Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Analyzing Integrity Protection in the SELinux Example Policy

Trent Jaeger et al. (presented by S. Kalkowski)

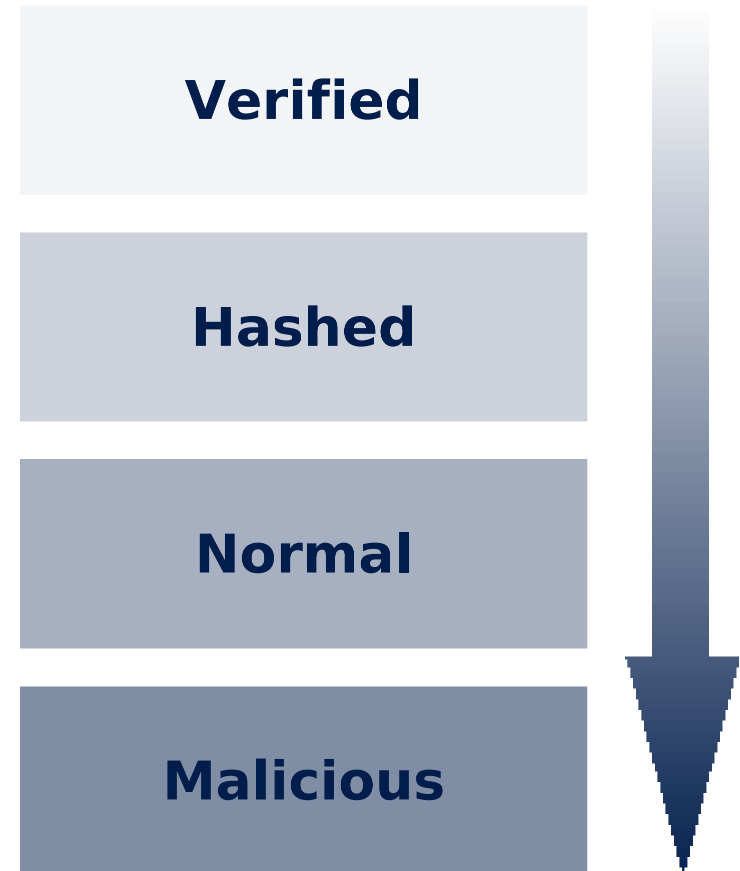
Dresden, 2008-01-23

Motivation

- SELinux example policy doesn't define protection goals, but demands of the different applications
- Define integrity protection goals for the overall TCB
- Handle conflicts with the example policy
- Provide a tool that eases up that work to system administrators and developers

Simple MAC :: Biba model

- Easy model
- Goal:
  - higher integrity layers have not to be polluted/dependent by/on lower ones
- Only two rules:
  - no write up
  - no read down
- Very strict and impractical



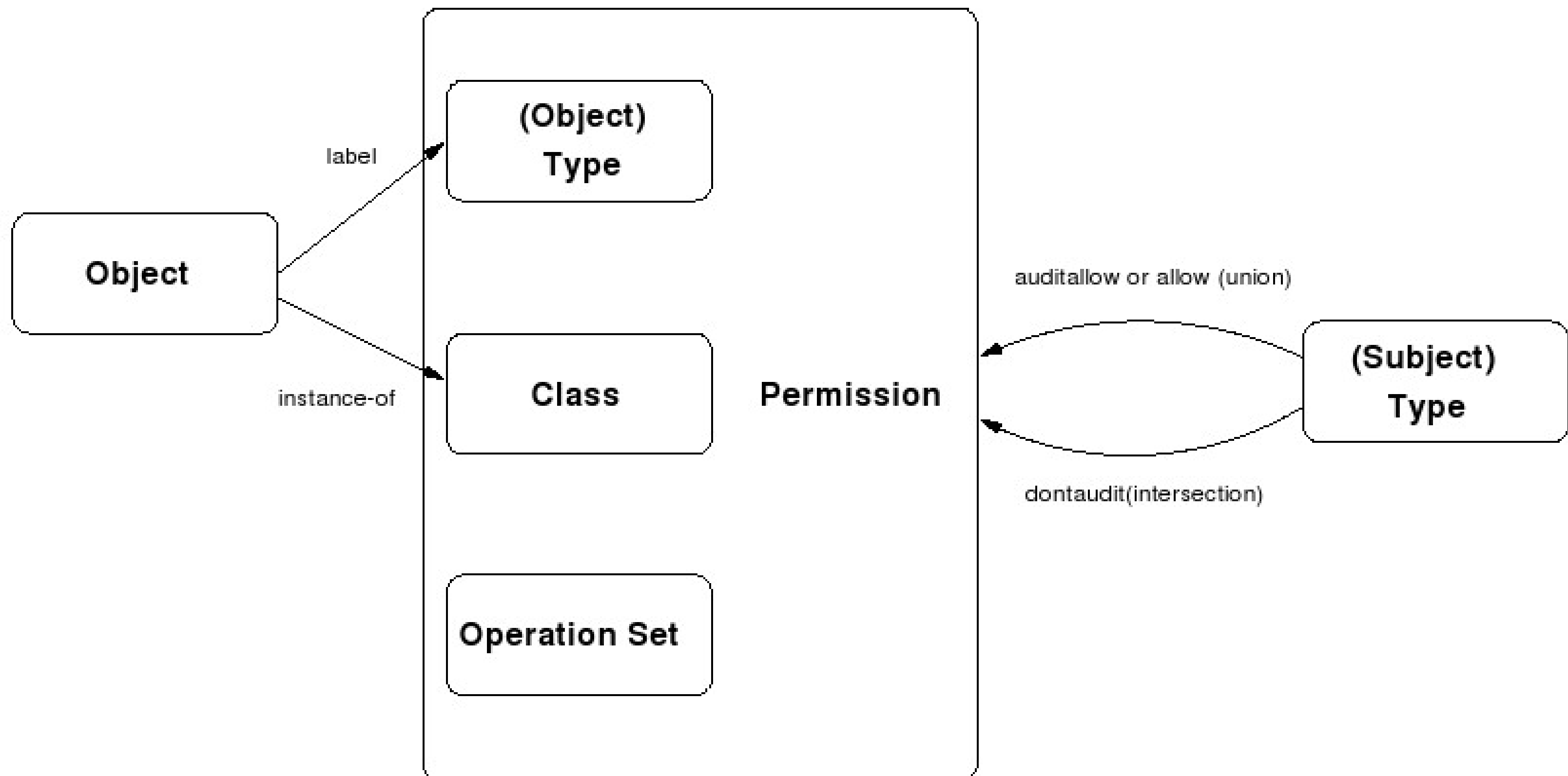
TECHNISCHE UNIVERSITÄT DRESDEN

More flexible Models

- Type Enforcement
- Role Based Access Control
- No strict hierarchy
- **Problem:**
  - No explicit security goals anymore
  - The more permissions statement the more tricky is a proof of certain security properties

SELinux Policy Model

- Extended **T**ype **E**nforcement model



TE + RBAC + IBAC

- Beside the type declarations and allow statements, there are:
- Attribute declarations
- Type transition statements
  - execution of certain programs
  - creation of files by certain processes
- Role to types transition statements
- Role transition statements when executing a new program (e.g.: login)
- User to roles mappings

➔ **Complex model!**

Example:: 1/3 of ssh.te

```
attribute ssh_server;
type ssh_exec_t;
corecmd_executable_file(ssh_exec_t)
type ssh_keygen_t;
type ssh_keygen_exec_t;
init_system_domain(ssh_keygen_t,ssh_keygen_exec_t)
role system_r types ssh_keygen_t;
type ssh_keysign_exec_t;
corecmd_executable_file(ssh_keysign_exec_t)
type sshd_exec_t;
corecmd_executable_file(sshd_exec_t)
type sshd_key_t;
files_type(sshd_key_t)
ifdef(`targeted_policy',`
unconfined_alias_domain(sshd_t)
init_system_domain(sshd_t,sshd_exec_t)
type sshd_var_run_t;
files_type(sshd_var_run_t)
ifdef(`enable_mcs',`
init_ranged_system_domain(sshd_t,sshd_exec_t,s0 - mcs_systemhigh)
')
',`
type ssh_agent_exec_t;
files_type(ssh_agent_exec_t)
ssh_server_template(sshd)
ssh_server_template(sshd_extern)
init_daemon_domain(sshd_t,sshd_exec_t)
ifdef(`enable_mcs',`
init_ranged_daemon_domain(sshd_t,sshd_exec_t,s0 - mcs_systemhigh)
')
#      ')
...
```

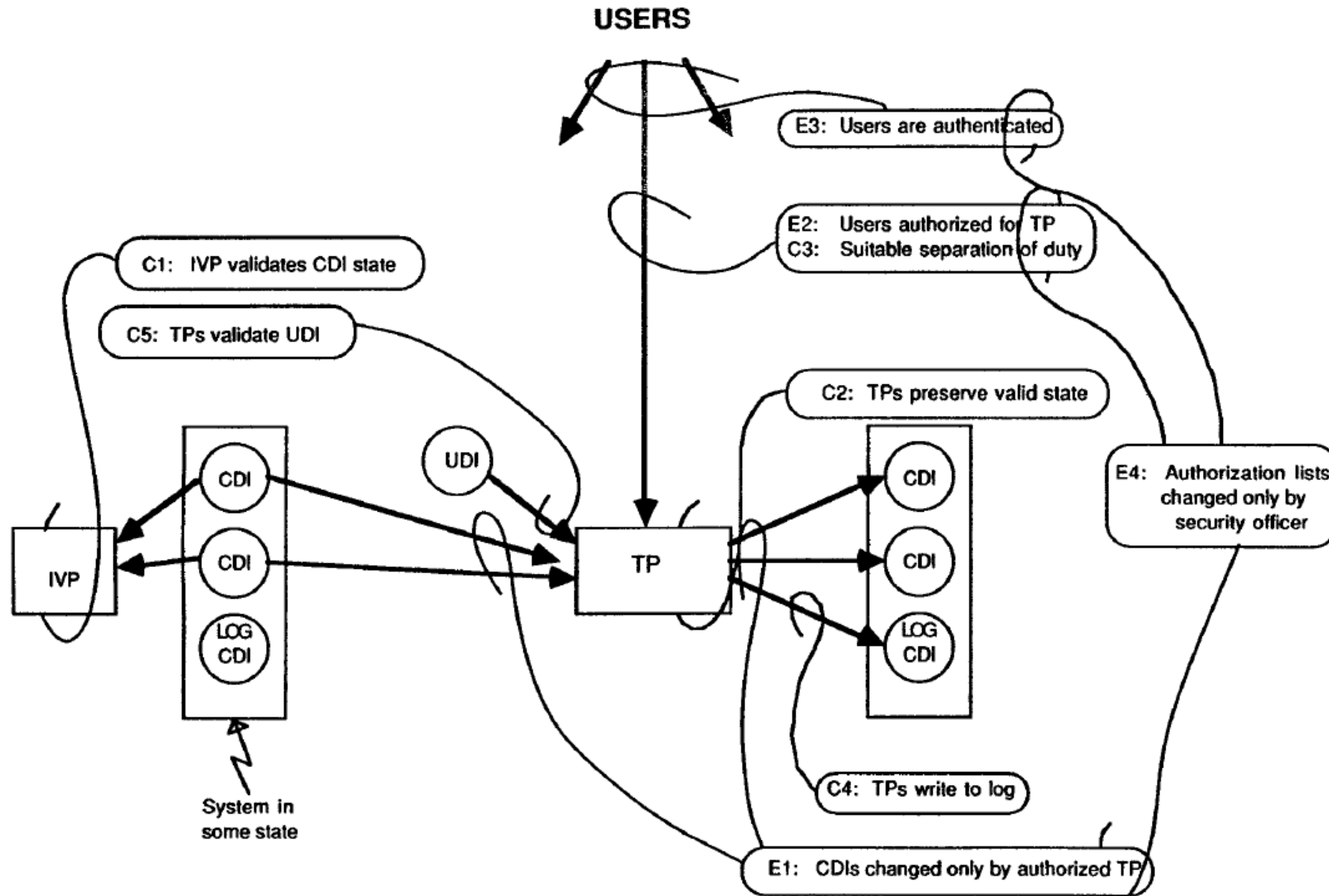
Problem Statement

- SELinux policy to complex to simply verify site goals, e.g.: of an organization, user, application developer
- Once some protection goals are verified, we cannot change the policy without verify it again
- How to ensure integrity of the common TCB

Goal: Integrity Protection

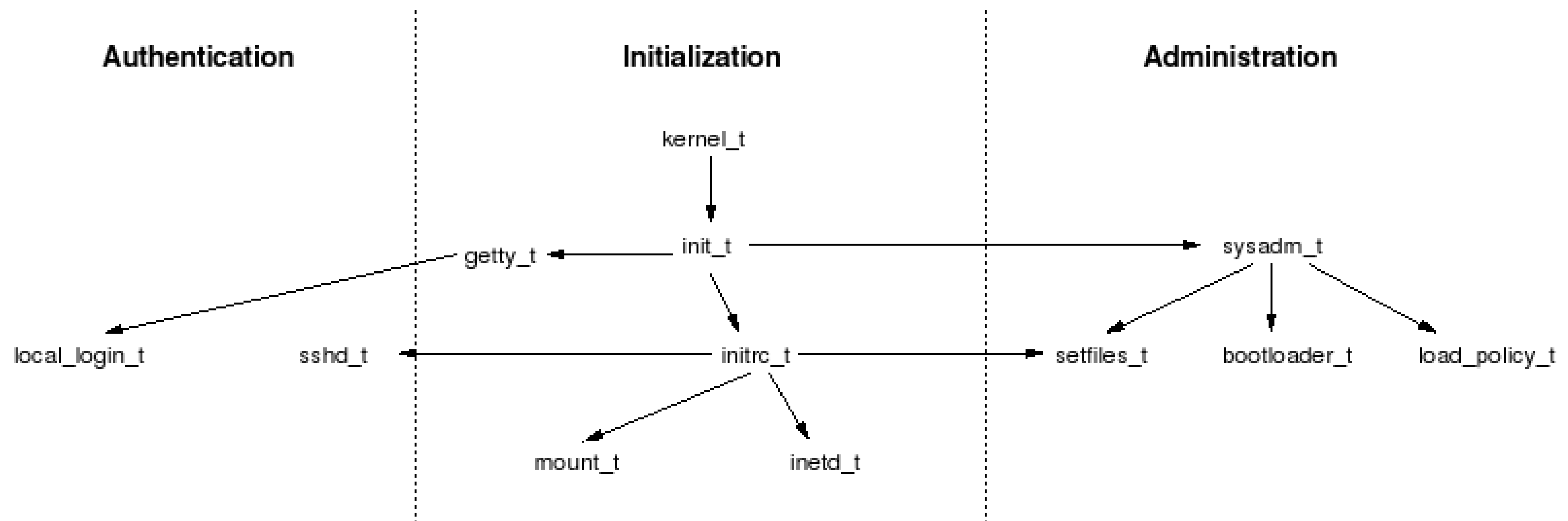
- Identify the TCB common to all components and protect its integrity
- Use Clark-Wilson Model:
  - Defines mainly two levels of integrity UDI and CDI
  - Defines how UDIs can be transformed to CDIs (by TPs) and how their integrity has to be verified (by IVPs)
  - Defines policy enforcement and certification rules
  - Introduced concept of **separation of duty** and **well-formed transaction**

Clark-Wilson model



TCB Integrity Protection

- Take initial TCB subject set from the SELinux policy (high integrity)
- Analyze read permissions to objects
- Find all non-TCB subjects (low integrity) that write to these object -> **integrity violation**



Identify Integrity Conflicts

- Using Gokyo – policy analysis tool:
  - Identify access control spaces of subjects
    - Set of all possible permission assignments
    - Set of prohibitions
    - Set of unknown rights
    - Overlapping regions
  - Define handling routines for whole subspaces
    - Granting or denying subspace
    - Manually changing the policy

Classify conflicts in policy

- Does the TCB subject only reads or also writes to the concerning object type?
  - read only data might be sanitized
- How many conflicts are caused by a non-TCB subject?
  - > 1 possibly a TCB candidate
- Can we exclude the conflicting subject ?
  - automatically detected by attributes
  - e.g.: httpd_*
  - also include related object types

Classify conflicts in policy

- Classifications for integrity conflicts

Class	*Description*
TCB or Candidate	Trusted subject types
Exclude Type	Type can be excluded from secure system with this TCB
Sanitize	A sanitized read may be used to protect TCB
Denial	Denial of conflicting rights can be used to protect TCB
Modify Policy	Policy must be edited to protect TCB

Practical Result

Trusted Type	*Conflict Type*	*Object Type & Op*	*Class*	*Resolution*
dpkg_t	tmpreaper_t	tmp_dpkg_t:file rw	exclude	exclude
initrc_t	many	file_type:blk/chr/file r	sanitize	sanitize
initrc_t	useradd_t	etc_t:file r	trust	trust
initrc_t	hwclock_t	clock_device_t:chr/blk rw	trust	trust
initrc_t	gpm_t	psaux_t:chr rw	exclude	exclude
initrc_t	sound_t, xdm_t	sound_device_t:chr rw	trust	exclude
initrc_t	httpd_admin_xserver_t	framebuf_device_t:chr rw	deny	exclude
initrc_t	many	initrc_t:fifo rw	deny	sanitize
kernel_t	slapd_t, squid_t, +	*:*_socket r	sanitize	sanitize
kernel_t	dhcpc_t	resolv_conf_t:file r	trust	exclude
kernel_t	dhcpd_t	var_run_[illegible]	trust	exclude
kernel_t	quota_t	file_t:file r	trust	trust
local_login_t	man[illegible]	proc_t:file r	sanitize	sanitize
local_login_t	insmod_t	[illegible]	[illegible]	[illegible]
local_login_t	logrotate_t	local_login_t:process r	trust	trust
mount_t	automount_t	autofs_t:dir rw	exclude	trust
mount_t	bootloader_t, fsadm_t	fixed_disk_device_t:* rw	trust	trust
sysadm_t	user_t	misc_device_t:* rw	deny	exclude obj
sysadm_t	many	sysadm_devpts_t/ptyfile:* rw	deny	change
sysadm_t	sysadm_*_t	sysadm_home_t:* rw	deny	change/sanitize one file
sysadm_t	sysadm_*_t	sysadm_tmp_t:file rw	exclude	change
sysadm_t	sysadm_irc_t	sysadm_irc_t:file rw	exclude	change/sanitize
sysadm_t	sysadm_xserver_t	sysadm_xserver_t:shm rw	exclude	exclude
sysadm_t	sysadm_xauth_t	sysadm_home_xauth_t:file rw	exclude	exclude
sysadm_t	admin	kernel_t:system avc_toggle rw	trust	trust
sshd_t	many	sshd_devpts_t/userpty:* rw	deny	change

46 %

“wrong classifications”

Discussion

- In general, are such techniques (for **automatically** resolving policy conflicts) needed or can we solve the problem of policy complexity (e.g. : by divide & conquer) ?
- What totally different approaches for handling policy complexity can we imagine -> e.g.: (transparent) user-interactive decisions
- Is policy complexity an artificial problem, as we have to implement least privilege beforehand