A Security Engineering Type Enforcement Framework for Cross Domain Solutions

Art Wilson
Tresys Technology

www.tresys.com
What is the Problem?

• Cross Domain Solutions (CDS) are often built with insufficient security engineering
• CDS developers focus on application issues, not infrastructure
• CDS certification and accreditation accounts for too much of the lifecycle cost
What Do We Want To Do?

• Build more secure systems
  – By enabling designers to more easily define and evaluate security properties

• Reduce cost
  – By reducing the time to design and develop infrastructure
  – By producing verification evidence as part of the design process
  – Through reuse
The CDS Framework Toolkit Vision

- Enable the application of type enforcement (TE) as a better mandatory access control (MAC) mechanism for cross domain solutions (CDS)
  - Enable and enforce good security engineering for CDS
  - Facilitate implementation of the infrastructure
  - Support certification and accreditation
  - Allow developers to focus development efforts on filters

- Build better, more easily certifiable and accreditable CDS in less time
What is a Cross Domain Solution?

• An information assurance solution that provides the ability to manually and/or automatically access and/or transfer between two or more differing security domains.

  Chairman of the Joint Chiefs of Staff Instruction 6211.02B
Why TE is Appropriate for CDS

- Type enforcement provides flexible mandatory access control
- Allows developers to encode security properties appropriate to the cross domain solution problem
  - Unlike rigid hierarchical MAC mechanisms
- Provides extremely granular control over information flow
  - But complex policies are challenging to develop, manage and validate!
What is the Security Engineering Framework Toolkit?

- **Security Engineering Framework**
  - Abstraction to capture security policy in simple components
  - Decomposition to manage complexity

- **Security Engineering Toolkit**
  - Provides a manageable user interface
  - Enforces security engineering discipline
  - Automates implementation of TE policy
  - Provides evidence supporting certification and accreditation
High Level View of a Sample CDS

Low Network → Cross Domain Solution → High Network
Framework Representation of Sample CDS

- Boxes = Process Domains
- Circles = Shared Resources
- Arrows = Access Arrows

Low Network

Low Domain  

Guard Domain  

High Domain

High Network
Security Engineering Framework
Components

• Process Domains
  – Subjects, represented by a box
  – Contains an active process and associated resources
    • Access is restricted within the process domain

• Shared Resources
  – Objects, represented by a circle
  – Passive entity, no processing
    • e.g., file, directory, pipe, etc.
  – The only way for process domains to communicate

• Access arrows
  – Define information flow between process domains and shared resources
Decomposition in the Security Engineering Framework

Low Domain → Guard Domain → High Domain

- Low Domain
- Virus Filter Domain
- Key Word Filter Domain
- High Domain

Computer Platform
Decomposition in the Security Engineering Framework
CDS Framework Demo System

Guard Process Domain

Virus Filter
guard__virus

Human Review Filter
guard__hrev

Router Filter
guard__router

ASCII Filter
guard__ascii

Dirty Word Filter
guard__dirtyw

Audit

Quarantine

In Process Domain
(Contains FTP Daemon and Eth1)

Out Process Domain
(Contains FTP Daemon and Eth2)
Framework Denotation

- One-to-one correspondence between components and rules
- These 35 lines define the example CDS
Framework Expansion

- 35 lines expand to about 600 TE rules specific to the example CDS
- Overall policy contains thousands of TE rules
Errant Router Filter

In Process Domain (Contains FTP Daemon and Eth1)

Guard Process Domain

Errant Router Filter
guard__router

Virus Filter
guard__virus

Human Review Filter
guard__hrev

ASCII Filter
guard__ascii

Dirty Word Filter
guard__dirtyw

Audit

Out Process Domain (Contains FTP Daemon and Eth2)
Malicious Router Filter

In Process Domain (Contains FTP Daemon and Eth1)

Guard Process Domain

1. Virus Filter
   guard__virus

2. ASCII Filter
   guard__ascii

3. Malicious Router Filter
   guard__router

4. Human Review Filter
   guard__hrev

5. Dirty Word Filter
   guard__dirtyw

Out Process Domain (Contains FTP Daemon and Eth2)

Audit

*/etc/shadow*
Benefits of the Security Engineering Framework

- Applies to more than CDS
- Abstracts security policy in simple components
- Provides decomposition to manage complexity
- Consistently represents concepts
- Supports design implementation and validation
- Enforces good security engineering
Benefits of the Security Engineering Framework Toolkit

• Provides a GUI for using the Framework
• Eliminates the need for engineers to understand type enforcement details
• Enforces process domain constraints in type enforcement policy
• Defines properties of the actual solution environment
• Automates type enforcement policy generation
• Provides certification and accreditation evidence
Why Does the Security Engineering Framework Toolkit Matter?

• Makes granular type enforcement accessible and verifiable
• Enforces good security engineering in an accessible way to make building more secure systems easier!
For More on Security Enhanced Linux

- www.nsa.gov/selinux
- www.tresys.com/selinux
- www.selinux-symposium.org
Questions?

Art Wilson

awilson@tresys.com 410-290-1411x104

Tresys Technology
8840 Stanford Boulevard
Suite 2100
Columbia, MD 21045

www.tresys.com
Backup
What is Type Enforcement?

• Typing
  – Processes and resources
• Access decision computation
  – Lookup within an access matrix
• All relationships between types must be explicitly defined in the access matrix
What is Mandatory Access Control?

- The need for a **mandatory access control** (MAC) mechanism arises when the security policy of a system dictates that:
  1. protection decisions must not be decided by the object owner.
  2. the system must enforce the protection decisions (i.e., the system enforces the security policy over the wishes or intentions of the object owner).