A Security Engineering Type Enforcement Framework for Cross Domain Solutions

Art Wilson
Tresys Technology

www.tresys.com

Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

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!

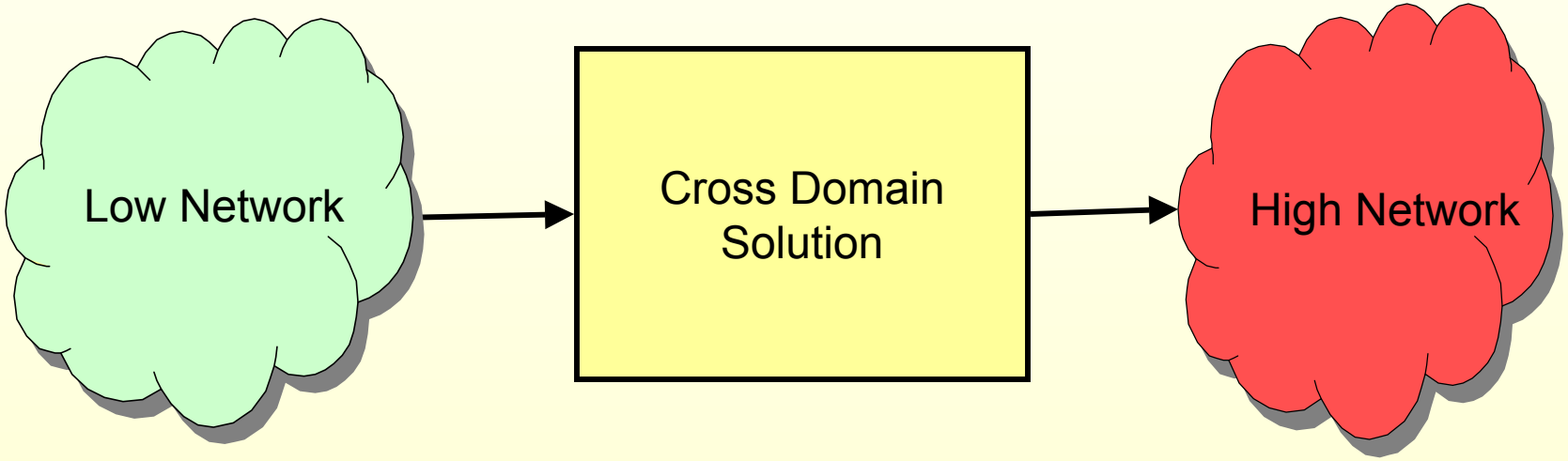
Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

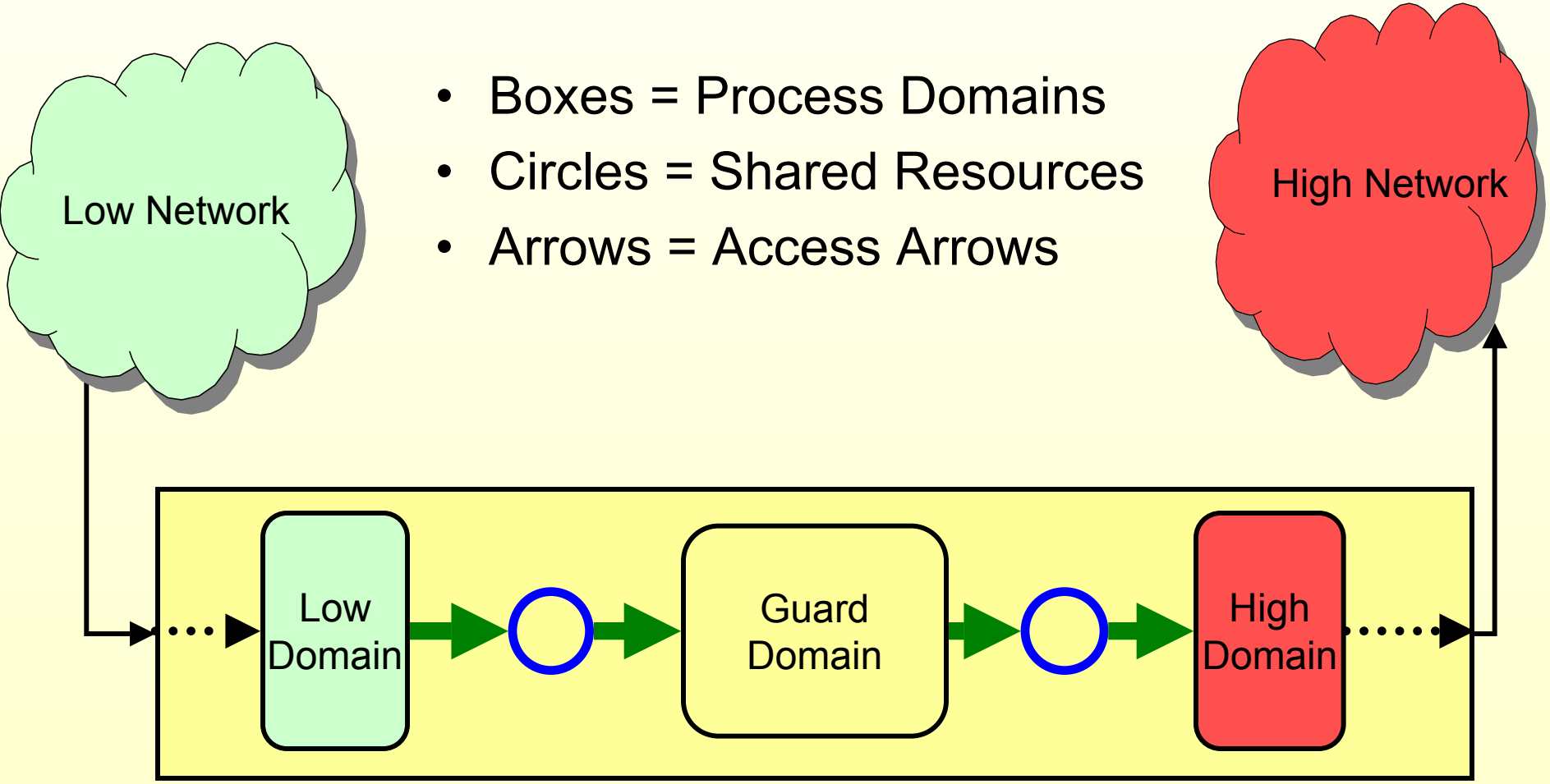
High Level View of a Sample CDS



Copyright (c) 2004 Tresys Technology, LLC

Framework Representation of Sample CDS

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



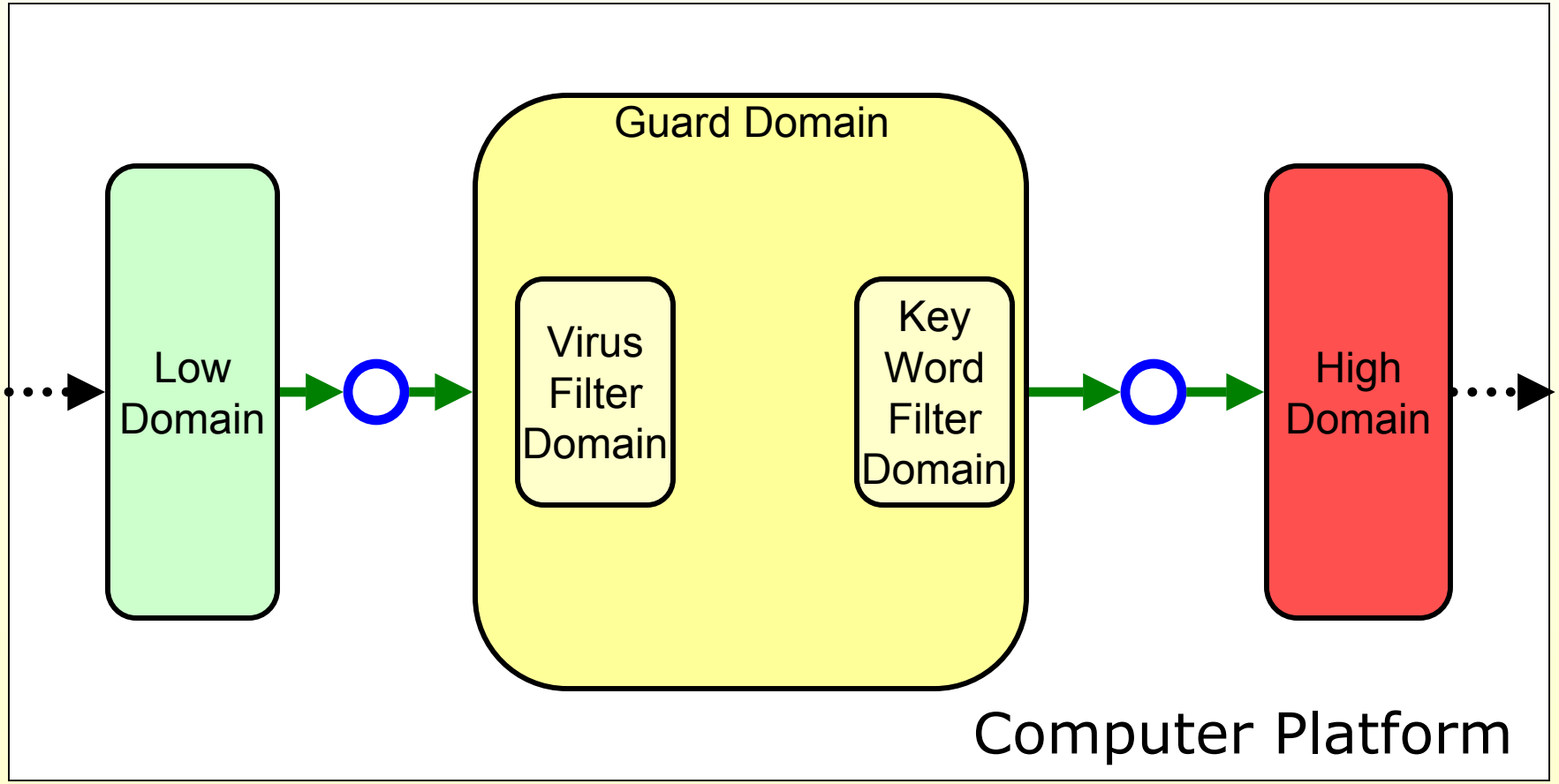
Copyright (c) 2004 Tresys Technology, LLC

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

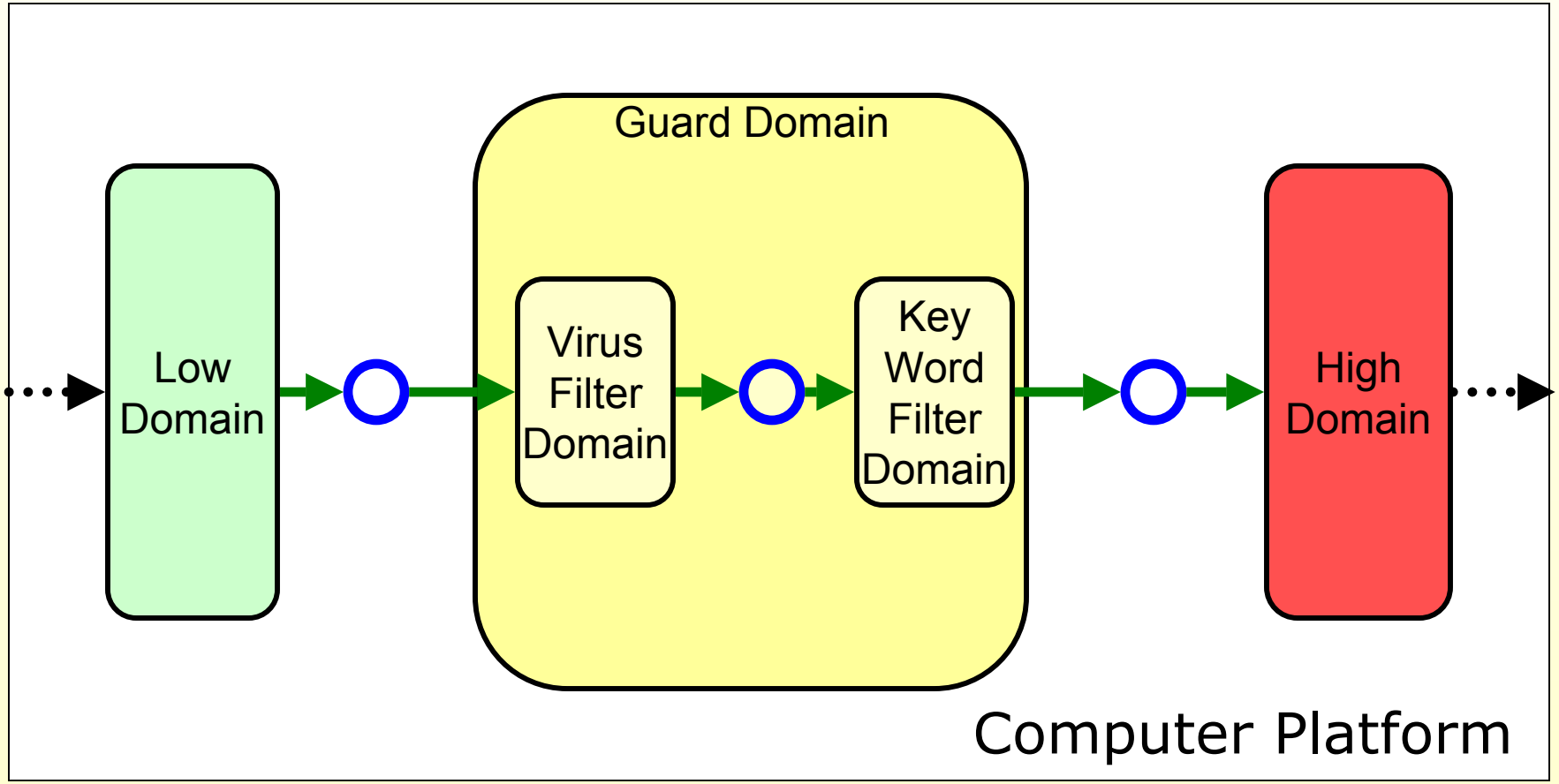
Copyright (c) 2004 Tresys Technology, LLC

Decomposition in the Security Engineering Framework



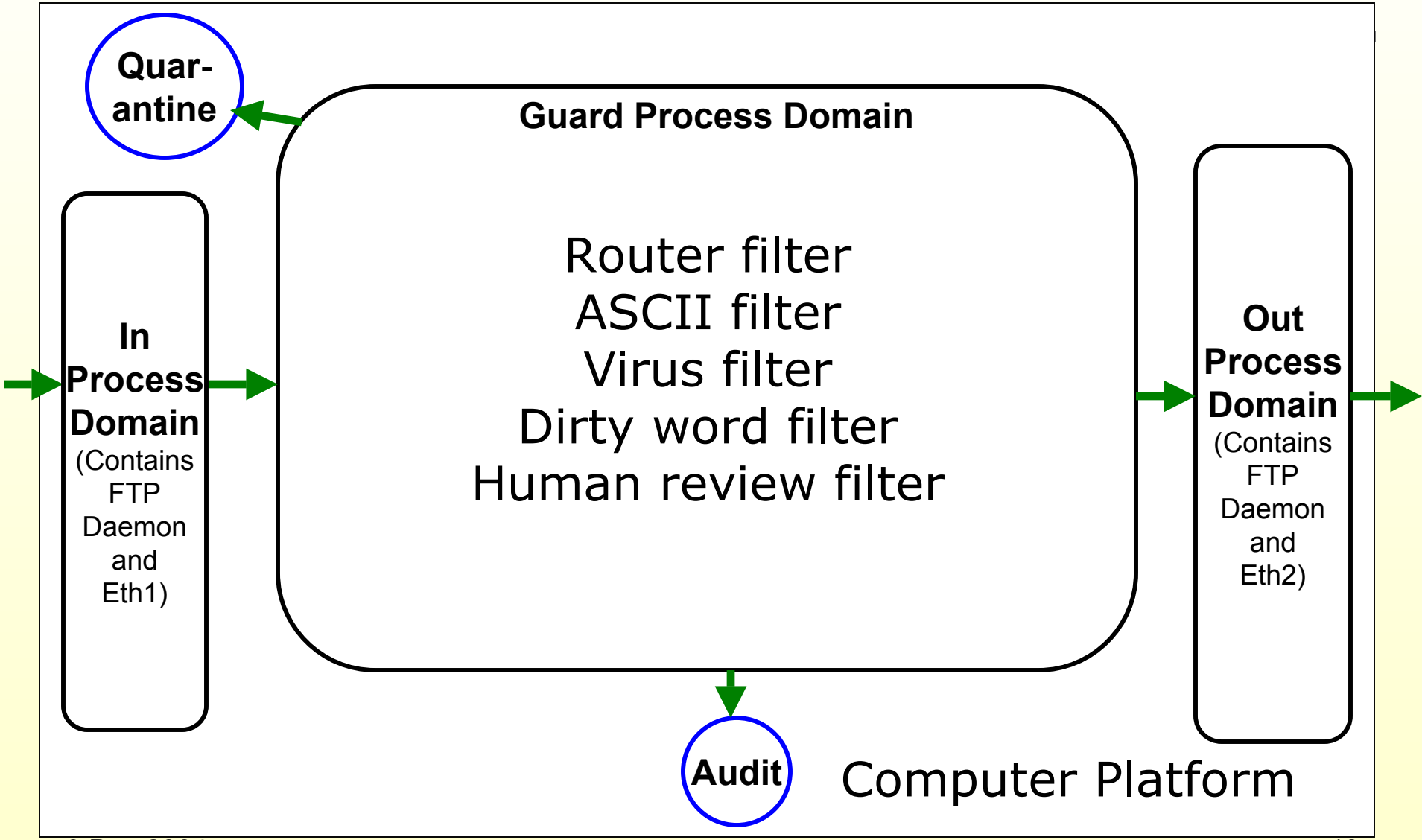
Copyright (c) 2004 Tresys Technology, LLC

Decomposition in the Security Engineering Framework



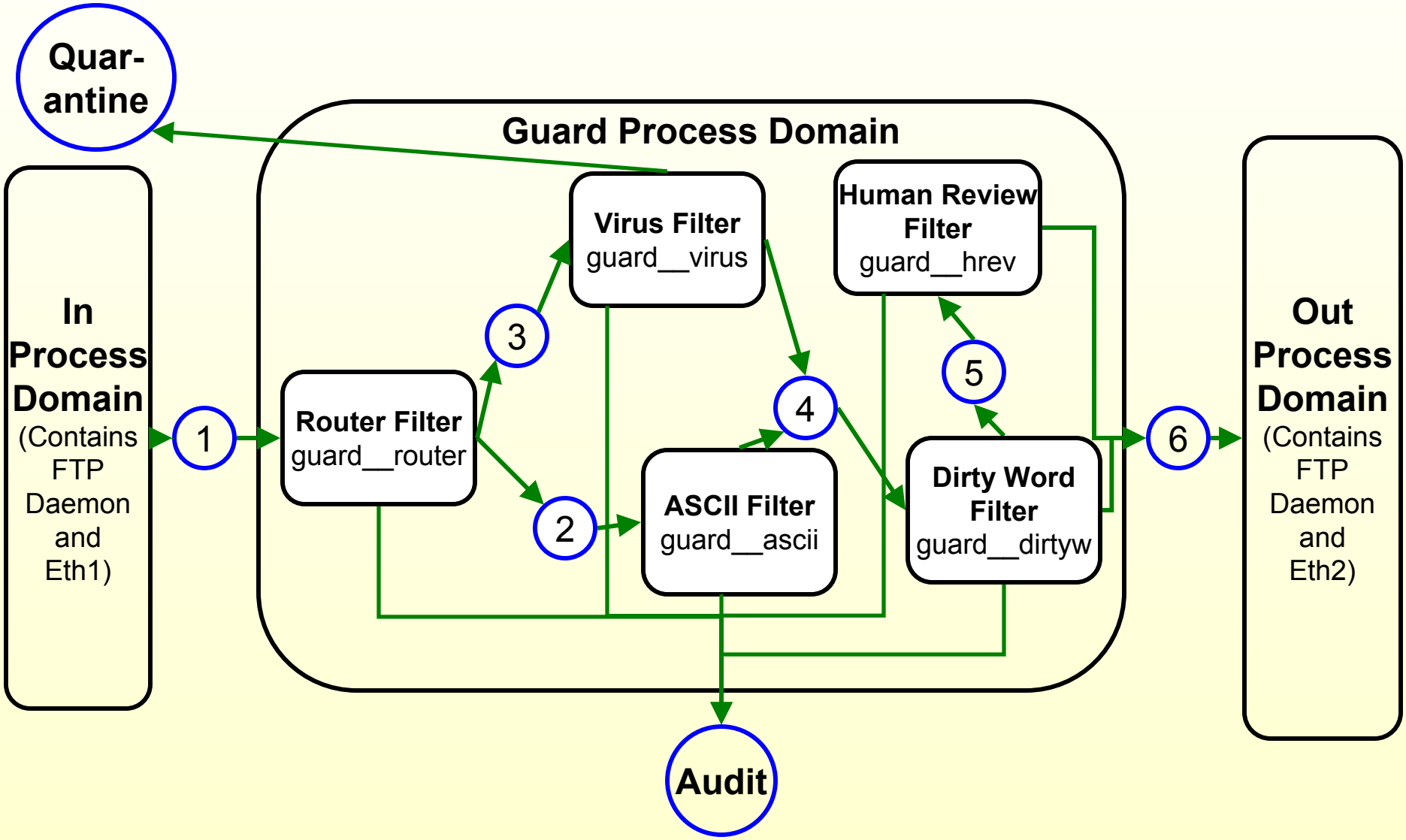
Copyright (c) 2004 Tresys Technology, LLC

CDS Framework Demo System



Copyright (c) 2004 Tresys Technology, LLC

CDS Framework Demo System



Framework Denotation

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

```
box_domain(in,file,pid,unix_stream,tcp_network)
box_domain(guard__router)
box_domain(guard__virus,file)
box_domain(guard__ascii,file)
box_domain(guard__dirtyw,file)
box_domain(guard__hrev,human)
box_domain(out,file,pid,unix_stream,tcp_network)

shared_resource(in_guard)
shared_resource(guard__router_ascii)
shared_resource(guard__router_virus)
shared_resource(guard__virus_ascii_dirtyw)
shared_resource(guard__dirtyw_hrev)
shared_resource(guard_out)
shared_resource(audit)
shared_resource(quarantine)

arrow(out,guard_out,file,read,unlink)
arrow(guard__router,in_guard,file,read,unlink)
arrow(guard__hrev,guard__dirtyw_hrev,file,read,unlink)
arrow(in,in_guard,file,write)
arrow(guard__dirtyw,guard__dirtyw_hrev,file,write)
arrow(guard__dirtyw,guard_out,file,write)
arrow(guard__hrev,guard_out,file,write)
arrow(guard__virus,quarantine,file,write)
arrow(guard__virus,guard__router_virus,pipe,read)
arrow(guard__ascii,guard__router_ascii,pipe,read)
arrow(guard__router,guard__router_virus,pipe,write)
arrow(guard__router,guard__router_ascii,pipe,write)
arrow(guard__dirtyw,guard__virus_ascii_dirtyw,unix_stream,read)
arrow(guard__ascii,guard__virus_ascii_dirtyw,unix_stream,write,guard__dirtyw)
arrow(guard__virus,guard__virus_ascii_dirtyw,unix_stream,write,guard__dirtyw)
arrow(guard__hrev,audit,file,append)
arrow(guard__dirtyw,audit,file,append)
arrow(guard__virus,audit,file,append)
arrow(guard__ascii,audit,file,append)
arrow(guard__router,audit,file,append)
```

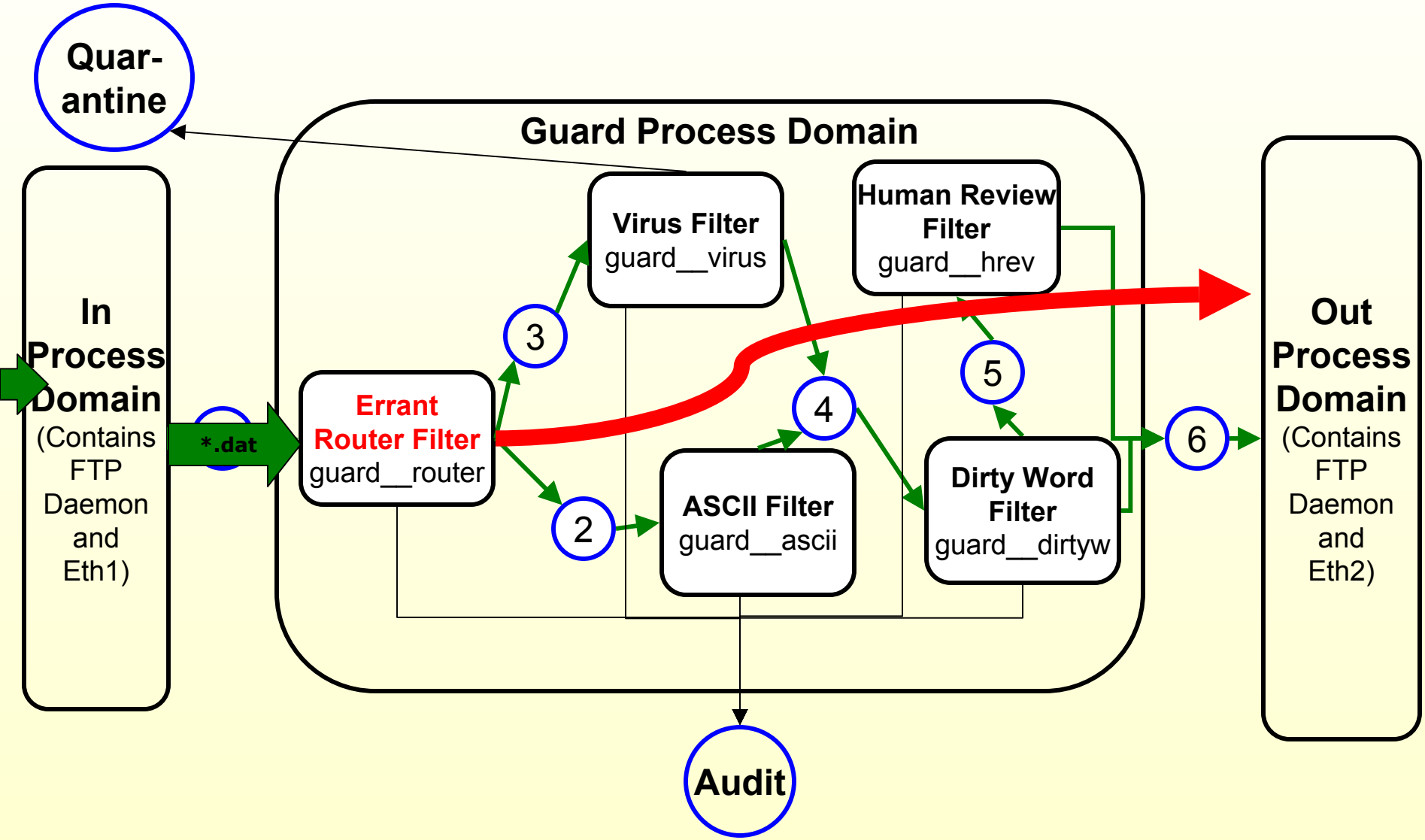

Copyright (c) 2004 Tresys Technology, LLC

Framework Expansion

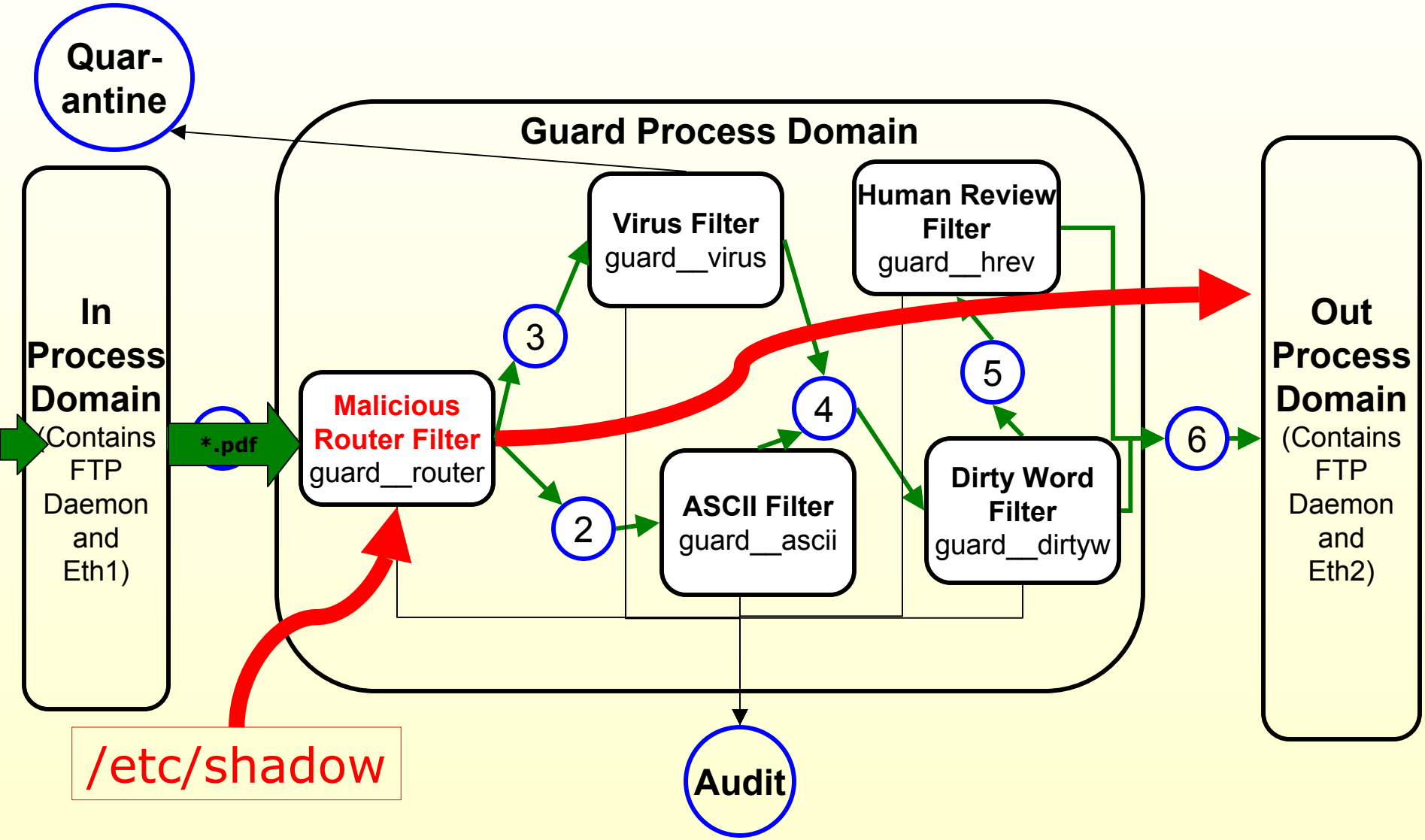
- 35 lines expand to about 600 TE rules specific to the example CDS
- Overall policy contains thousands of TE rules

Copyright (c) 2004 Tresys Technology, LLC

Errant Router Filter



Malicious Router Filter



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

Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

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!

Copyright (c) 2004 Tresys Technology, LLC

For More on Security Enhanced Linux

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

Copyright (c) 2004 Tresys Technology, LLC

Questions?

Art Wilson

awilson@tresys.com 410-290-1411x104

Tresys Technology
8840 Stanford Boulevard
Suite 2100
Columbia, MD 21045

www.tresys.com

Backup

Copyright (c) 2004 Tresys Technology, LLC

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

Copyright (c) 2004 Tresys Technology, LLC

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).

Copyright (c) 2004 Tresys Technology, LLC