Type Enforcement Access Control

Security Policy Issues with Security Enhanced Linux

Frank Mayer
Tresys Technology
mayerf@tresys.com

ACM Conference on Computer & Communications Security
Washington, DC
October 29, 2003

Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Topics

- Type Enforcement (TE) Overview
  - ala Security Enhanced Linux
- TE Policy Analysis
  - understanding complex TE policies
- TE Policy Development
  - evolving the tools and capabilities
- TE Policy Management
  - heading towards distributed, dynamic mandatory security

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

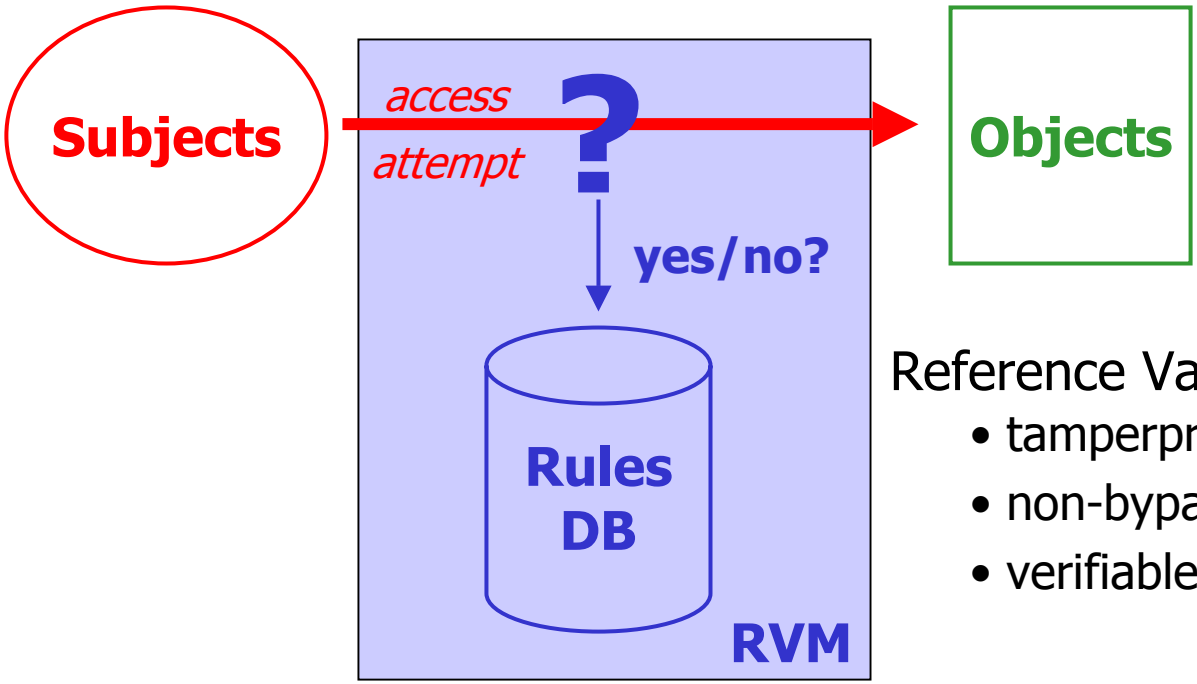
Type Enforcement Overview

As implemented in SE Linux

Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Reference Monitor Concept



Reference Validation Mechanism

- tamperproof
- non-bypassable
- verifiable

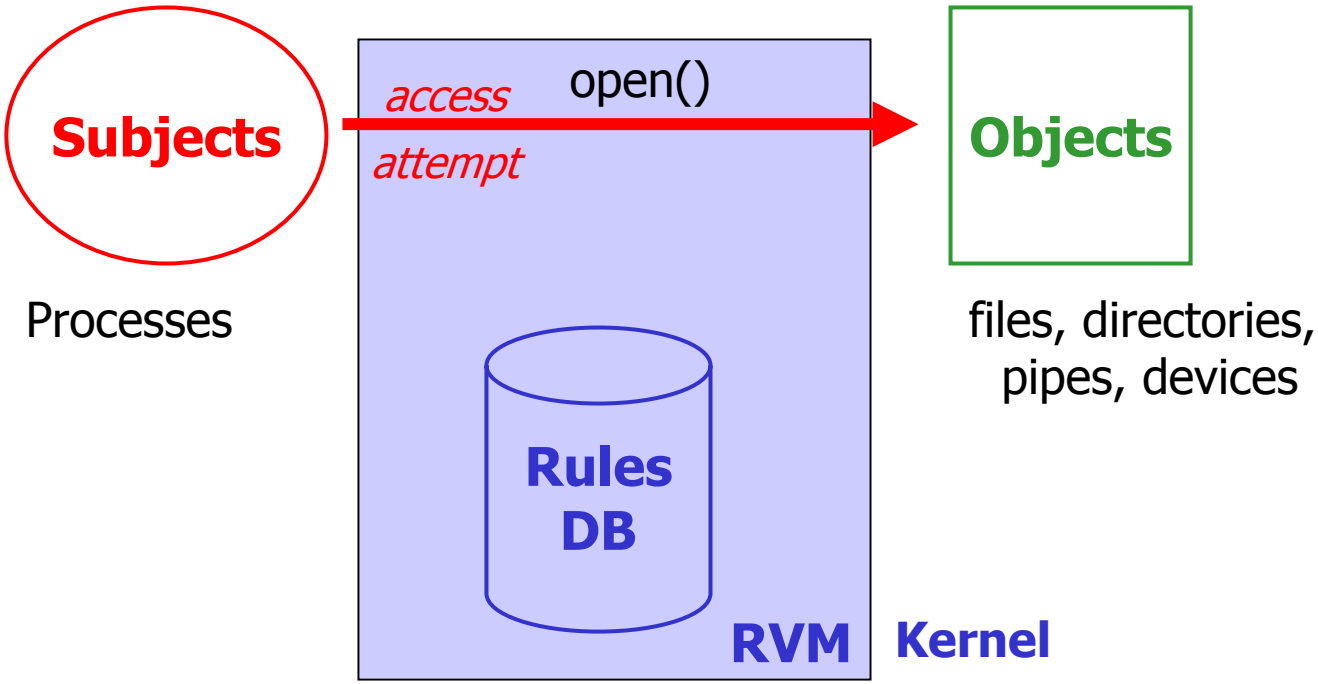
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Types of Access Control

- Access Control in general
  - subjects and objects have security attributes
  - access determined based on policy rules
- Discretionary Access Control
  - users can change security attributes at request
  - allowing programs running on behalf of a user to affect the results of access rules
- Mandatory Access Control
  - users cannot change security attributes at request (non-discretionary)
  - user programs must work within the constraints of access rules
  - MAC access rules are controlled by the organization, not the user

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Standard Linux Access Control



Fixed, hard-coded policy rules

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Standard Linux Access Control

- Processes
  - real and effective user and group IDs
- File system objects
  - access modes in inode

rwx r-x --- uid gid

not user or group permissions

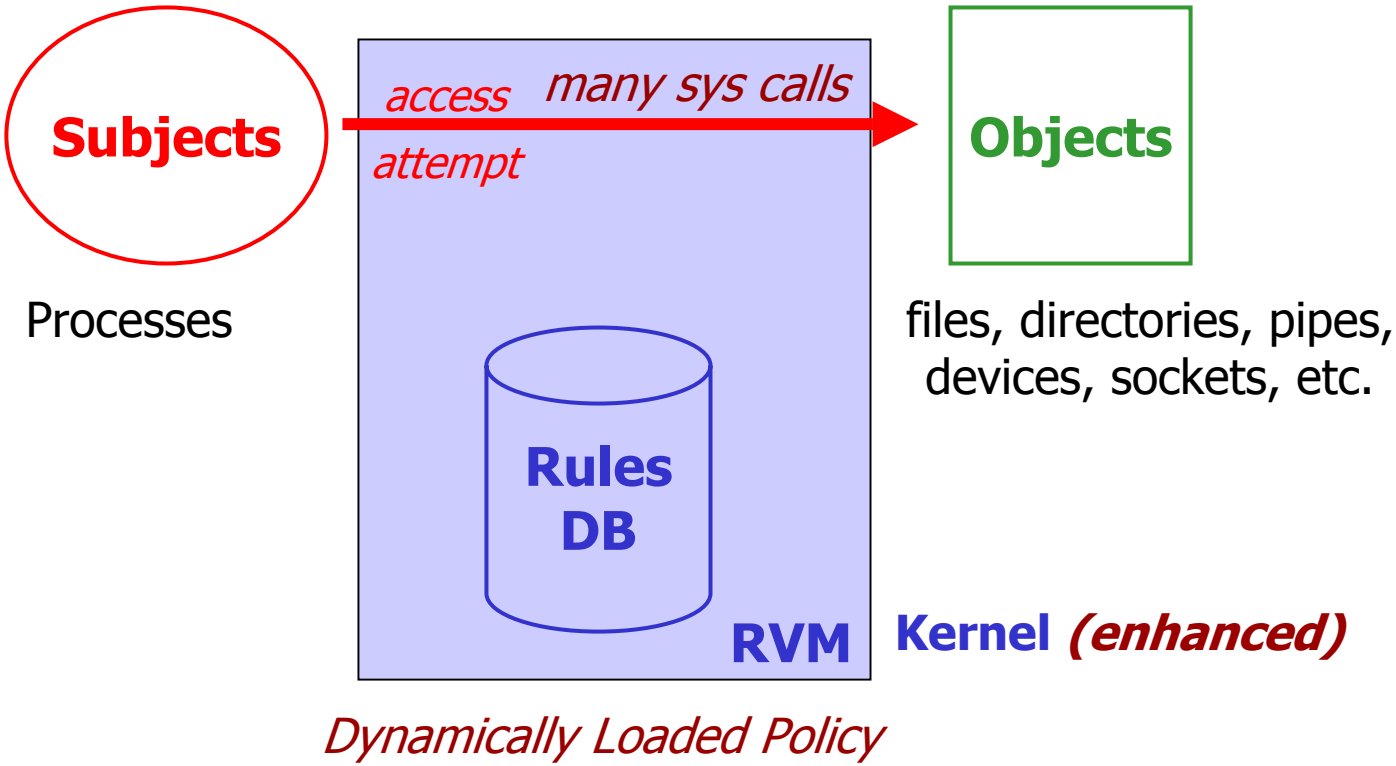
group, not user permissions

user permissions

- Access Rules: Fixed, hard-coded in the kernel
  - typically checked on open prior to descriptor create

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

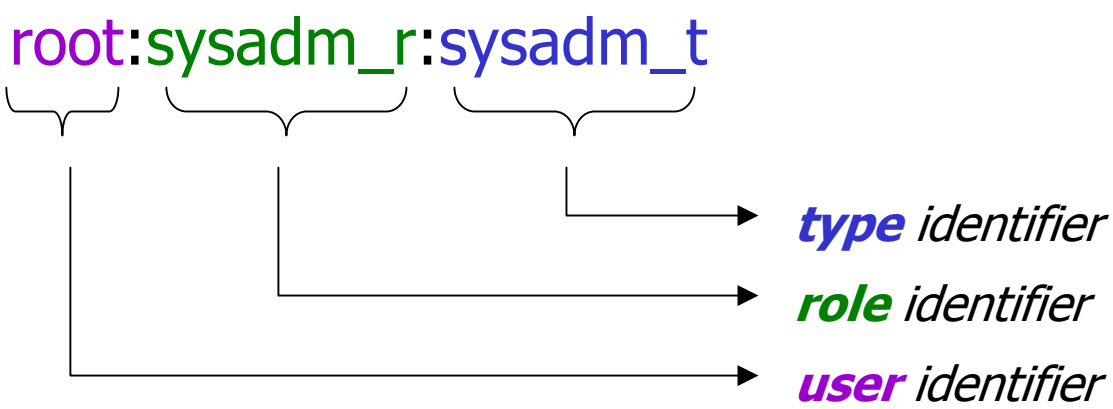
SE Linux Access Control



Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SELinux Access Control Attributes

- SE Linux assigns subject and objects a security context:



- Security context is only access control attribute in SE Linux

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Object Classes and Permissions

- SE Linux currently defines 29 kernel object classes

1) security	2) process	3) system	4) capability
5) filesystem	6) file	7) dir	8) fd
9) lnk_file	10) chr_file	11) blk_file	12) sock_file
13) fifo_file	14) socket	15) tcp_socket	16) udp_socket
17) msgq	18) sem	19) msg	20) shm
21) ipc	22) node	23) netif	24) netlink_socket
25) packet_socket	26) key_socket	27) rawip_socket	
28) unix_stream_socket		29) unix_dgram_socket	

- Each with their own fine-grained permissions
  - For example, file object class has 19 permissions:

ioctl	read	write
create	getattr	setattr
lock	relabelfrom	relabelto
append	unlink	link
rename	execute	swapon
quotaon	mounton	execute_no_trans
entrypoint		

Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Simple TE Allow Statement

allow user_t bin_t : file { read execute } ;

domain type — user_t

object type — bin_t

object class — file

permissions — { read execute }

<u>Translation</u>: A process with type user_t can read or execute files of type bin_t

- All access is denied unless explicitly allowed

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

passwd Program Example

allow passwd_t shadow_t : file
{ create ioctl read getattr lock write setattr append link unlink rename };

- Allows processes with passwd_t domain type read, write, and create access to files with shadow_t type
  - Purpose: passwd program runs with passwd_t type, allowing it to change shadow password file (/etc/shadow)

- Shadow password file attributes:

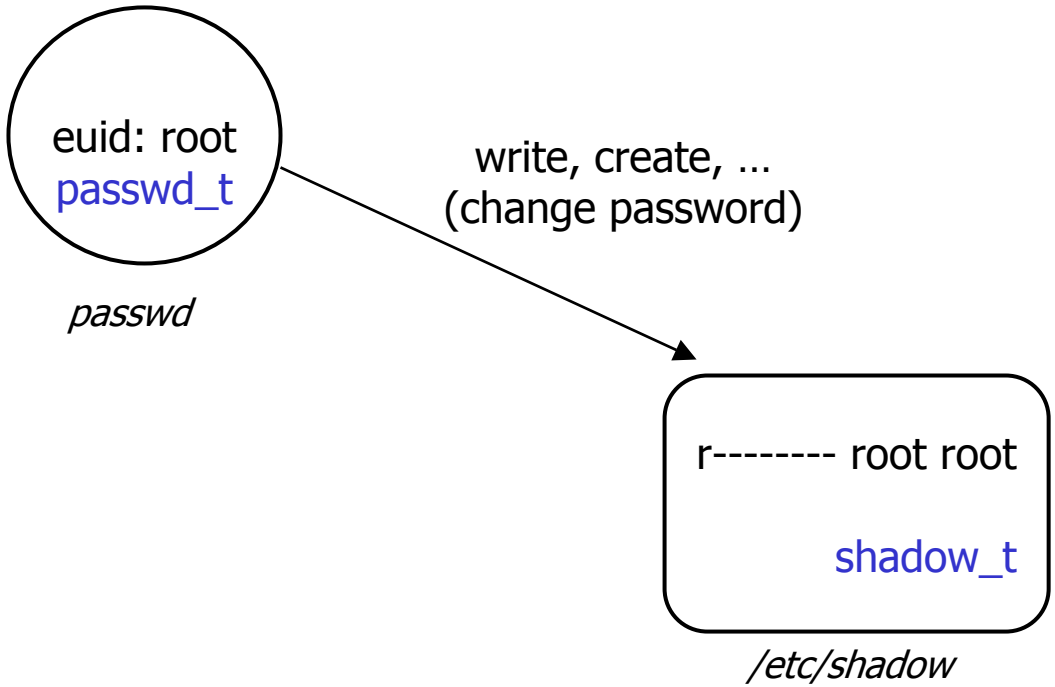
  -r-------- root root system_u:object_r:shadow_t /etc/shadow

  standard Linux — *only allows root to create new copies of file (via uid==0 privilege)*

  SE Linux — *only allows passwd_t domain (via above allow rule) to modify file*

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

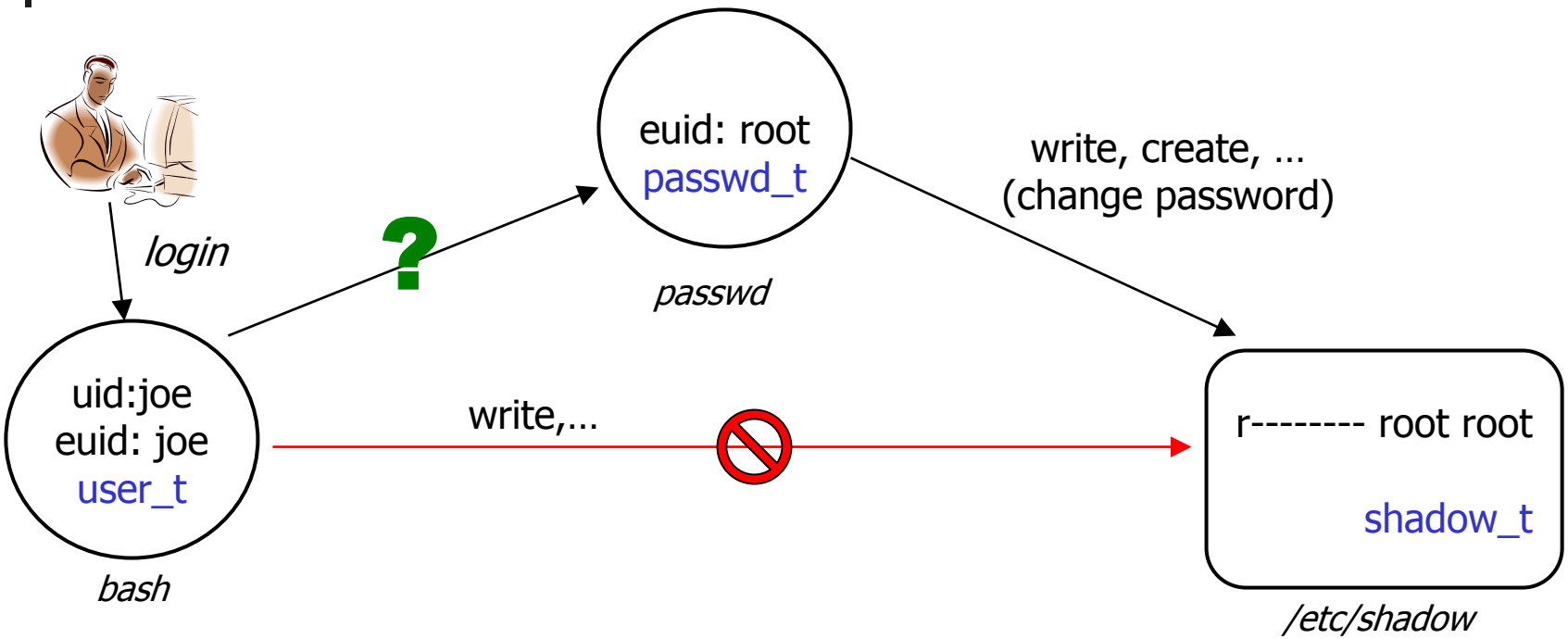
passwd Program Example




```
allow  passwd_t  shadow_t :  file
            {  read getattr write setattr append };
```


Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

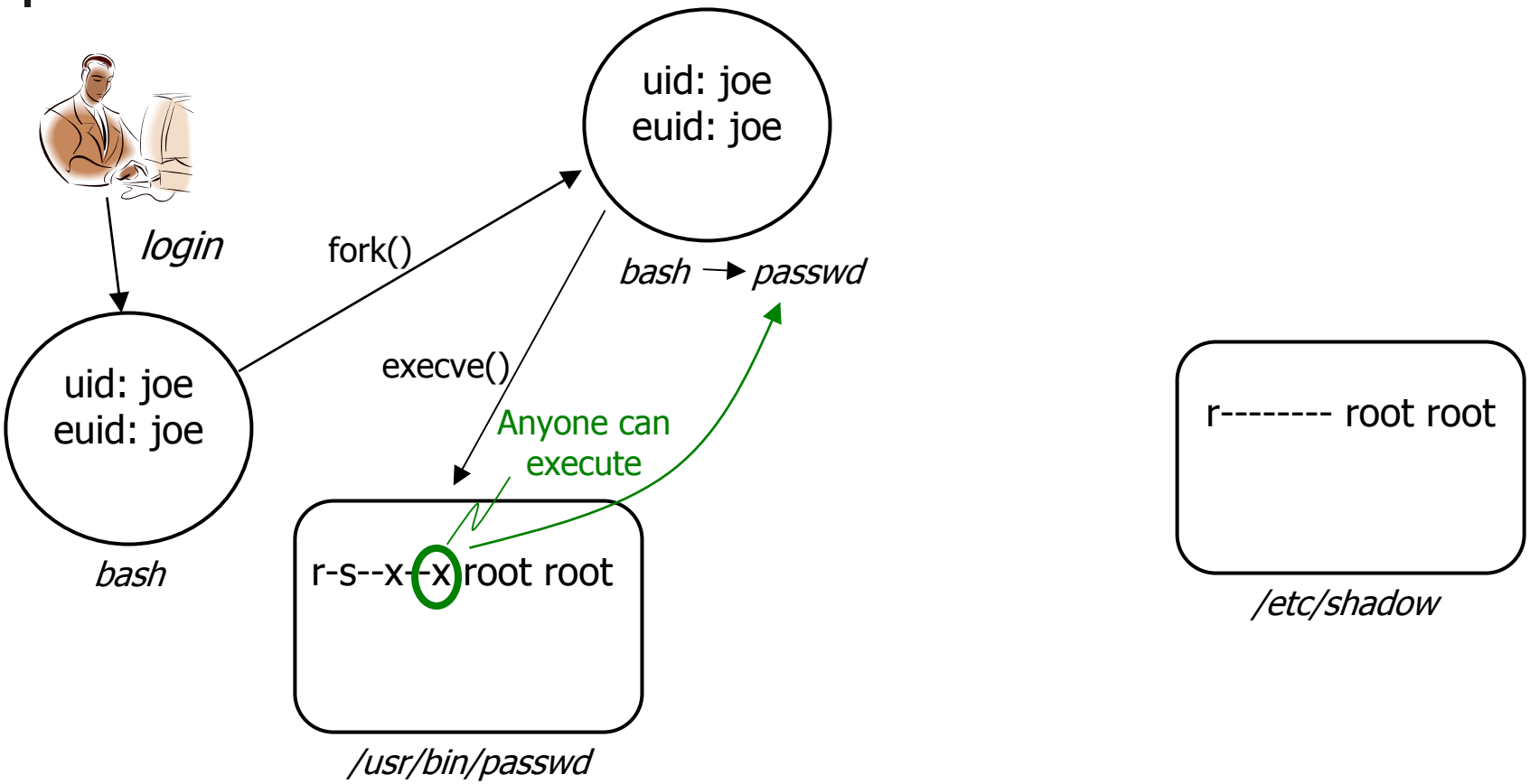
Problem of Domain Transitions




```
allow  passwd_t  shadow_t :  file
                 {  read getattr write setattr append };
```

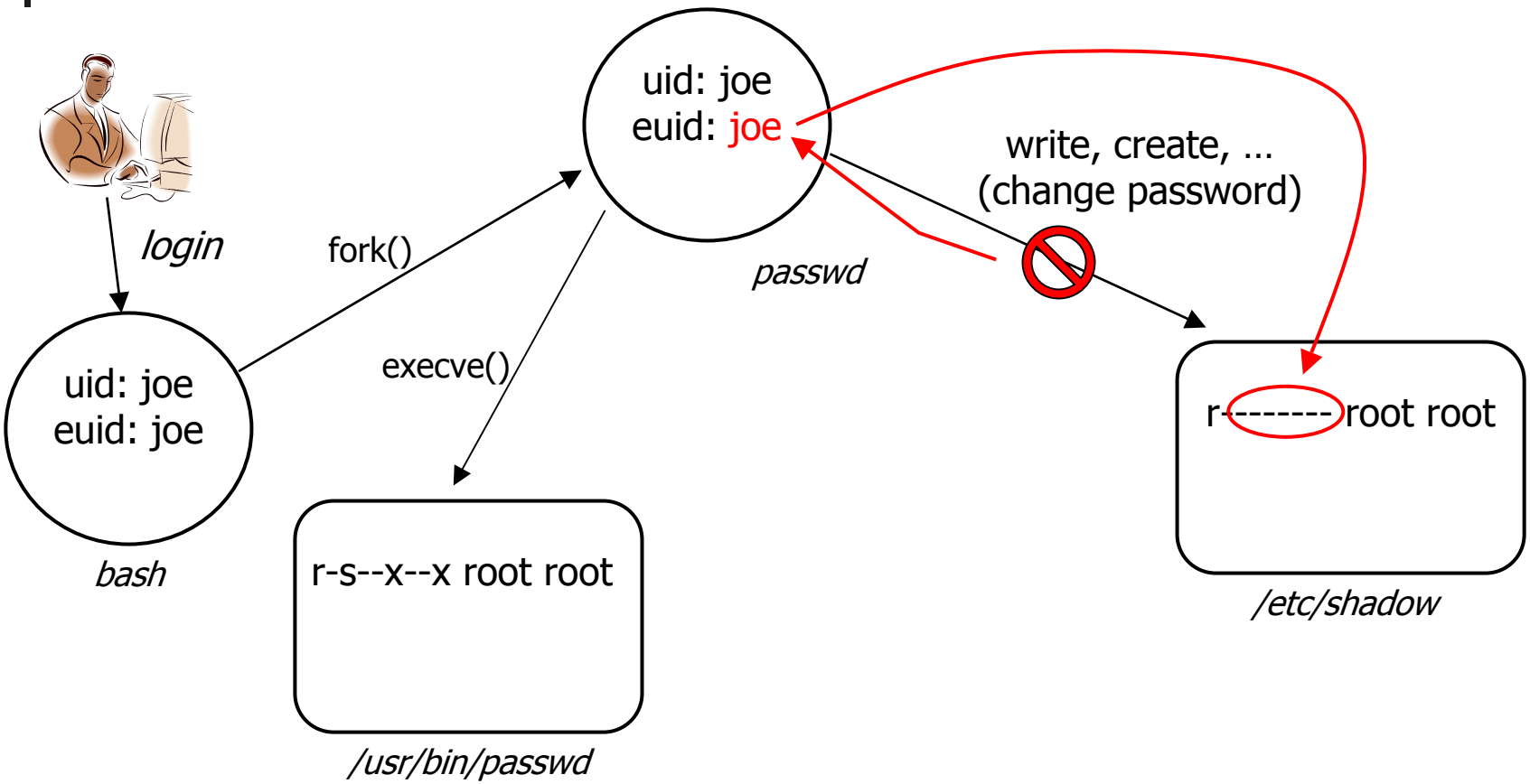

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Standard Linux passwd Security



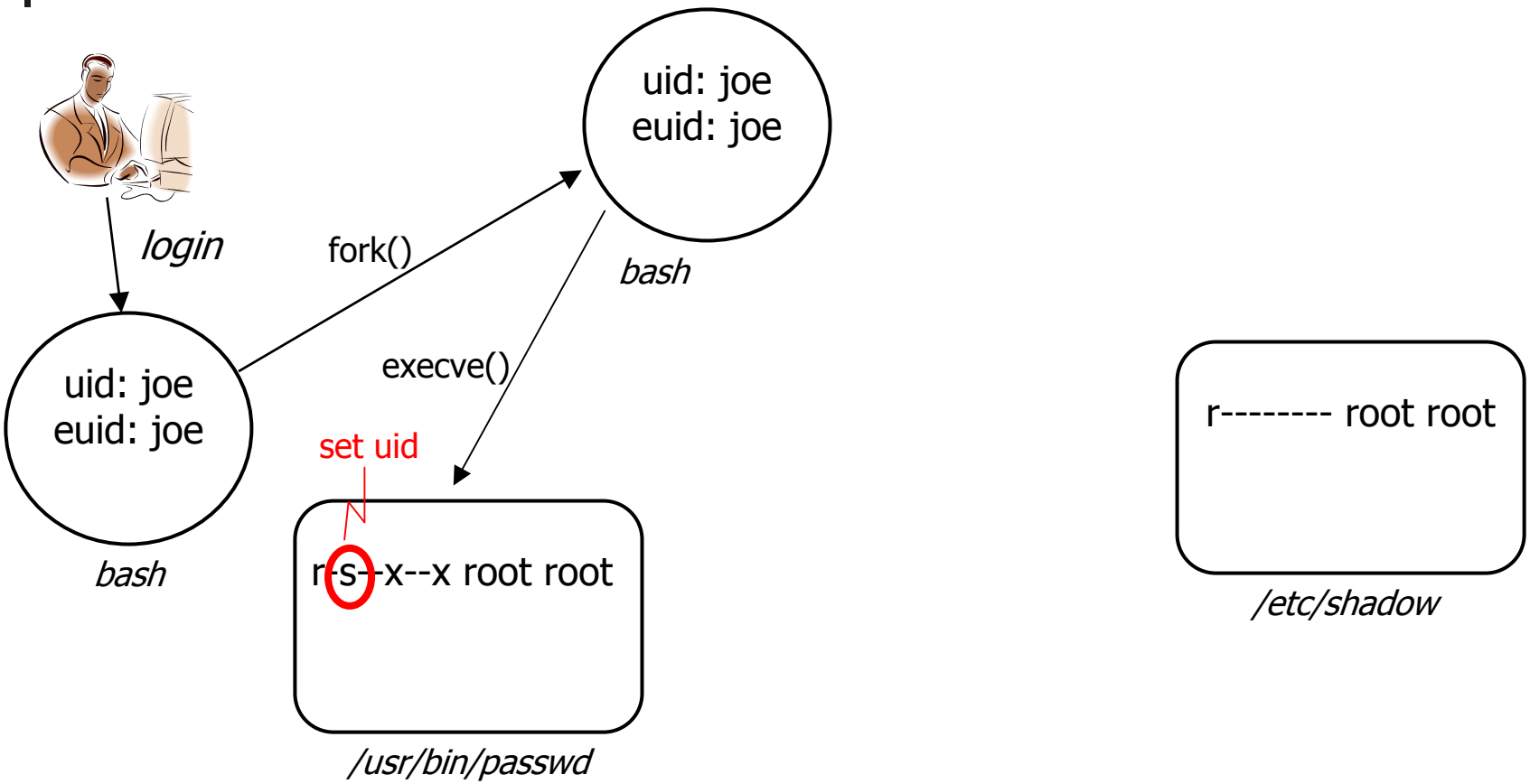
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Standard Linux passwd Security



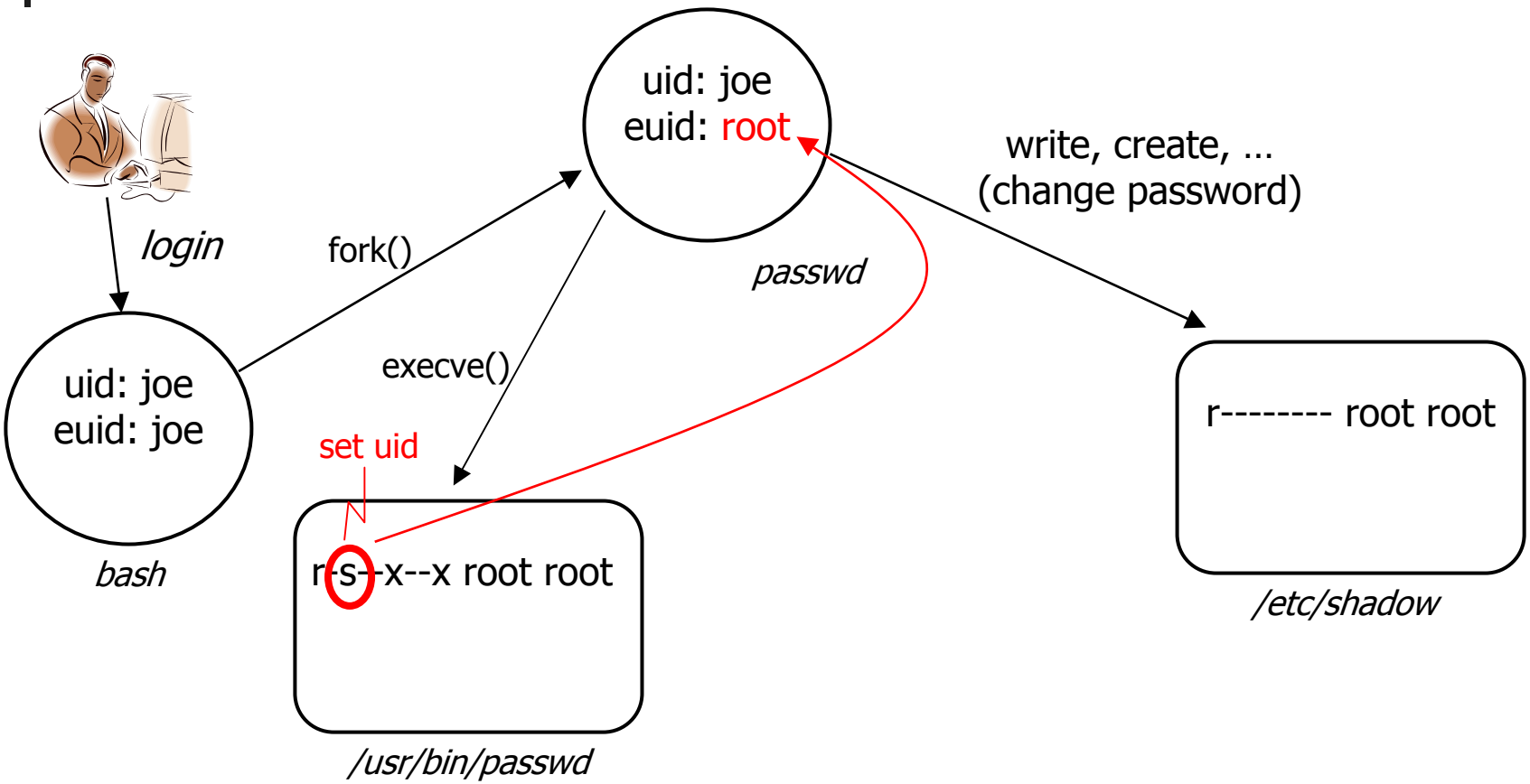
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Standard Linux passwd Security



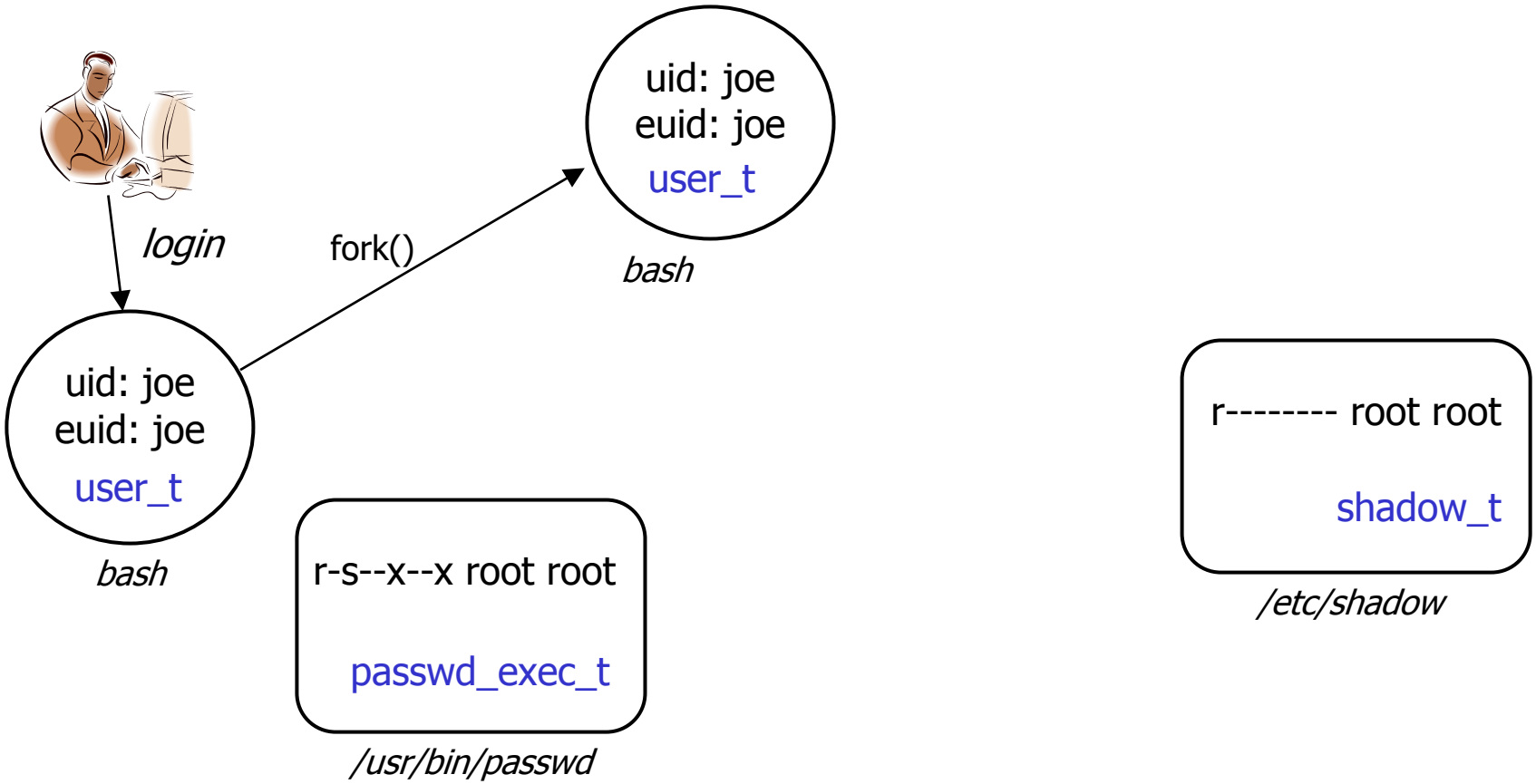
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Standard Linux passwd Security



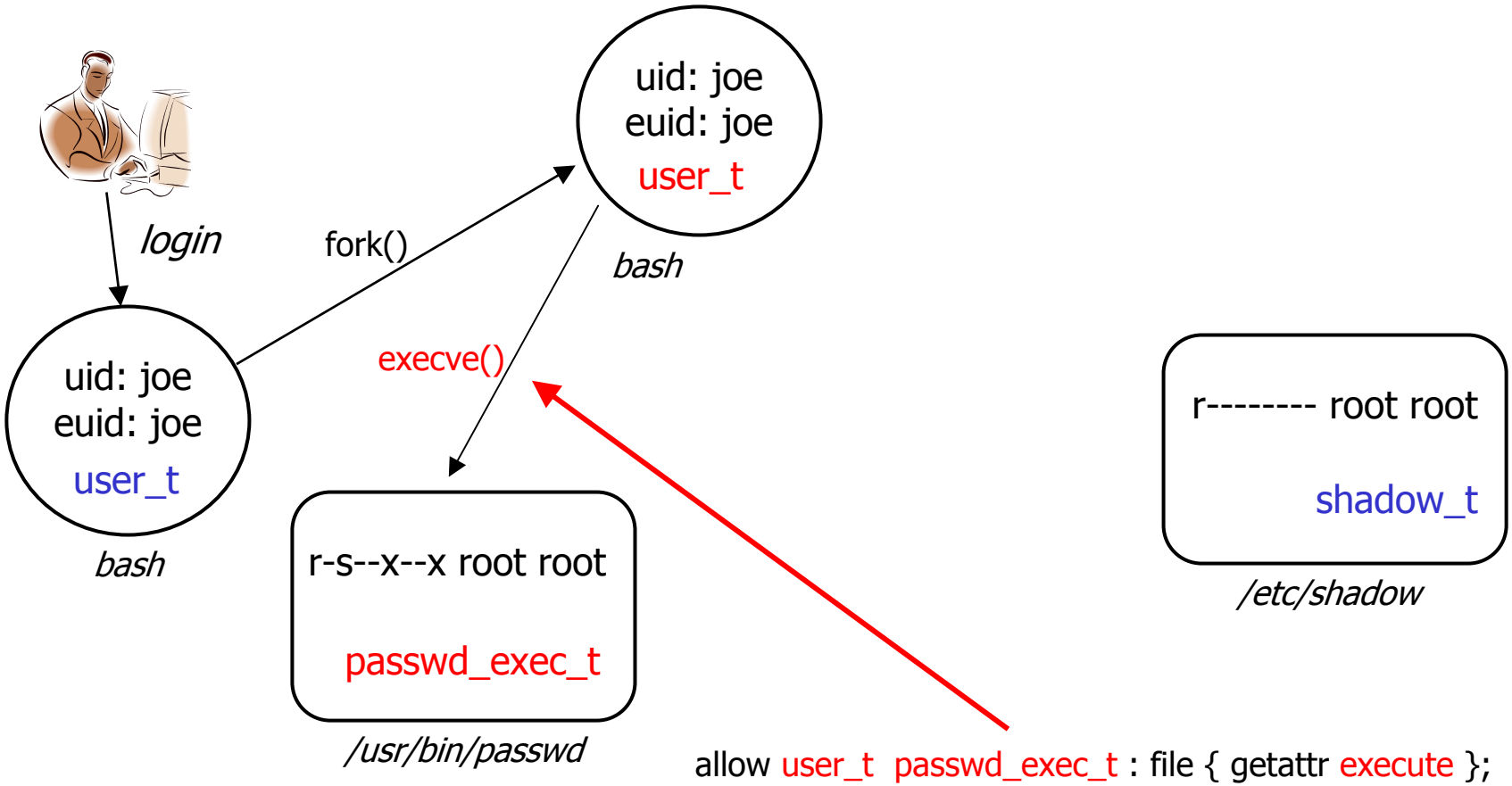
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SE Linux Domain Transitions



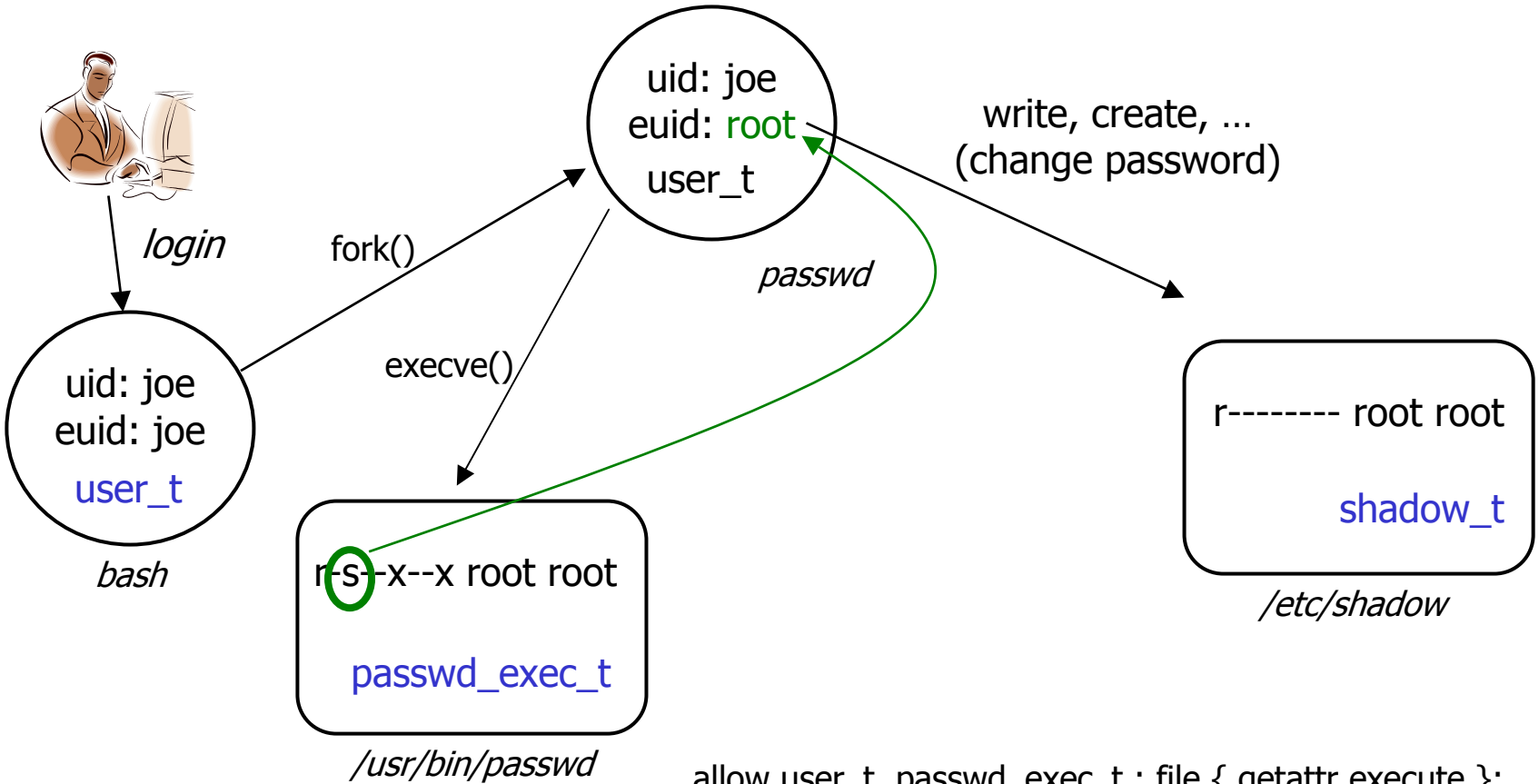
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SE Linux Domain Transitions



Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

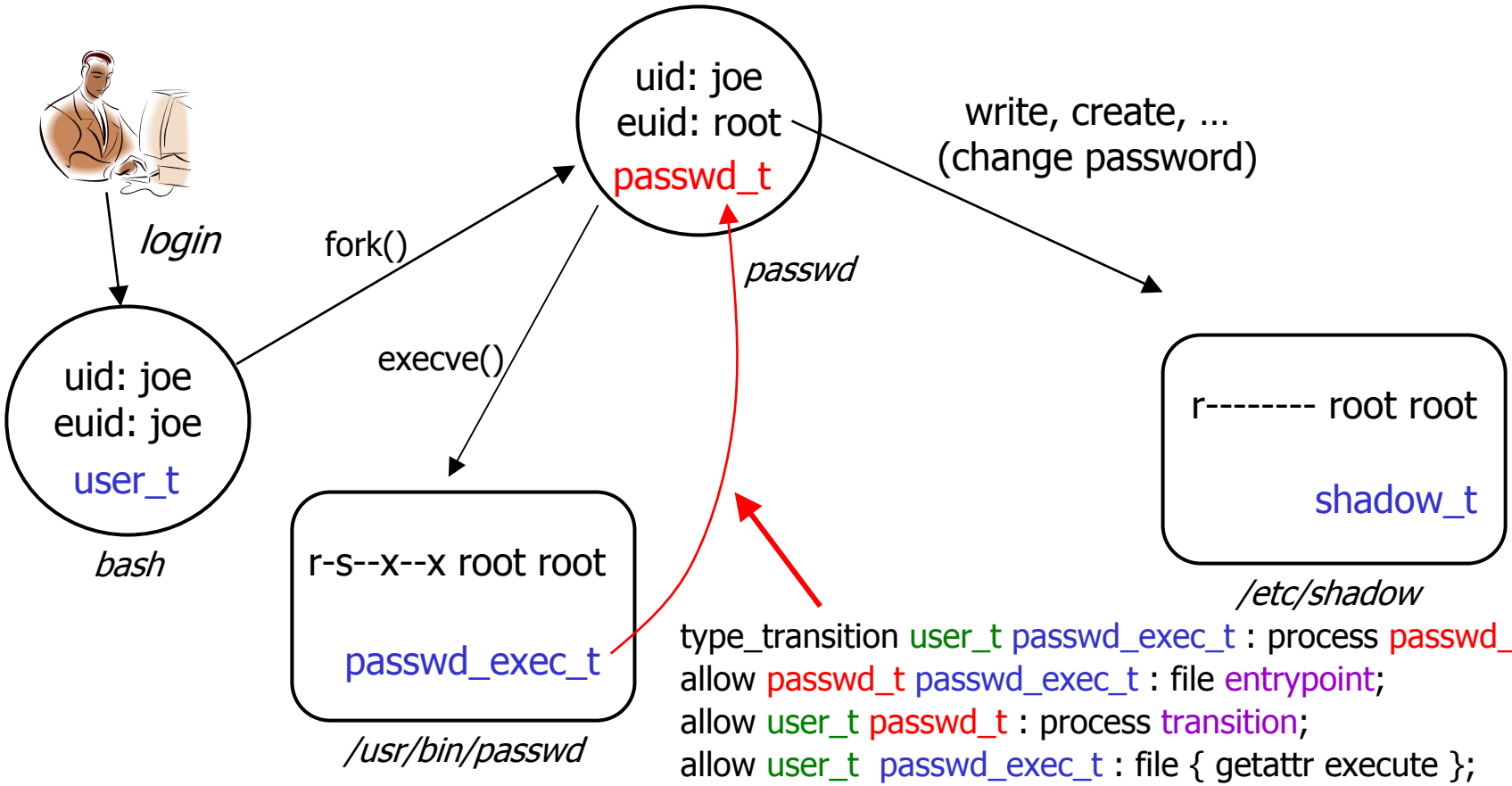
SE Linux Domain Transitions



allow user_t passwd_exec_t : file { getattr execute };

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SE Linux Domain Transitions



Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Why Type Enforcement

- Extremely configurable mandatory access control
  - flexible (not tied to a single security objective)
  - dynamic (loadable policy, future work)
  - possible to be pragmatic within a policy
    - even necessary in Linux due to legacy!
  - fine-grained access control
    - object classes and permissions, unlimited types and rules
- But…

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Challenges with SE Linux

- Policies can be complex
  - legacy issues with Linux/Unix
  - need for pragmatism
- Flexibility comes with a price!
  - tens object classes, over a hundred permissions
  - unlimited domain, types, object instances
  - tens of thousands of rules not uncommon
- Assurance of mechanism untested in traditional manner
  - open source model helps, modest assurance at best (B2/EAL5)
  - certainly no worse than Linux (or other mainstream OSs)
  - in fact much better with a good TE policy

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

TE Security Policy Analysis

Understanding a SE Linux TE Policy

Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

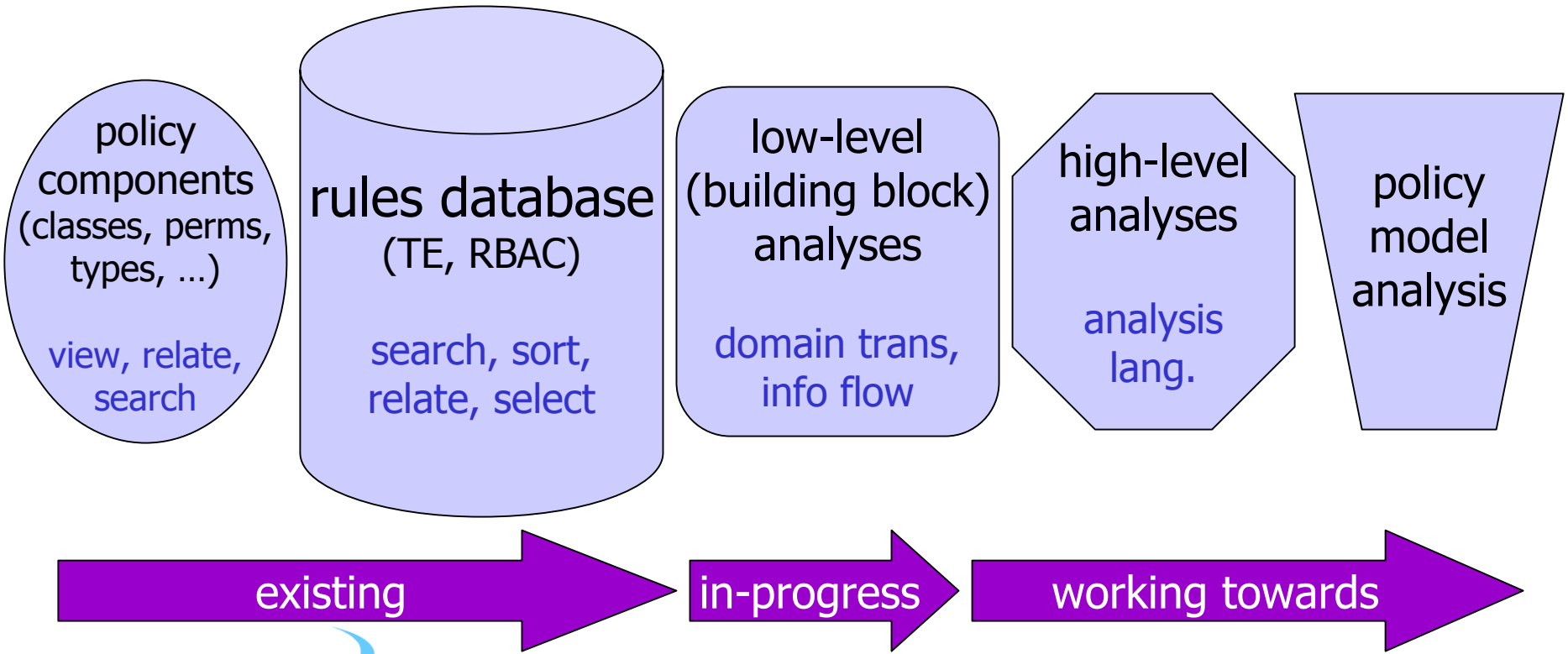
Why Analyze TE Policies?

- TE does not imply particular security properties
  - unlike most non-discretionary access control mechanisms
  - desired security properties must be demonstrated
- SE Linux is configurable and dynamic
  - given application policy must be certified
- SE Linux can have some scary policy complexity
  - large, complex OS (compared to security kernel)
  - current worse case default policy:
    - 1,000 types
    - 17,870 allow rules
    - 926 type_transition rules

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

APOL (Policy Analysis Tool)

- Evolutionary Development (driven by practice)
  - from basic components to abstract analyses



Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

APOL: Policy Analysis Tool

File Search View Advanced Help

Policy Components | Policy Rules | Analysis | policy.conf

Analysis Type

Forward Domain Transition
Reverse Domain Transition
Direct Information Flow
Transitive Information Flow (Exper)

Analysis Options

Required parameters

Starting type: sysadm_t

Select starting type using attrib: domain

Flow direction: In Out Either Both

Optional result filters

Filter results by object class:
blk_file
capability
chr_file
dir
fd
fifo_file
file

Select All | Clear All

Filter end types using regular expression: ^user

New | Update | Info

Analysis Results

Empty Tab | Results 1 | Results 2 | Results 3

Direct Information Flow Tree

sysadm_t
user_chkpwd_t
user_cron_spool_t
user_crond_t
user_crontab_t
user_devpts_t
user_home_dir_t
user_home_ssh_t
user_home_t
user_ssh_t
user_su_t
user_t
user_tmp_t
user_tmpfs_t
user_tty_device_t

Direct Information Flow Data

Information flows into sysadm_t - from user_chkpwd_t

Object classes for IN flows:
file
(5885) allow sysadm_t domain : { file lnk_file sock_file fifo_file } { read getattr lock ioctl };

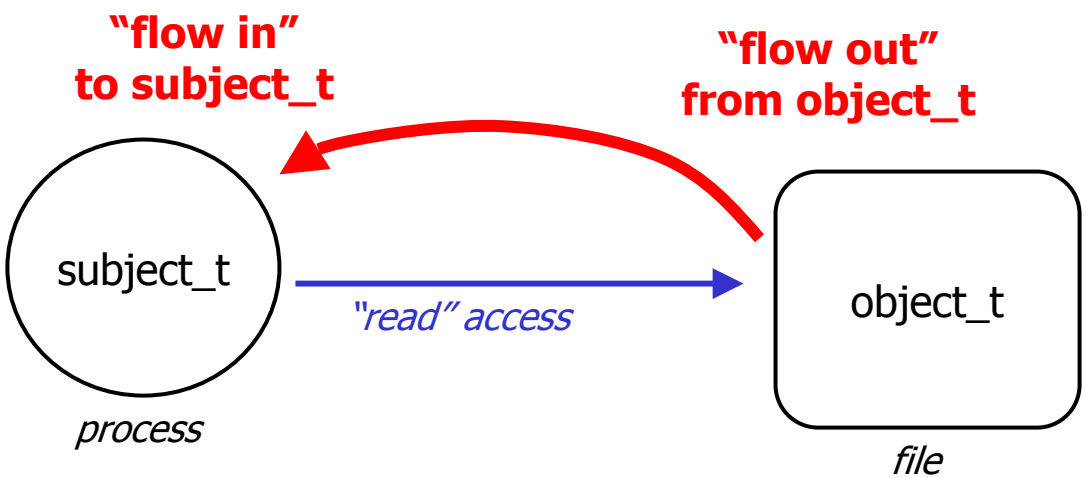
Close Tab

Open a new policy | Classes: 30 Perms: 122 Types: 272 Attribs: 48 TE rules: 4144 Roles: 4 RBAC rules: 7 Users: 5 | v.13 -- v.15

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Information Flow Analysis

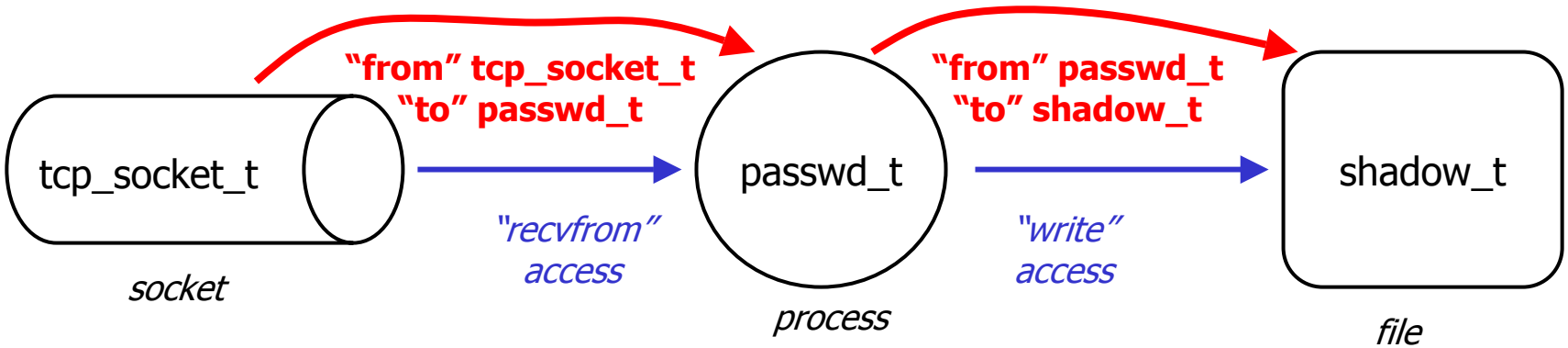
- low-level: direct analysis
  - map each permission to: read, write, both, none
  - from "starting type" determine: in, out, either, both
  - file class example:



Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Information Flow Analysis

- low-level: transitive analysis
  - determine multi-hop paths between two types
  - transitive closure and shortest path analyses
  - greatly complicated by classes and permissions
    - weighted perm maps ("overt" vs. "covert" flows)
    - filter by types ("trusted") and object class



Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Information Flow Analysis

- high-level: invariant language (planned)
  - "no flow", "only flow", etc.
- example: high-to-low "guard" application

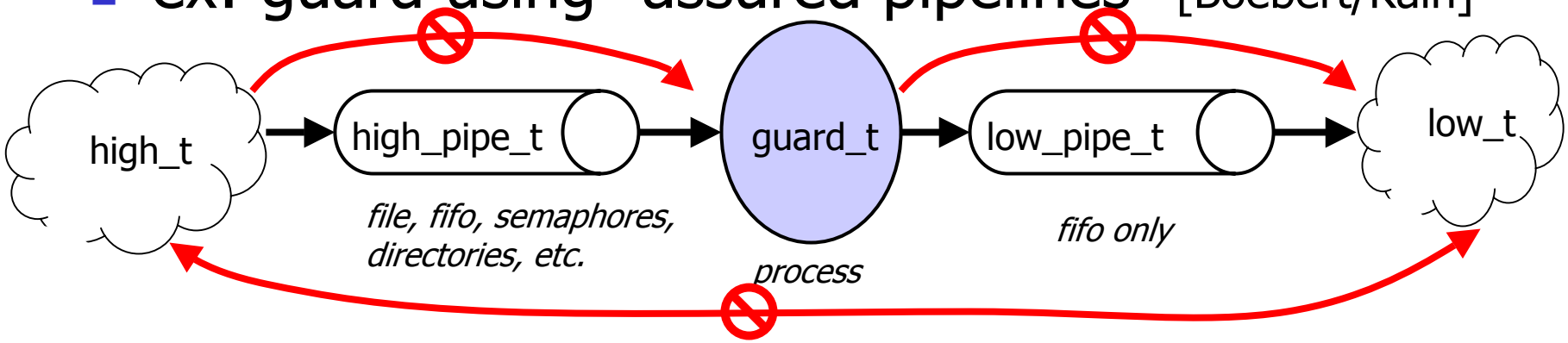
```
no_flow high_t low_t : * guard_t;
```

"from type" *"to type"* *"classes"* *"except these types"*

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Information Flow Analysis

- policy model validation: (goal)
- ex: guard using "assured pipelines" [Boebert/Kain]




```
no_flow high_t low_t : * guard_t;  //domain isolation
no_flow low_t high_t : * guard_t;  //same
no_flow high_t guard_t : * high_pipe_t;
no_flow guard_t low_t : fifo low_pipe_t;
no_flow high_pipe_t low_pipe_t : * guard_t;
...
```


Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

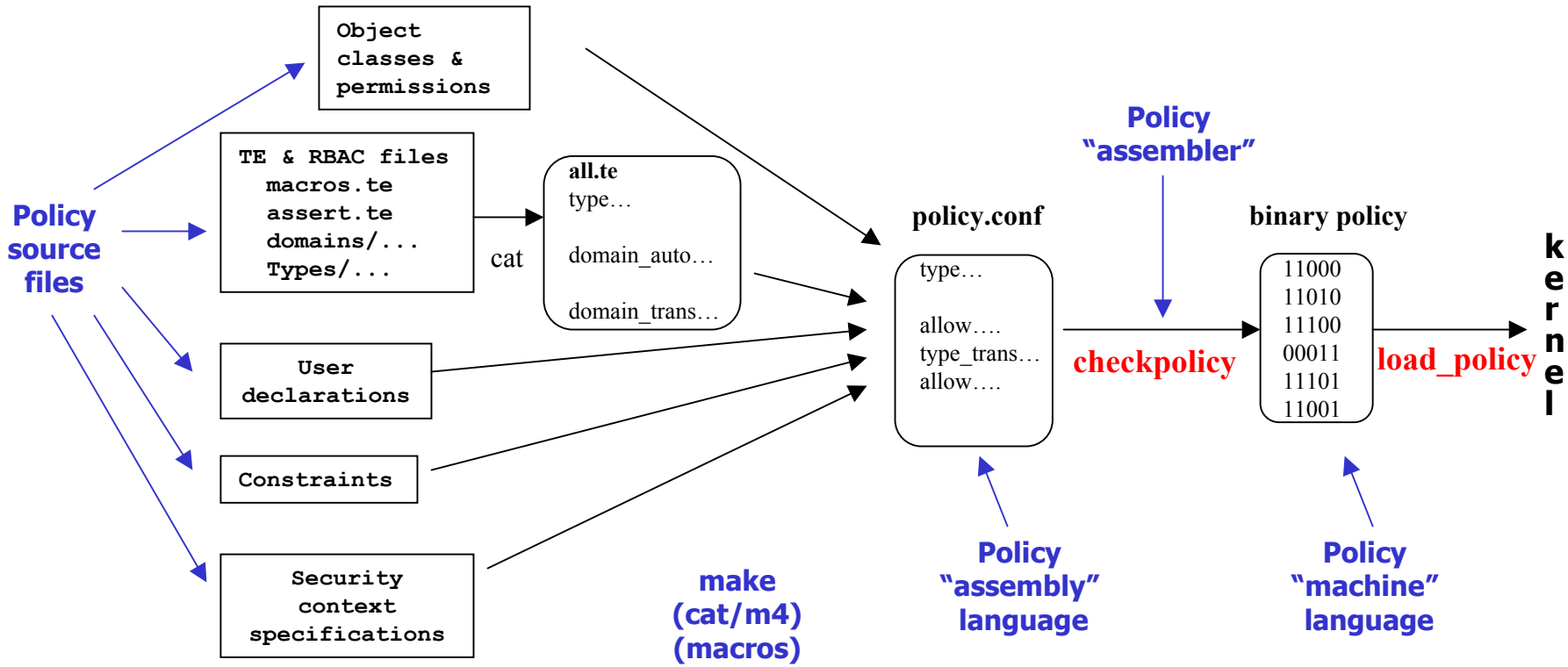
TE Security Policy Development and Management

Working Towards More Useable Type Enforcement Capabilities with SE Linux

Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SE Linux TE Policy Development



Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SE Linux TE Policy Development

- Development practice fairly primitive
  - text files, m4 macros, low-level language
  - some evolving abstractions (via macros)
- Debugging tools nearly nonexistent
  - sys logs, “checkpolicy” errors
  - setools analysis (apol) and editor (sepcut)
- Much room for work and research
  - abstract language
  - policy model analysis tied to development
  - much more…

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SELinux TE Policy Management

- Even more primitive
  - entire policy loadable
- On-going work to make policy dynamic
  - conditional policy language (nearly complete)
  - binary modules (under design)
    - significant challenges; significant benefits
    - requires load-time validation of security properties
  - relating object labeling with policy changes
- Future
  - fully dynamic mandatory security, with validation
  - distributed management

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

Backup information on Tresys setools

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SeTools Overview

- Apol: Policy analysis/debug tool
  - allows analysis of policy via policy.conf
  - all major policy components; search and find
  - emerging automated analyses
    - information flow and domain transition
- SePCuT: Policy customization, basic editing
- SeUser: Example user manager for SELinux
  - command line and GUI
- SeAudit: audit message analyzer (NEW)
- Misc. command line tools in progress

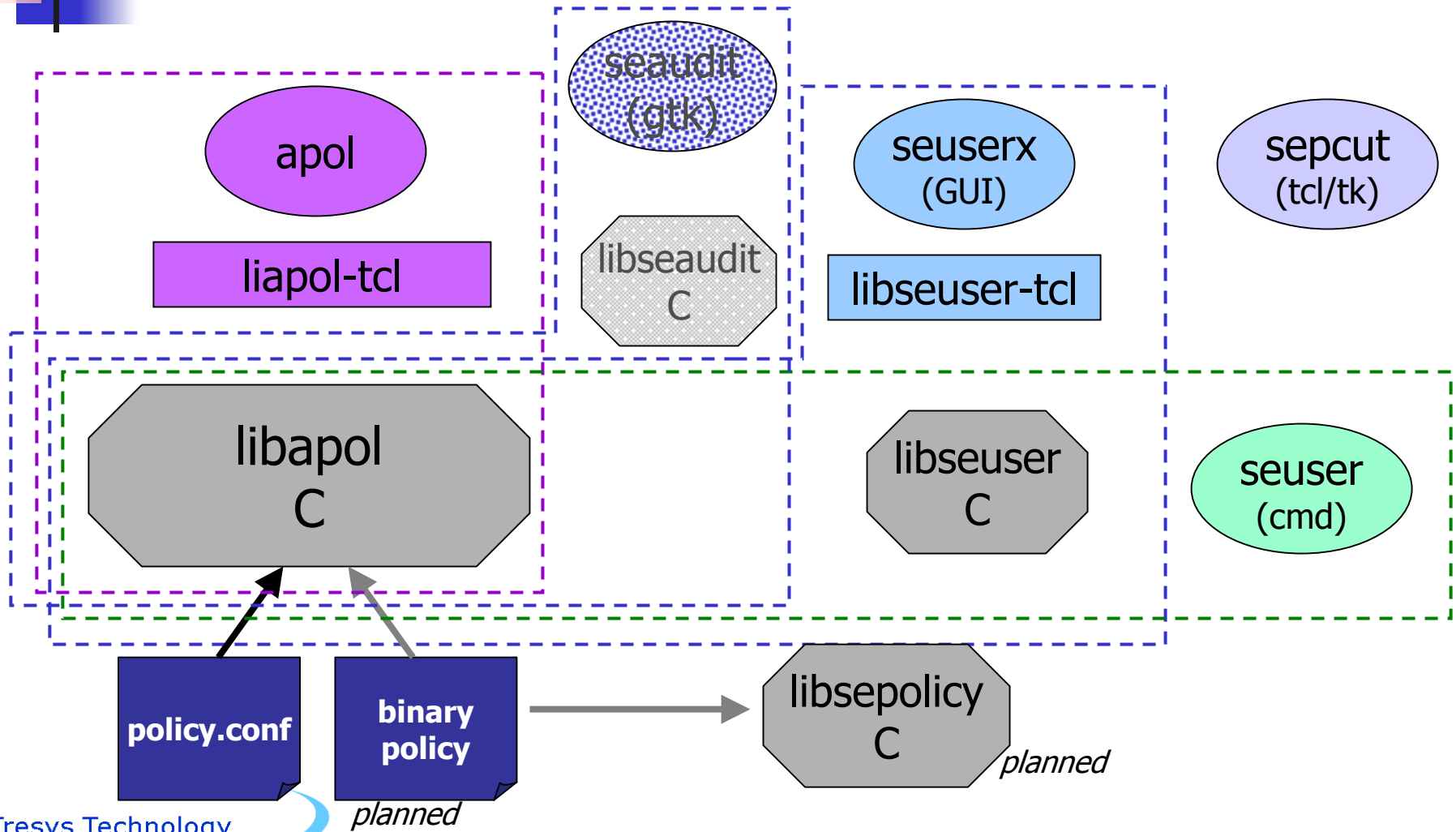
Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SeTools Background

- Motivation
  - TE policy analysis
    - security evaluation and certification
  - TE policy development
    - writing policies
    - debugging policies (related to analysis)
  - TE policy management
- Evolutionary development, GPL source code
  - www.tresys.com/selinux
  - also in NSA distribution (sourceforge.net/projects/selinux)

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SeTools Architecture Overview



Tresys Technology

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

SeTools Future Directions

- Binary policy library and tools
  - inevitable evolution of binary format
  - evolution of binary policy mgmt infrastructure
- Additional apol analyses
  - policy invariant analysis statements
  - more automation
- Integration of policy analysis and development

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)

QUESTIONS?

Frank Mayer
Tresys Technology
mayerf@tresys.com

www.tresys.com/selinux

Copyright © 2002, 2003 Tresys Technology, LLC (www.tresys.com/selinux, selinux@tresys.com)