Lessons Learned Developing Cross-Domain Solutions on SELinux

2006 SELinux Symposium

Chad Sellers, Karl MacMillan, Spencer Shimko, Frank Mayer, and Art Wilson

Tresys Technology, USA

TRESYS TECHNOLOGY

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Cross-Domain Solutions

- What is a CDS?
  - Controlled channel for data transfer across security boundaries
    - e.g. bridging Top Secret and Secret networks
  - Can provide confidentiality and/or integrity protection
- Why do we need CDS?
  - System-high computing environments
    - all data is treated at the network level
    - e.g. Secret data on Top Secret network
  - Untrusted software
    - e.g. Windows
    - can't trust this software to make security decisions

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

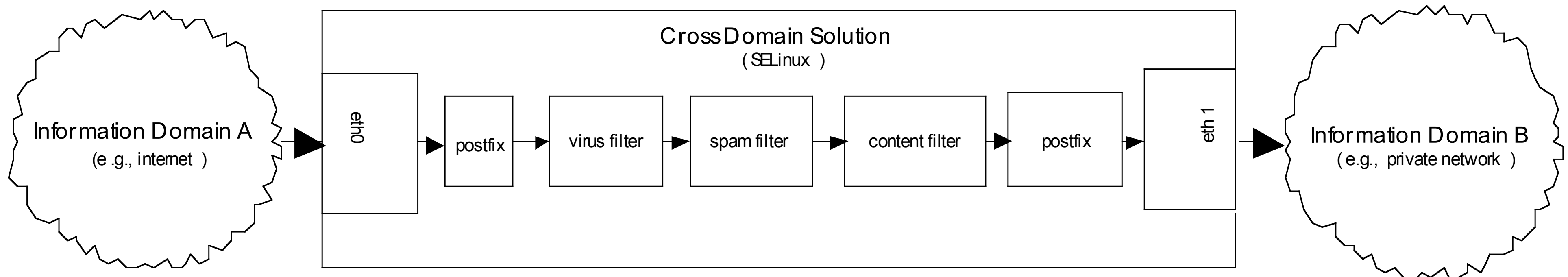
Primary Security Goals

- Prevent unwanted disclosure
  - Goal - to keep secrets secret
  - e.g. high-to-low document transfer
    - documents A and B on Secret system
    - A is Secret and B is Unclassified
    - CDS connects Secret network to Unclassified network
    - goal is to keep A from reaching the Unclassified network
- Ensure network integrity
  - Goal - protect from those less trusted
  - Opposite direction from previous goal
  - e.g. keeping viruses off of an internal network

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

CDS Architecture

- Information flow pipeline
  - Each stage responsible for a single security goal
  - OS responsible for confining data to the pipeline



© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Type Enforcement in CDS

- BLP/Biba MLS models insufficient
  - The purpose of a CDS is to break these models
  - These models are too limited to meet architecture goals
    - exceptions to policy necessary for functionality
    - ill-suited to processing pipelines
    - lack granularity for system self-protection
- TE is well-suited to meeting architecture goals
  - Policy governs entire system
  - Scalable domain separation
  - Lends itself toward specification of processing pipelines
  - Self-protection as implemented in current policies
  - Fine-grained least privilege administration

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Practical Approach to SELinux CDS

- Start with Red Hat Enterprise Linux 4
  - Good choice for two primary reasons
    - support
    - certification activities
  - Minimize set of applications and services
  - Strict NSA example policy or Reference Policy as base
- Customize policy for particular CDS
  - Remove unneeded modules
    - e.g. hopefully remove X server
  - Tighten modules as needed
  - Implement new modules for CDS applications

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Addressing Policy Completeness

- Let SELinux do the heavy lifting
  - This is the goal
- Transfer policy enforcement infeasible
  - e.g. “document files cannot have embedded executables”
  - Filter applications must enforce transfer policy
  - SELinux policy can only ensure
    - data travels through those filters
    - each filter is only trusted to ensure its single security goal
- Maximizing decomposition into separate processes
  - Minimize trust placed in an individual filter
  - Maximize enforcement burden placed on SELinux

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Addressing Policy Completeness

- Extra information flow paths
  - Real world policies will have extra paths
    - outside the primary data transfer path
    - for initialization, management, etc.
  - The processes on these extra paths must be trusted
    - not to pass data around the primary pipeline
    - to limit channels other processes could take advantage of
- Limiting extra information flow paths
  - Bandwidth limiting IPC
  - One-way flow mechanisms
  - System state limitations

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

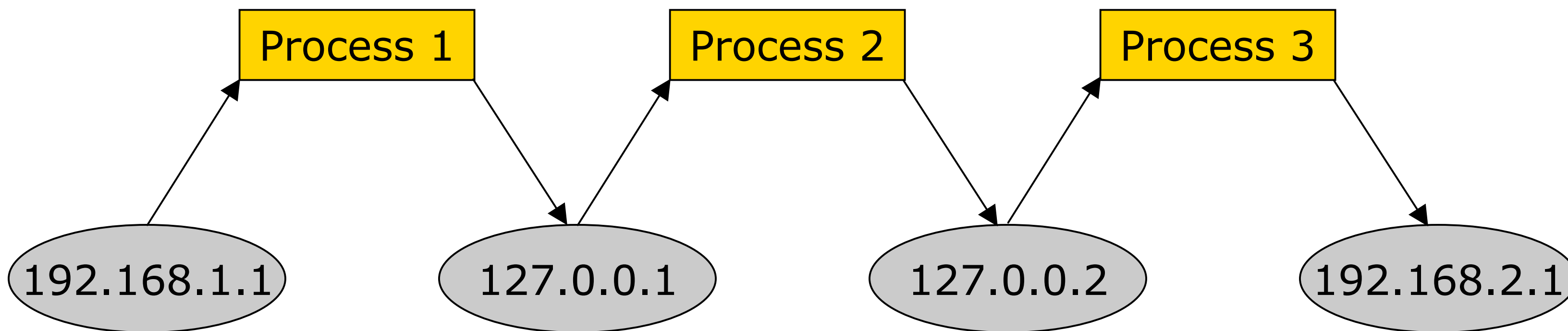
Separating Information Domains

- Must control access to networks
  - i.e. must ensure data cannot pass between the networks
  - This is the first step
- Many current policies' network access controls
  - Coarse grained
  - Lumps interfaces and nodes together
- Problem software
  - Assumes all networking is always available
    - may not fail gracefully
    - e.g. attempt to bind to all interfaces and bail if this fails
  - Uses network for IPC
    - e.g. Java RMI

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Separating Information Domains

- Addressing problem software
  - Utilize localhost IP aliasing and node separation
    - create many IP aliases for IPC resources
    - label them individually and limit access to them
  - Example

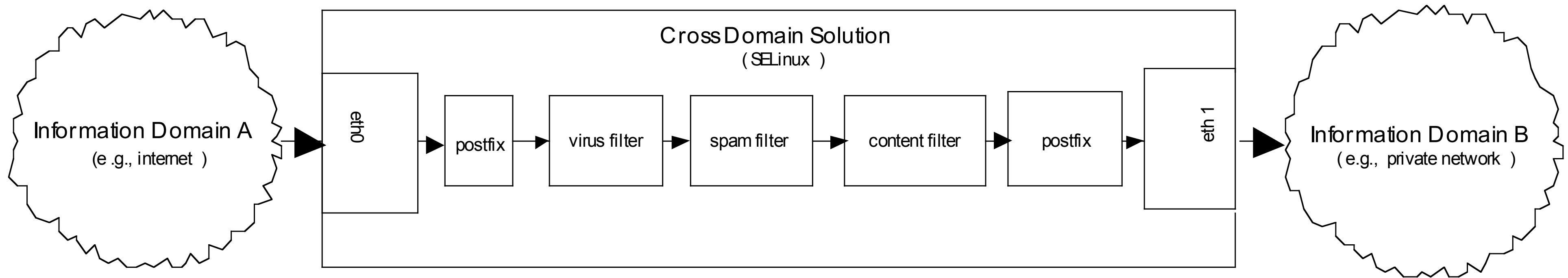


- Some software will require modification

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Non-bypassable processing pipelines

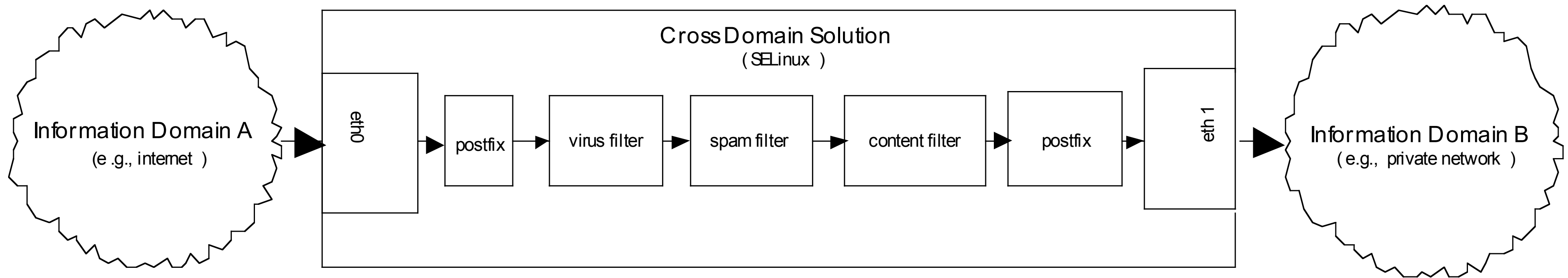
- Remember the information flow pipeline



© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Non-bypassable processing pipelines

- Remember the information flow pipeline



  - TE domain types for each step
  - IPC mechanism defined for domain interaction
  - Only communications with adjacent domains allowed
- Requirements for building this on real systems?
  - One-way IPC mechanisms
  - Precise policies specifying flow

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

One-way IPC

- Many IPC mechanisms inherently bidirectional
  - TCP, UNIX stream sockets require response messages
  - Pipes provide timing back channels
  - etc.
- Existing software designed for bidirectional IPC
  - Utilized bidirectional protocols
    - TCP
    - HTTP
  - Built on higher level constructs requiring bidirectionality
    - Java RMI on TCP
    - SOAP on HTTP

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

One-way IPC - Policy Reuse

- Policies exist for many components
  - ssh, postfix, etc.
  - policies contain deep knowledge of the component
- Policies usually not focused on information flow
  - usually focused on least privilege or on functionality
  - often use excessive macros (e.g. can_network)
  - require major overhaul to enforce information flow
    - go through all reused policies
    - look for excessive macro use or allow rules
    - rewrite to avoid these but maintain functionality

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

System self-protection

- Aligned with many current policies
  - Protecting system resources common goal to most systems
    - kernel
    - policy
- Room for improvement
  - Functionality compromises have been made
  - Those compromises may not be acceptable for you
- Bottom line
  - Understand what the base policy does
  - Minimal changes may be necessary

TRESYS TECHNOLOGY

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Least-privilege administration

- Very difficult with example policy
  - Splitting up sysadm_r requires complete policy audit
    - look at all accesses for all types authorized for sysadm_r
    - move some types to appropriate new roles
    - split other types where access splits new role boundary
  - Modifying roles requires repeating this process
  - Deployment variations require complete policy rework
    - i.e. too difficult to modify roles for a customer environment
- Reference policy working on a solution
  - Interfaces make this possible
  - Utilize role dominance mechanism

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Conclusion

- Type Enforcement is an excellent mechanism for CDS
  - Lends itself to creating assured pipelines
    - removes trust from applications
    - OS controls where data flows
  - Means of expressing many different security goals
    - self-protection, least privilege, domain separation, etc.
- SELinux provides strong foundation for development
  - Fine-grained specification of policy
  - Previous base and application policies to work from
    - no need to rework the wheel
- Many practical issues to overcome

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

QUESTIONS?

© 2006 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)