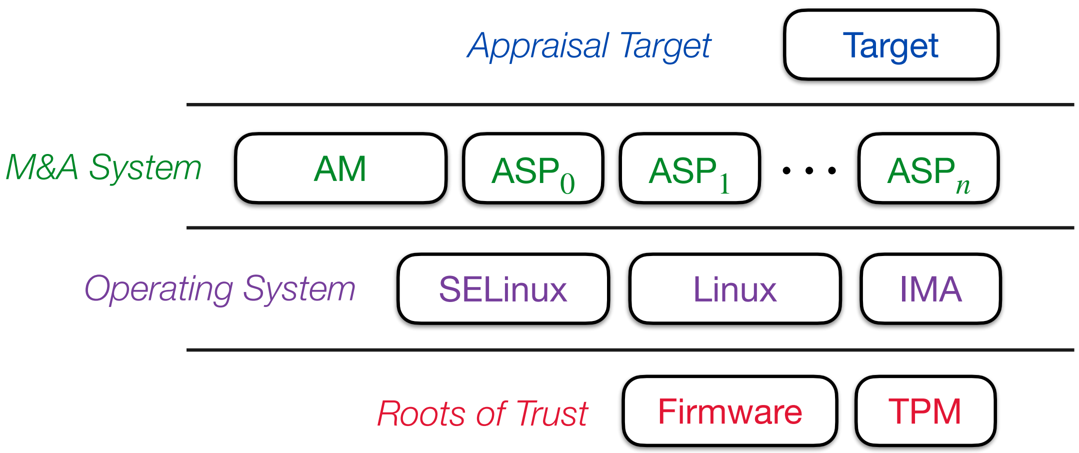


- **Research Goals**
  - establish TPM as a functioning root of trust
  - bring up a trustworthy runtime attestation system
  - perform runtime attestation on the CDS
  - perform empirical testing

- **Tools and Infrastructure**
  - Copland attestation protocol language
  - verified MAESTRO attestation synthesis tools
  - formally verified MAESTRO attestation manager
  - Invary LKIM

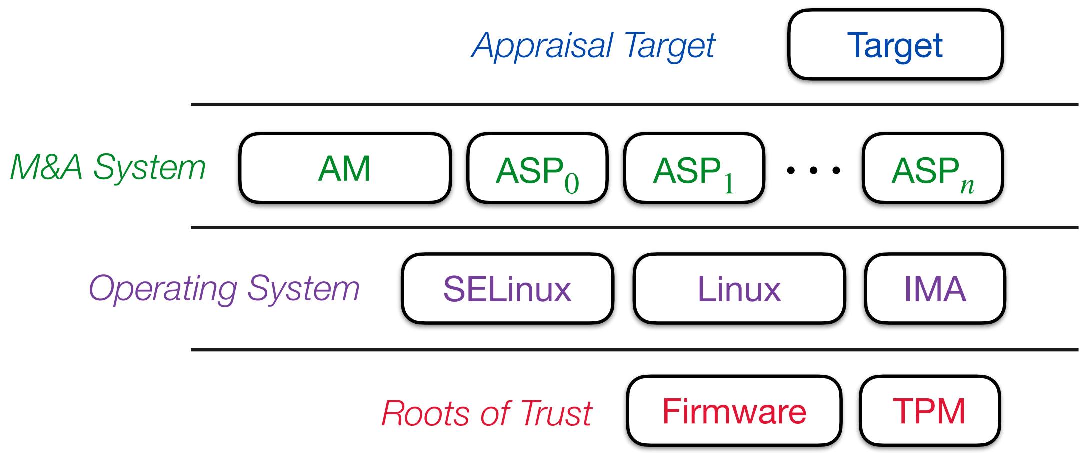
Layered Runtime Attestation

- Target
  - system to be appraised at runtime
  - cross domain system for this experiment
- M&A Subsystem
  - MAESTRO attestation manager (AM)
  - attestation manager key (AM^{-1})
  - attestation service providers (ASPs)
  - Copland attestation protocol
- Operating System
  - RedHat Linux
  - SELinux
  - IMA
- Roots of Trust
  - storage and reporting (TPM)
  - measurement (Firmware)



Layered Runtime Attestation

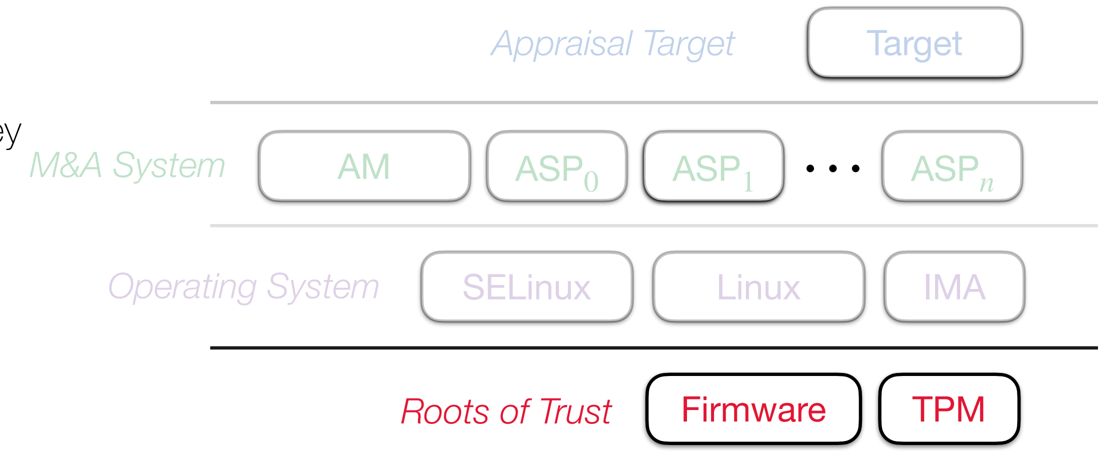
- Evidence { E_0 ;; E_1 ;; … ;; E_n }$_{AM^{-1}}$
  - E_k - evidence from ASP$_k$ execution
  - ;; - bundling operator from protocol
  - $\{\ldots\}_{AM^{-1}}$ - Attestation Manager signature
- AM^{-1} signing key
  - signing memorializes good boot
  - only AM can access the key
  - access is allowed only on good boot to AM



$$\left.\begin{array}{l}\text{proper bundling}\\ \wedge\ \text{satisfies appraisal policy}\\ \wedge\ \text{valid signature}\end{array}\right\} \Rightarrow \text{trustworthy target}$$

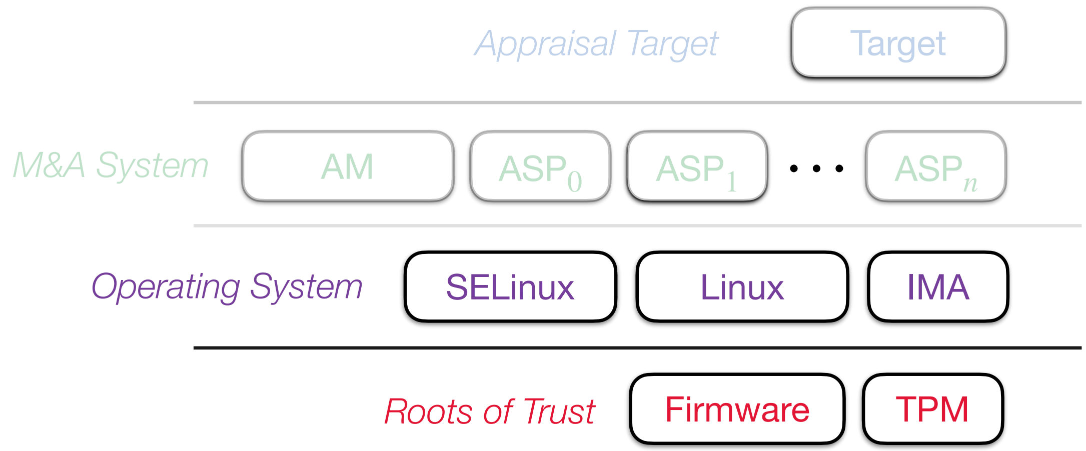
Roots of Trust Base

- TPM
  - Root of trust for Storage and Reporting
  - trusted *a priori*
  - evidence signing
  - generates, stores and seals AM's signing key
  - binds signing key to an AM
- Firmware
  - Root of trust for Measurement
  - trusted *a priori*
  - bootloader measurement and initiation



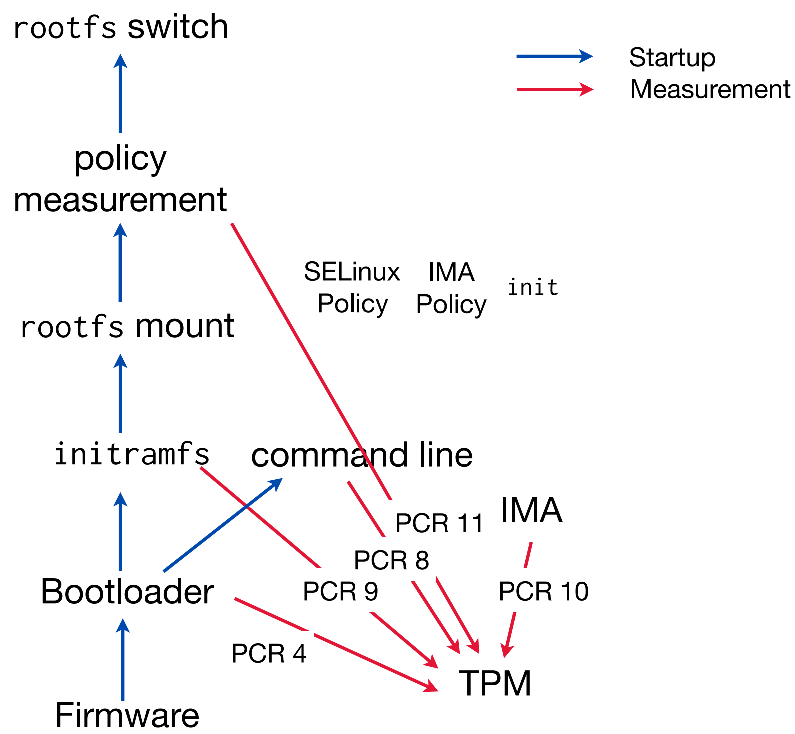
Operating System Layer

- Measure and start Linux
- Measure policy and start SELinux
- Measure policy and start IMA



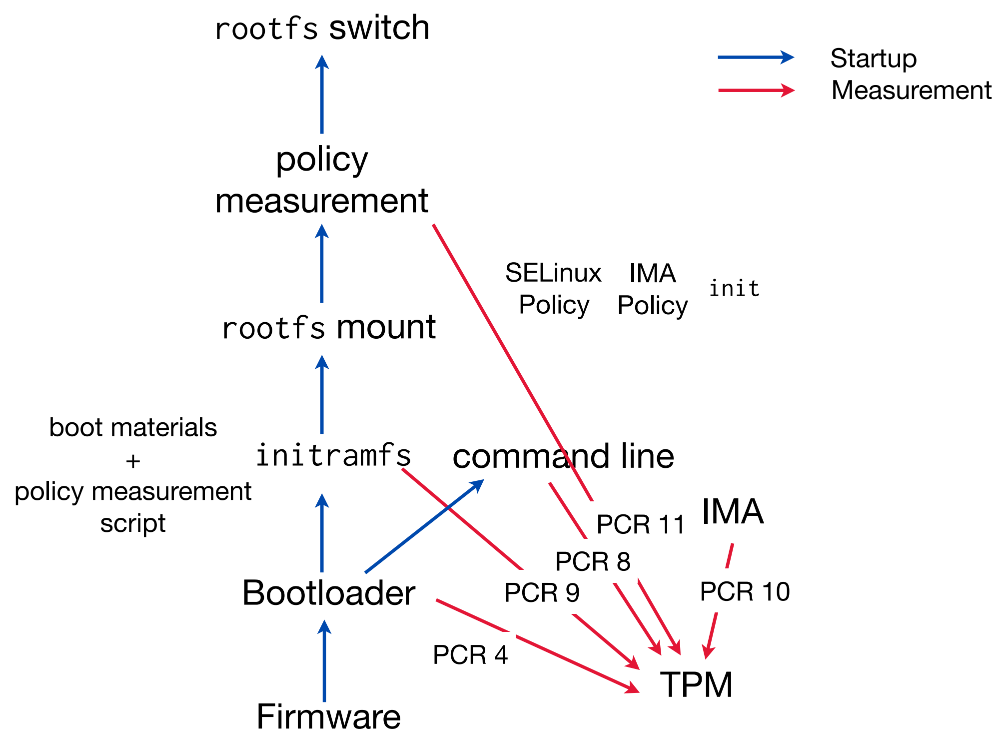
Trusted OS Infrastructure

- Firmware measures and starts boot loader
  - firmware hashes and starts boot loader (PCR 4)
- `initramfs` contents
  - traditional boot materials
  - custom measurement script for SELinux and IMA policies and `init` system
  - IMA will use SELinux types requiring early policy measurement and SELinux start
- Boot `initramfs`
  - bootloader hashes command line to start `initramfs` (PCR 8)
  - bootloader hashes and starts `initramfs` (PCR 9)
- Switch to `rootfs`
  - mount `rootfs`
  - hash IMA and SELinux policies (PCR 11)
  - hash `init` binary
  - execute `init` binary on `rootfs`
  - kernel running with measured IMA and SELinux policies



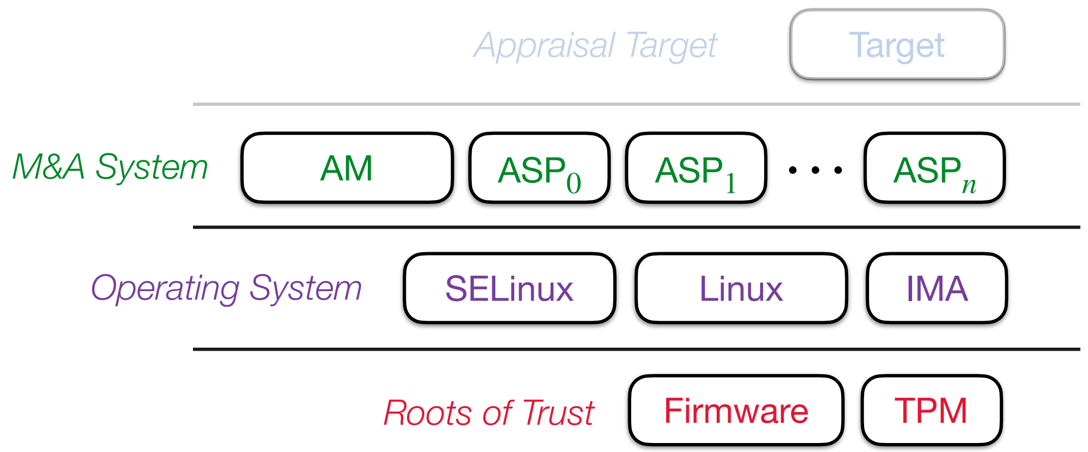
TPM State

- Good PCR 4
  - good bootloader
  - should measure `initramfs`
  - should use command line specification to start
- Good PCR 8 & 9
  - good command line starts `initramfs`
  - good `initramfs`
  - good boot materials
  - *good policy measurement script*
  - *good measurement script invocation*
- PCR 10 (ignored)
  - memorializes IMA trace
  - not useful for sealing
- Good PCR 11
  - policy measurement ran
  - good initial SELinux and IMA policies
  - good `init` indicates start with good policies



Runtime Attestation Layer

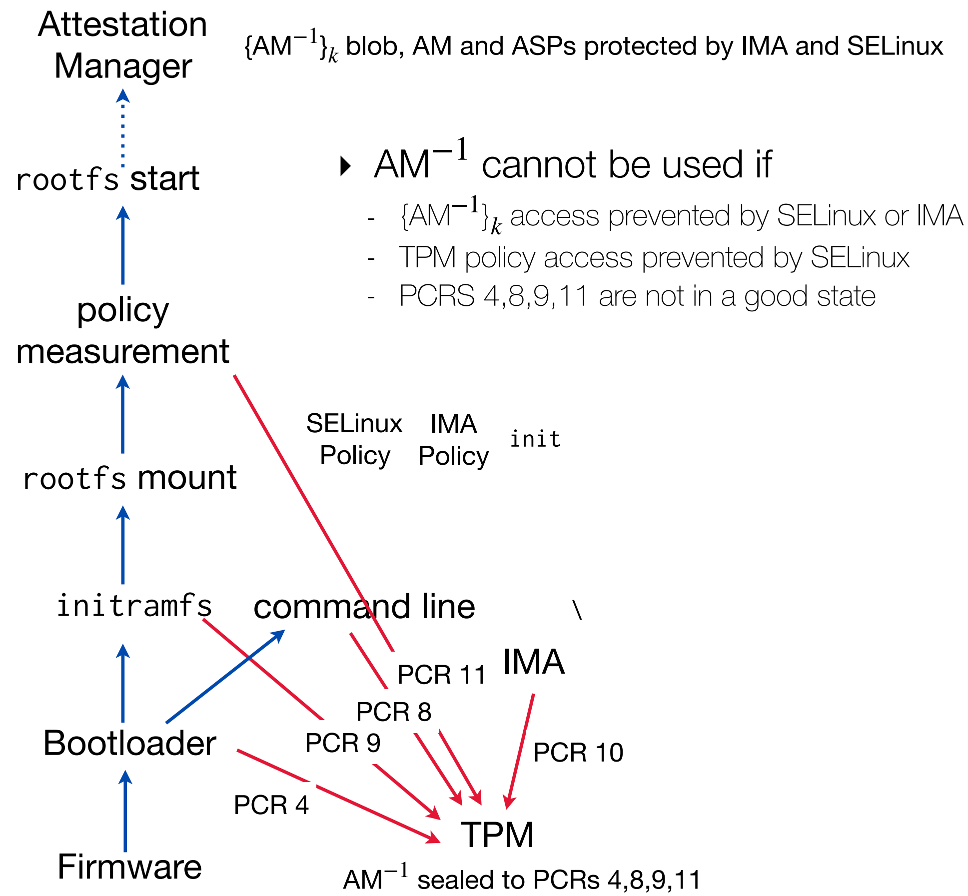
- Measure and start AM
- Establish ASP libraries
- Ensure AM^{-1} availability
- Begin Copland protocol execution



AM^{-1} Protection and Use

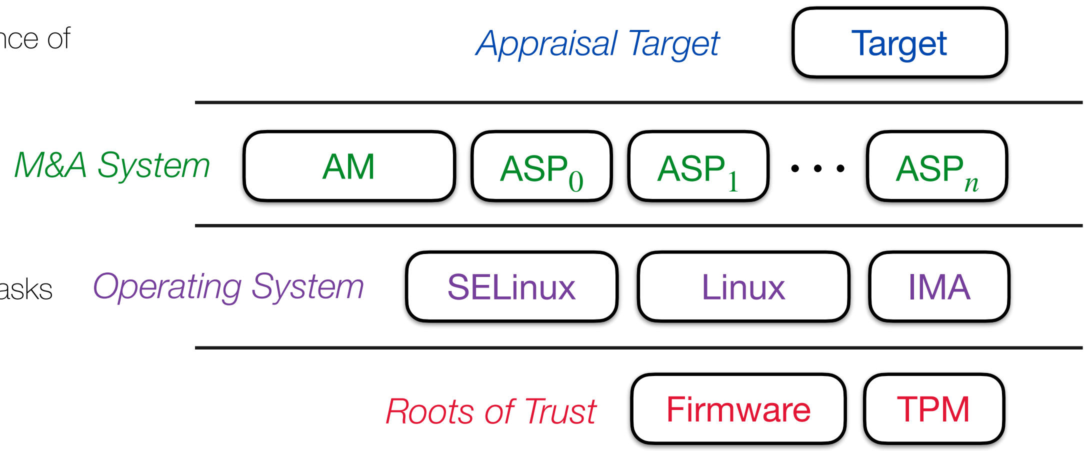
- Starting and Protecting AM
  - IMA policy prevents bad AM binary starting
  - IMA policy prevents bad ASPs from running
  - SELinux provides runtime access control
  - AM is formally verified to properly execute Copland protocols
- Generating and Protecting AM^{-1}
  - TPM generates AM^{-1} from $\{AM^{-1}\}_k$ blob
  - SELinux enforces $\{AM^{-1}\}_k$ access control
  - IMA Extended Verification Mode (EVM) protects $\{AM^{-1}\}_k$ permissions
  - Authorized TPM policy must be loaded to enable key
  - SELinux enforces access control over TPM Policy
  - Authorized Policy seals AM^{-1} to PCRs 4,8,9,11
- Using AM^{-1}
  - key is a strongly bound identifier for the AM
  - AM signature binds evidence to the associated AM
  - AM signature memorializes boot
  - effectively extends trust to user-space attestation mechanisms

- AM^{-1} cannot be used if
  - $\{AM^{-1}\}_k$ access prevented by SELinux or IMA
  - TPM policy access prevented by SELinux
  - PCRS 4,8,9,11 are not in a good state



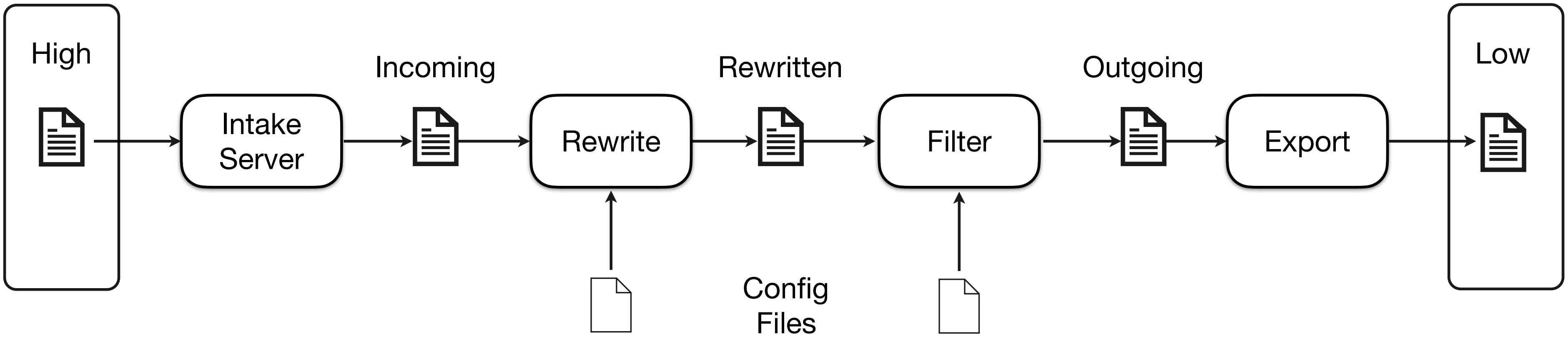
General Purpose Runtime Attestation

- Boot to AM is generic
  - any good signature over evidence $\forall e \,.\, \{e\}_{AM^{-1}}$ is evidence of trusted AM
  - configurable, formally verified
  - small, memory safe
- M&A Subsystem
  - runs arbitrary Copland attestation protocols
  - attestation service providers (ASPs) perform attestation tasks
  - Copland attestation protocols sequence ASP execution
  - AM signing itself is an ASP
- Appraisal Targets
  - customize ASPs and protocol for specific applications
  - no requirement to customize target
- Evidence $\{ E \}_{AM^{-1}}$
  - check signature to assure evidence integrity and good boot
  - check evidence to establish trust in target
  - formal semantics for protocol and evidence



$$\left.\begin{array}{l}\text{proper bundling}\\ \wedge\ \text{satisfies appraisal policy}\\ \wedge\ \text{valid signature}\end{array}\right\} \Rightarrow \text{trustworthy target}$$

Cross Domain System

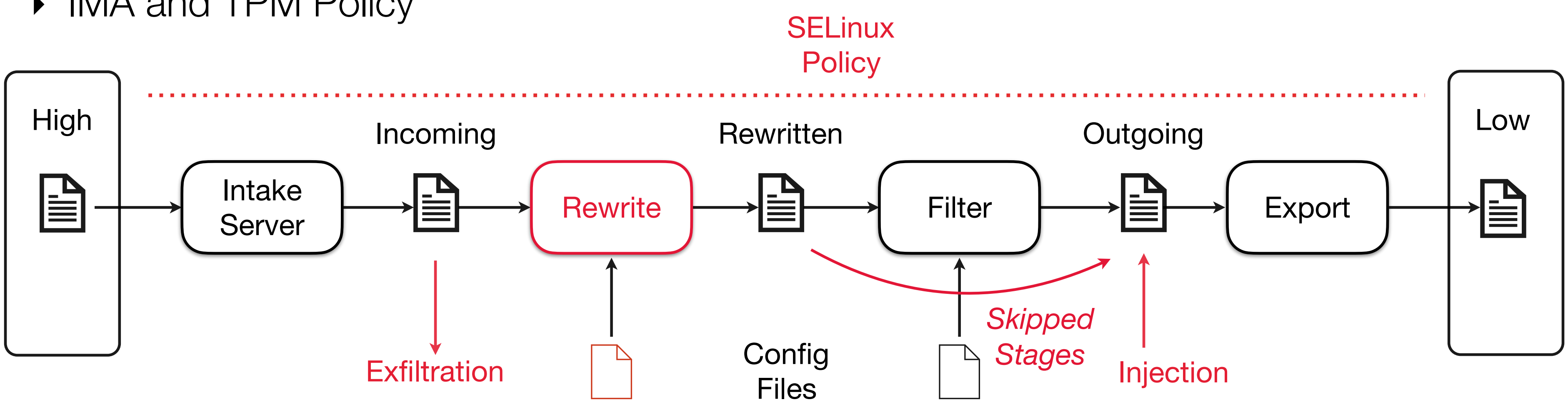


- Moving messages between security domains
  - intake receives a message from the high-side writes to incoming buffer
  - rewriter reads from the incoming buffer, applies rewrite rules, and writes to rewritten buffer
  - filter reads from the rewritten buffer, applies address filtering rules, and writes to outgoing buffer
  - export reads from outgoing buffer and outputs to low-side client
- Configuration
  - rewrite and filter processes have configuration files
  - SELinux policy enforces flow through the system
- Messages reaching the low-side client must be:
  - received from the high-side client
  - rewritten by a properly configured rewriter
  - filtered by a properly configured filter

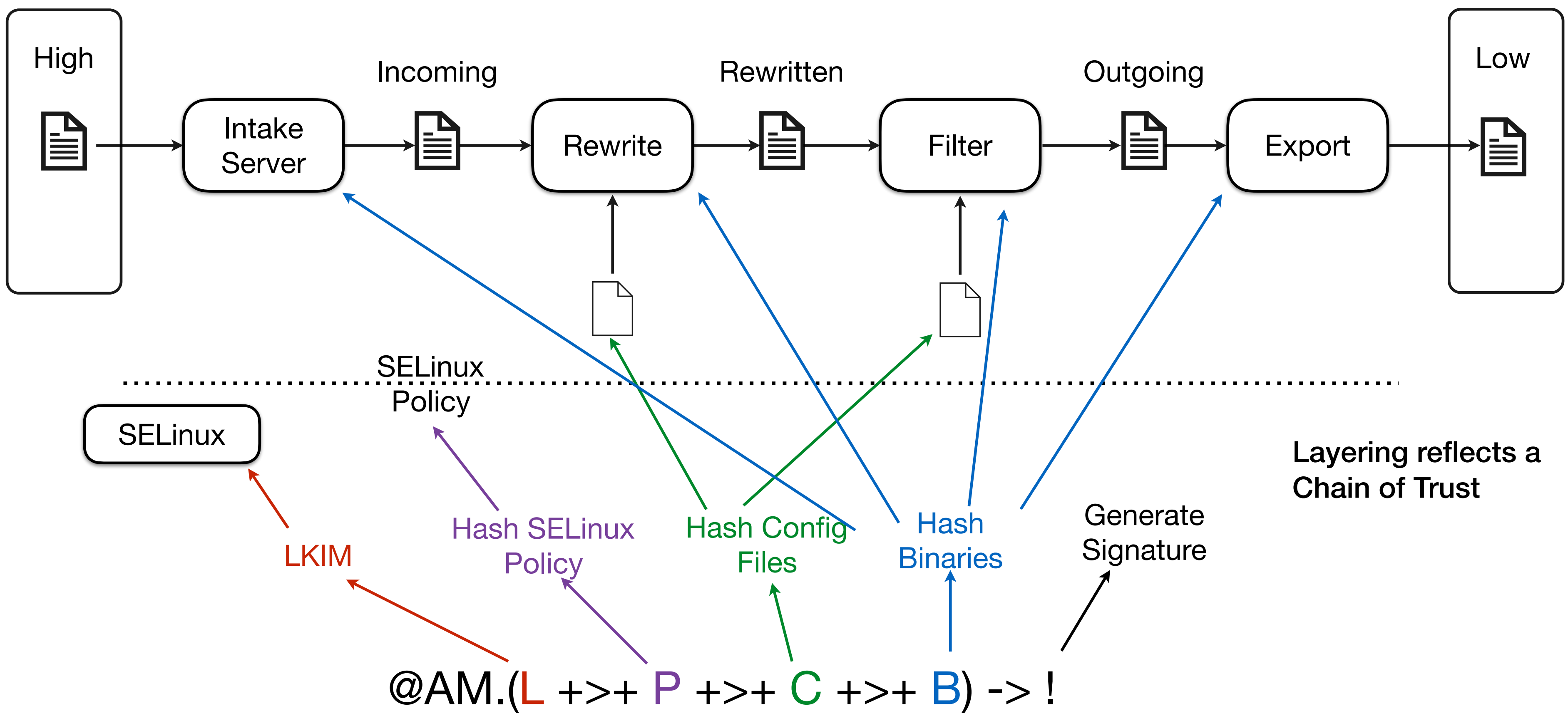
Adversary Targets

- Configuration files for pipeline binaries
- Pipeline binaries themselves
- Communication paths and buffers
- SELinux Policy
- IMA and TPM Policy

The adversary's primary goal is convincing a relying party to trust something it should not



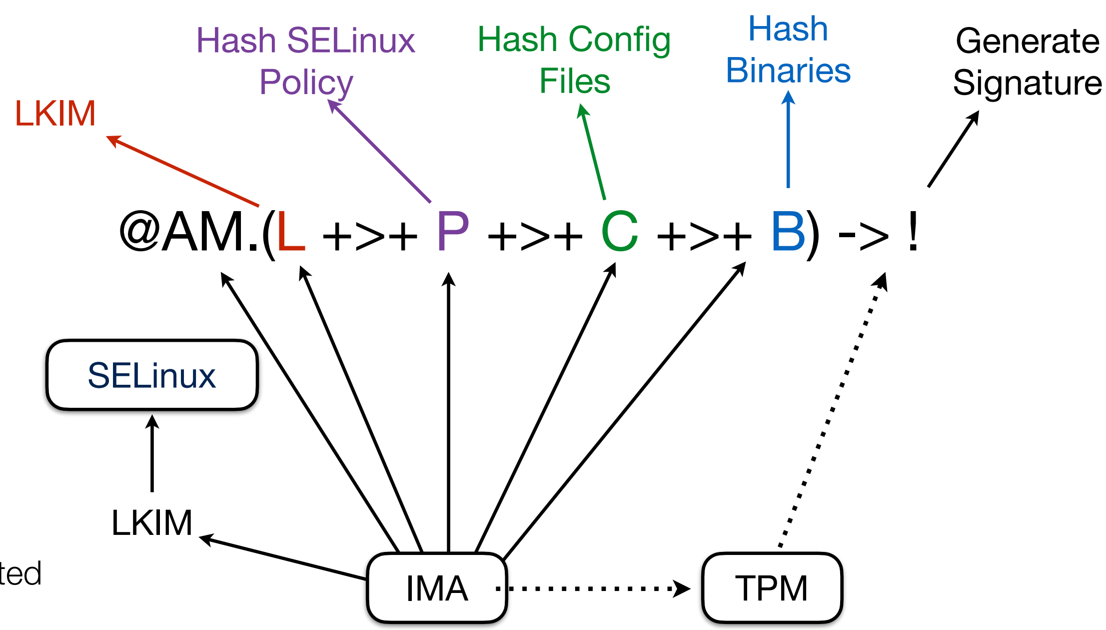
ASPs and Protocol



Protecting Attestation at Runtime

- Runtime IMA Measurements
  - Policy specifies hashes for ASPs
  - Policy specifies a hash for AM
  - IMA writes log to TPM PCR 10 (currently unused)
- AM^{-1} Signature
  - key is TPM resident
  - SELinux controls access to key blob
  - IMA EVM controls key blob permissions
- Linux
  - measured during boot using Invary LKIM
  - remeasured at runtime using Invary LKIM
  - SELinux policy dumped and hashed
  - good signature memorializes boot
  - the AM’s key is not available if boot policy is violated

Signature snaps runtime and boot trust together



Appraising Attestation Results

- Trustworthy target if
  - proper bundling
  - evidence satisfies appraisal policy
  - valid signature
- Proper bundling
  - indicates measurement ordering
  - generated by verified AM
- Satisfies appraisal policy
  - E_L - LKIM policy appraisal
  - E_{P-B} - Hashes checked against golden values
  - AM^{-1} - Signature checked with public AM key
- Provisioning requirements
  - gather good hashes
  - generate and distribute AM key pair
  - define LKIM appraisal policy

@AM.(L +>+ P +>+ C +>+ B) -> !

proper bundling

valid signature

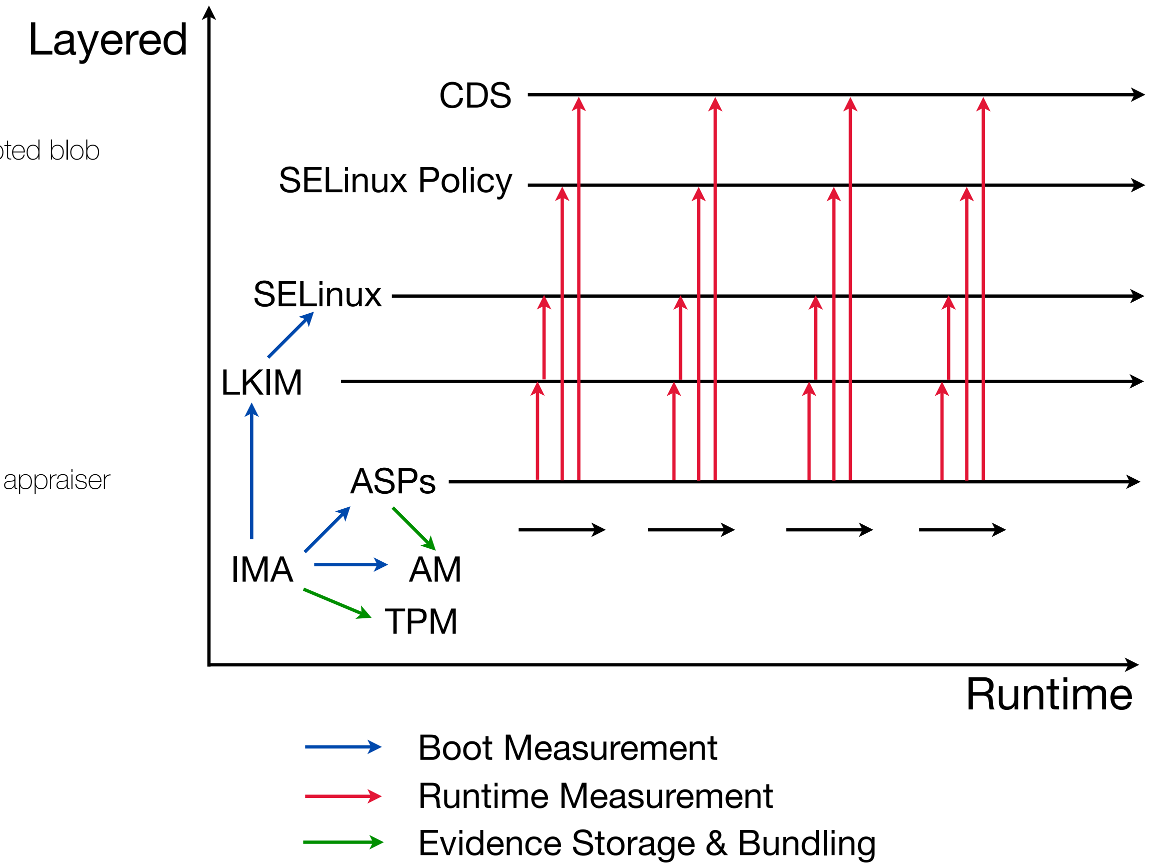
$\{ E_L ;; E_P ;; E_C ;; E_B \}_{AM^{-1}}$

good LKIM evidence ∧ good hashes

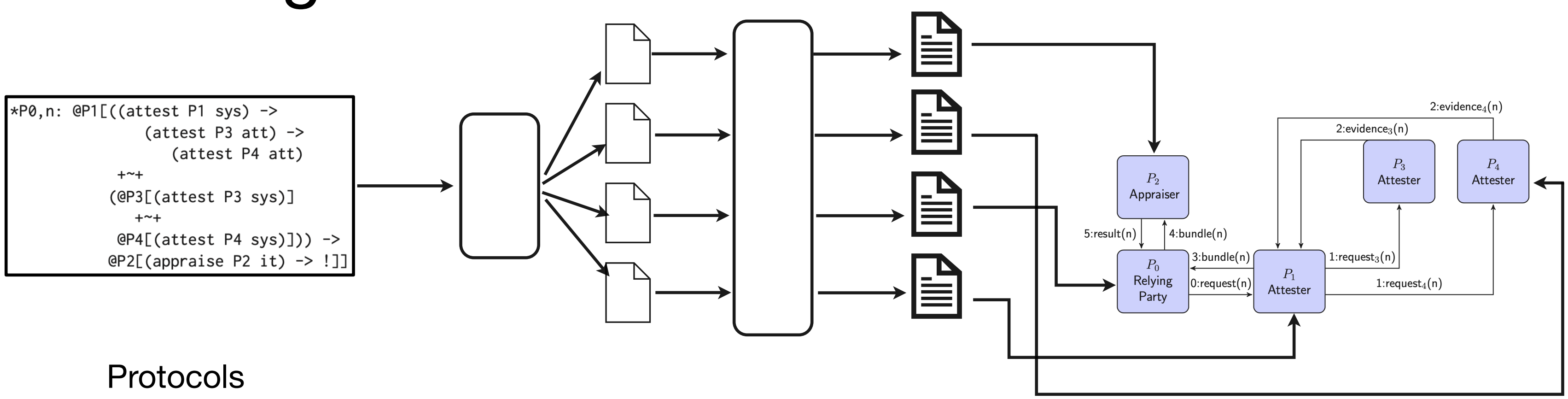
$$\left.\begin{array}{l}\text{proper bundling}\\ \wedge\ \text{satisfies appraisal policy}\\ \wedge\ \text{valid signature}\end{array}\right\} \Rightarrow \text{trustworthy target}$$

Layered Runtime Attestation

- Boot to an initial measured state
  - establish running AM with bound key
  - IMA hashes and checks AM on invocation
  - AM^{-1} is available on good PCRs, good AM and encrypted blob
- Remeasure at runtime
  - AM executes Copland attestation protocols
  - ASPs gather information after IMA check by IMA
  - Protocol execution bundles evidence
  - AM signs gathered evidence with AM^{-1}
- Appraisal and Remeasurement
  - AM communicates with relying party
  - Appraisal may occur in AM, Relying Party, or third party appraiser
  - Remeasurement may occur in AM or Relying party
- PCRs are the trust link
  - boot measured into PCRs
  - signing key sealed by PCRs
  - signature carries trust meta-evidence
- Layering builds trust bottom up
  - dependencies measured first
  - bundled evidence reflects measurement order
  - verified in earlier work



Synthesizing Attestation Infrastructure



- Protocol
  - user writes a Copland protocol identifying places and resources
  - evaluating various flexible mechanisms
- Manifest Generator
  - automatically generate manifests for attestation managers
  - formally verified to preserve semantics
- Manifest Compiler
  - automatically generate configurations for verified attestation manager
  - formally verified to preserve semantics
- Attestation Test Bed
  - controlled evaluation environment
  - mixed architecture - ARM, Intel, IoT, Xen, KVM

Attestation infrastructure is simpler to verify than the attestation target

Adversary Goals and Attack Mechanisms

The adversary's primary goal is convincing a relying party to trust something it should not

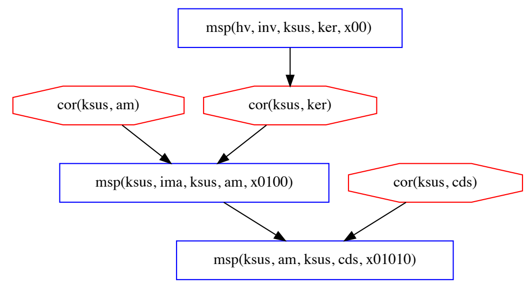
The adversary's secondary goal is convincing a relying party not to trust something it should

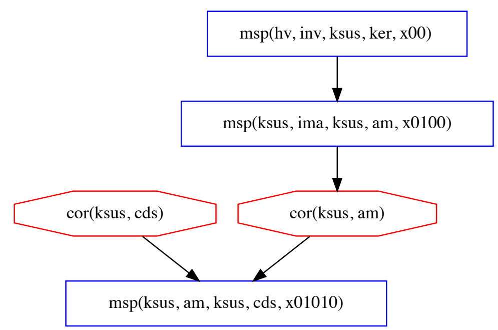
- Attacks on attestation target
  - change target without impacting policy compliance
  - change target and repair before measurement (TOCTAU)
- Attacks on evidence and meta-evidence
  - post measurement changes directly to evidence
  - generate signatures using incorrect components
  - cache alterations and poisoning
  - evidence package replay and spoofing
- Attacks on attestation infrastructure
  - compromise AM identity and steal AM's signing key
  - compromise AM execution and ASP ordering
  - alter ASPs to report incorrect, but compliant evidence
  - attack crypto and attestation protocol infrastructure
  - incorrectly report appraisal results
- Attacks on system infrastructure
  - compromises to hardware
  - changing boot images and boot order
  - TPM, IMA, and SELinux policy modifications

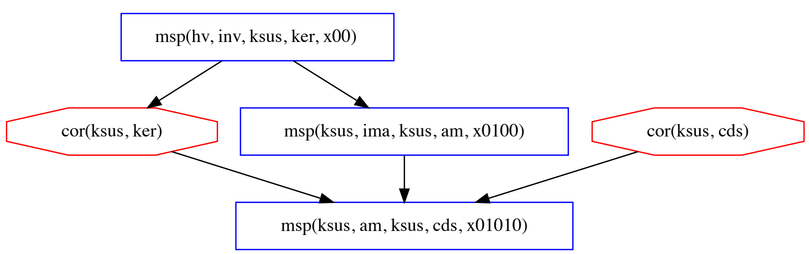
Attack Generation and Testing

- Generate attacks from CHASE outputs
  - CHASE generates all models allowed by a constraint set
  - specialized to generate all allowed attack graphs for a Copland protocol
  - use attack graphs for generating actual attacks on implementations
- Implementing tradeoff studies
  - deep vs shallow attestation implementations
  - caching measurements of deep components
  - tradeoff costs and time vs attack detection
- Protocol ordering
  - formally comparing protocols continuing
  - refinement of the “stronger” concept with utility of evidence
  - heuristics implemented in automated lint-like tools

Attack graphs define event orderings in successful attacks







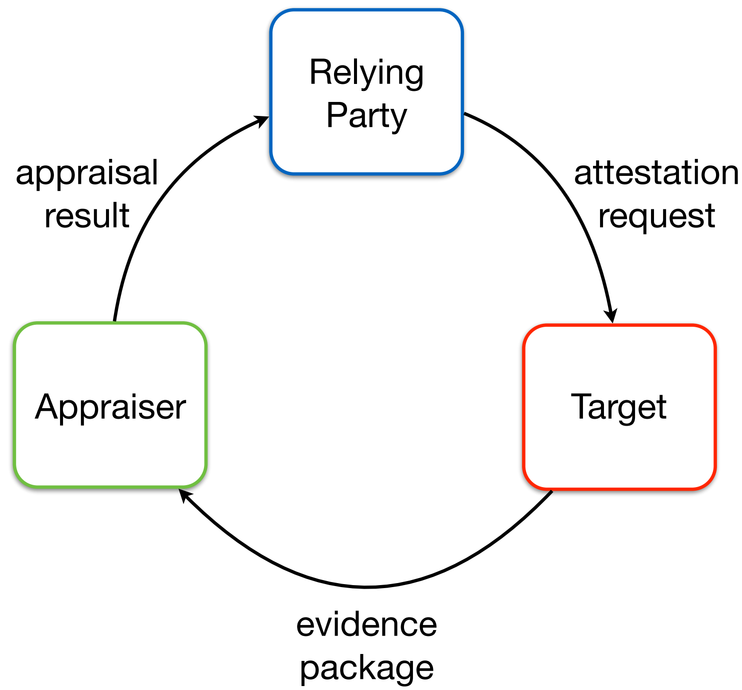
Testing Results

- Components targeted in testing
  - boot measurement infrastructure
  - runtime measurement infrastructure
  - CDS system configuration and components
- Attacks on configurations
  - altering component configuration
  - changing SELinux, IMA and TPM policy
- Attacks on executables
  - changing component runtime behavior
  - replacing or modifying executables
- Attacks across lifecycle
  - boot time attacks
  - runtime attacks
  - transitioning from boot trust to runtime trust

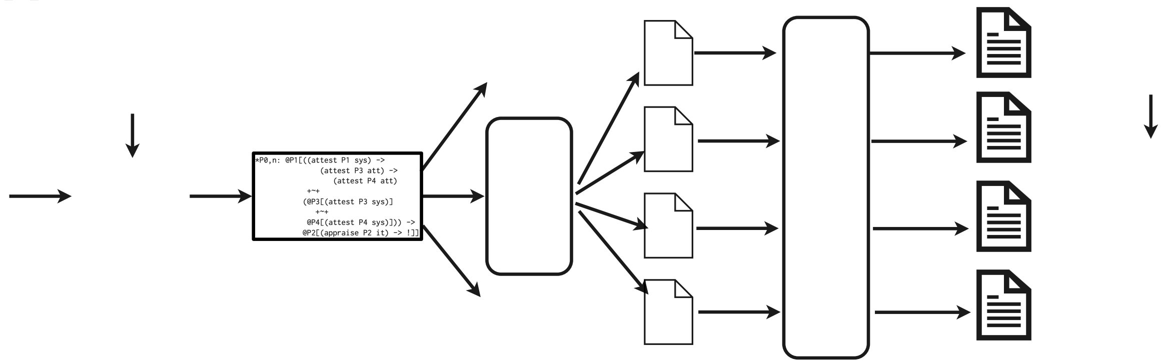
Attacks Considered		
Component	Configuration	Executable
Hardware	✘	✘
TPM	✔	✘
Bootloader	✔	✘
LKIM	✔	✘
Kernel	✔	✔
IMA	✔	✔
SELinux	✔	✔
AMs	✔	✔
ASPs	✔	✔
CDS Comp	✔	✔

What We Learned

- Boot transition to runtime is messy
  - boot trust must be reflected in runtime appraisal
  - yet there is no moment when runtime starts
  - integration with low level apparatus helps (IMA, SELinux, TPM)
- The AM's signing key is critical
  - a good AM key signature memorializes trusted boot
  - AM key compromise invalidates all attestation results
  - the AM key is long-lived and difficult to protect
- Design for attestation
  - short lived processes are more difficult to attack
  - processes run only when needed
  - dependencies first and layering is essential
  - separate infrastructure from application
- M&A must be easier to verify than its target
  - an attestation system is simpler than its target
  - managers, ASPs, policies are reusable
  - boot to a good attestation manager is reusable



Next Up…



- Long-running attestation
  - re-measurement intervals
  - evidence caching and behavior
  - evidence behavior over time
- Larger layered targets
  - multi-machine attestations and appraisal
  - evidence bundling and abstraction
  - external appraisal services
- Evidence as program understanding
  - formal notions of measurement and abstraction
  - temporal evidence properties
  - composition evidence properties
- Protocols From Systems
  - move the user from protocol authoring to system modeling
  - generate protocols from system models
  - include adversary models
- Put Evidence Semantics to Work
  - linter to provide protocol writing guidance
  - type analysis to predict protocol behavior
  - understanding protocol orderings
- Separation issues in AM and ASPs
  - compartmentalization of ASP execution
  - separation within the AM
  - verus modeling for ASPs

People and Publications

- Perry Alexander - KU
  - palexand@ku.edu
- Adam Petz - KU
  - ampetz@ku.edu
- Will Thomas - KU
  - 30wthomas@ku.edu
- Logan Schmalz - KU
  - loganschmalz@ku.edu
- Sarah Johnson - KU
  - sarahjohnson@ku.edu
- Joshua Guttman - MITRE
  - guttman@mitre.org
- Paul Rowe - MITRE
  - prowe@mitre.org
- James Carter - NSA
- Stephen Smalley - NSA
- Daniel DeGraff - NSA

- Petz, A., W. Thomas, A. Fritz, T. Barclay, L. Schmalz, and Perry Alexander, "Verified Configuration and Deployment of Layered Attestation Managers," *Proceedings of the 22nd International Conference on Software Engineering and Formal Methods (SEFM'24)*, LNCS 15280, November 4-8, 2024, Aveiro, Portugal.
- Petz, A., P. Alexander, "An Infrastructure for Faithful Execution of Remote Attestation Protocols," Proceedings of the NASA Formal Methods Symposium (NFM'21), May 24-28, Norfolk, VA.
- Johnson, S. and P. Alexander, "Ordering Attestation Protocols," n preparation for *Network and Distributed System Security (NDSS'26) Symposium*, San Diego, California, February 23-27, 2026.
- Thomas, W., L. Schmalz, A. Petz, P. Alexander, J. Guttman, "Layered Attestation in Action: Attesting to a Cross-Domain Solution," in preparation for *Network and Distributed System Security (NDSS'26) Symposium*, San Diego, California, February 23-27, 2026.