SE Linux Core Features

Russell Coker <russell@coker.com.au>
Internet and Security Consulting

I prefer an interactive format for my presentations, please ask questions at any time.

What is SE Linux?

- A system for Mandatory Access Control (MAC) based on the Linux Security Modules (LSM) framework
- Uses features of role-based and domain-type access control
- Tracks user identity through all operations
- Removes the power of UID 0, I have run several machines on the Internet with root as the guest account

LSM - Linux Security Modules

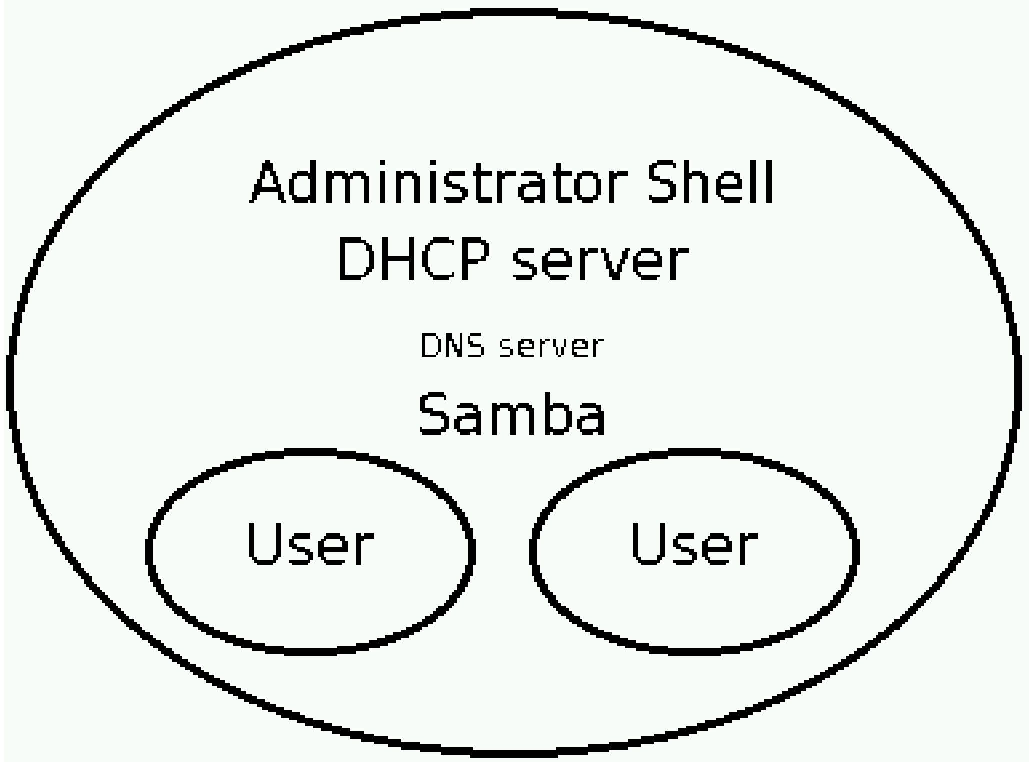
- Framework for security enhancements to the Linux kernel
- Devised at the request of Linus so that one kernel source tree can have multiple security modules available (Linus did not want to choose between the available modules)
- Restrictive controls only
- SE Linux is based on LSM and is included in the standard 2.6.x kernel
- Other LSM security modules have been written, but SE Linux is the only usable one (and the only serious one included in the standard kernel)

What is wrong with Unix security?

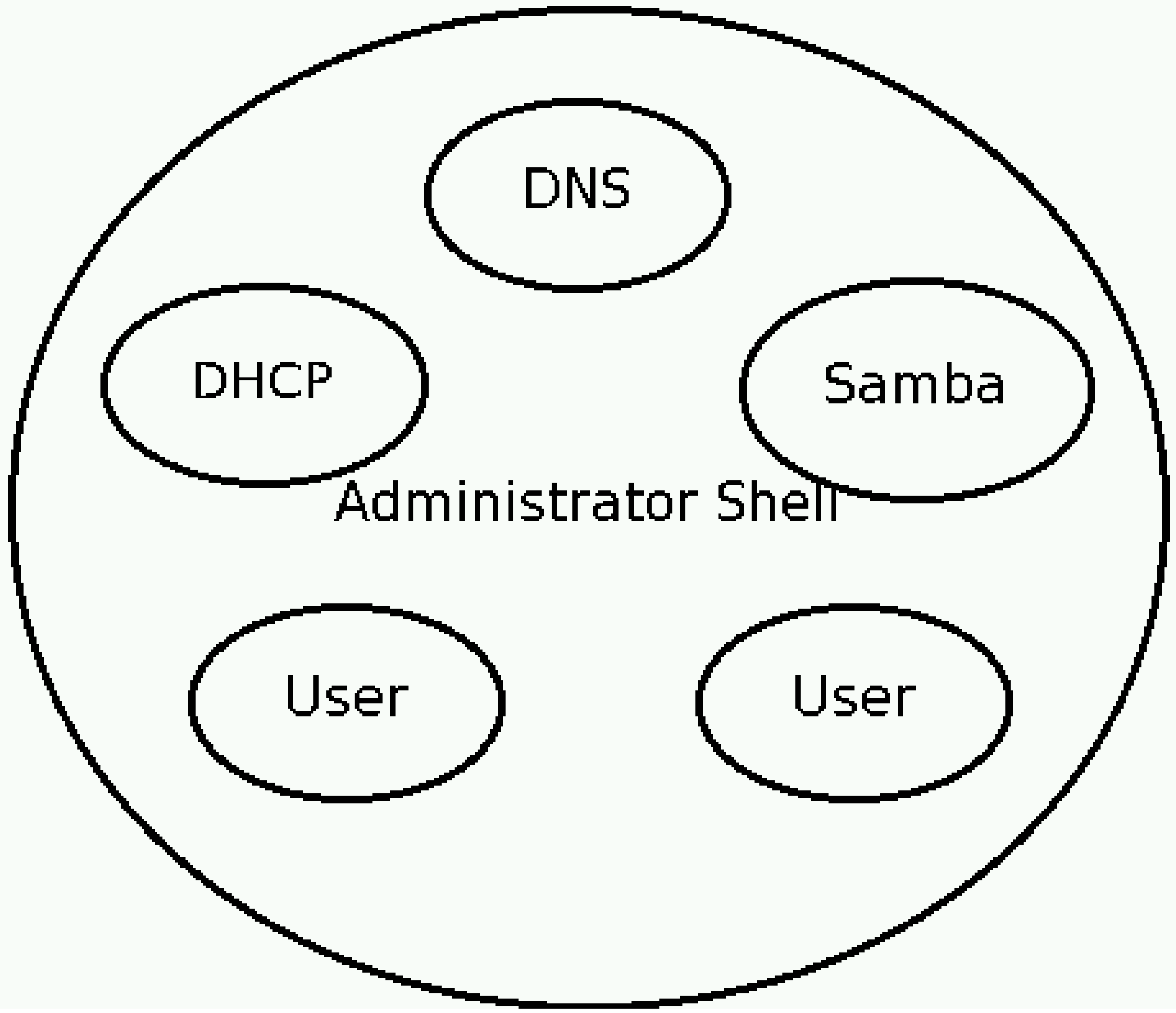
- Programs have full control over the access given to files they create (DAC)
- Therefore no protection against malicious software, "social engineering", and bugs in privileged software which may result in the software granting inappropriate access to files (EG creating a mode 777 file in /tmp)
- Too coarse grained - root vs non-root gives boolean security model for many cases
- Security model does not allow tracking of identity across change of UID

How SE Linux separates programs

Without SE Linux all server programs run with ultimate access to the system. Only the Administrator needs ultimate access, but all servers get it.



With SE Linux each server program gets limited access to the system. Servers are isolated from each other and prevented from harming the rest of the system.
Only the Administrator has ultimate access.



Domain Type Access Control 1/2

- Every process has a domain, every object (file, directory, socket, etc) has a type. The domains are a sub-set of the types (IE types that can apply to processes).
- A domain is the type for a process, SE Linux does not strongly distinguish between domains and types.
- The domain of a process will be used as a target context for operations such as sending signals
- The domain of a process may be changed at exec time either automatically through policy or through code in login type programs
- Different domains have different access rights, no domain is required to have a superset of the access of other domains

Domain Type Access Control 2/2

- The user can re-authenticate at any time to change domains with newrole and some other similar programs, or the domain can be changed automatically by file execution according to policy
- Each object that a process may act on has a type
- Policy rules determine what access every domain has to each type
- The number of domains can be varied according to needs, having a single domain would give the same result as a non-SE system. The more domains the more detailed the control you have over security and the more work to set it up.
- The new Targeted policy for Fedora gives less control and therefore less configuration work

Security Contexts

- Security context has the form Identity:Role:Type:LowLevel-HighLevel
- The levels are of the form Classification:CategoriesLow
- Classification and Categories are only used by MLS/MCS (explained in the next talk), if not using MLS/MCS then it's only Identity:Role:Type
- For disk objects the role is always "object_r"
- The same form of security context is used for processes and objects that they may access

Displaying Security Contexts

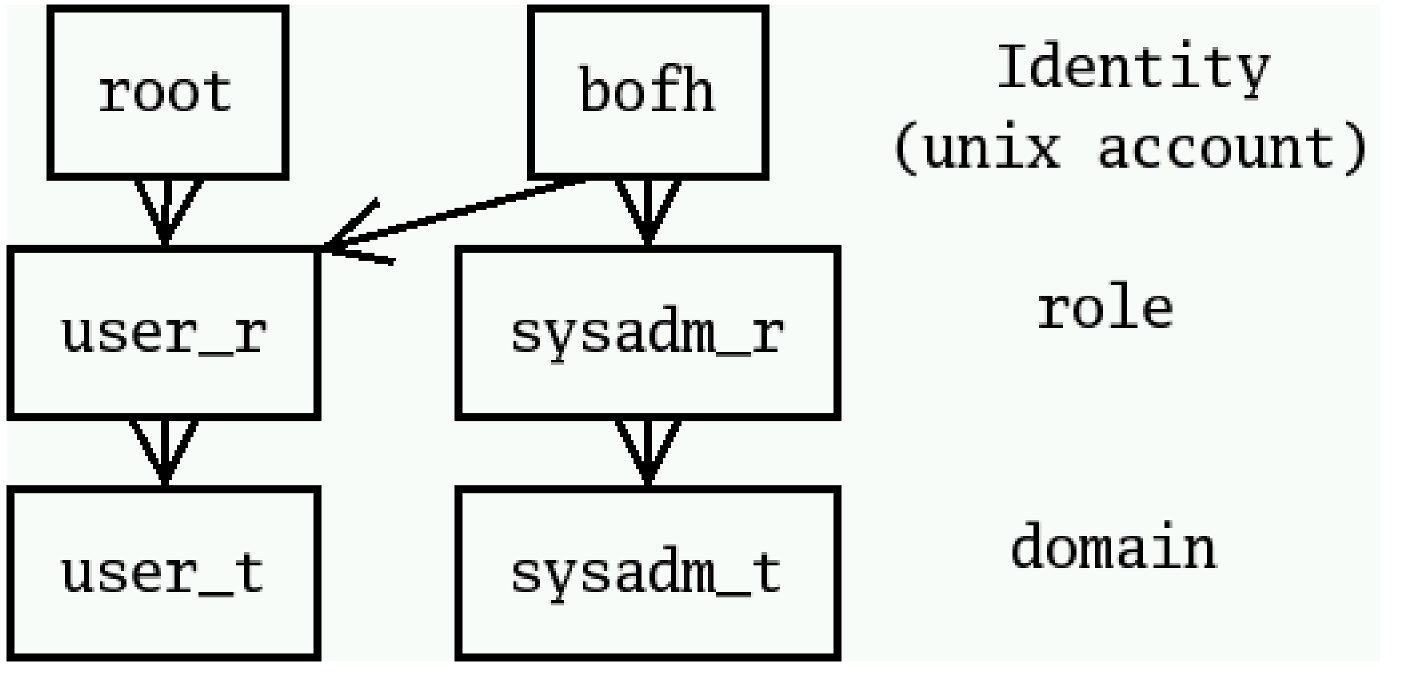
- $ ls -Z
  - -rw-r--r-- rjc rjc user_u:object_r:user_home_t play.mgp
- $ id -Z
  - user_u:system_r:unconfined_t
- $ id
  - uid=500(rjc) gid=500(rjc) groups=500(rjc) context=user_u:system_r:unconfined_t
- $ ps -Z
  - LABEL PID TTY TIME CMD
  - user_u:system_r:unconfined_t 2099 pts/0 00:00:00 bash
  - user_u:system_r:unconfined_t 2117 pts/0 00:00:00 vim
  - user_u:system_r:unconfined_t 2140 pts/0 00:00:00 ps
- $ cat /proc/self/attr/current
  - user_u:system_r:unconfined_t:s0
- $ cat /proc/1721/attr/current
  - system_u:system_r:crond_t:s0-s0:c0.c255

Role Based Access Control

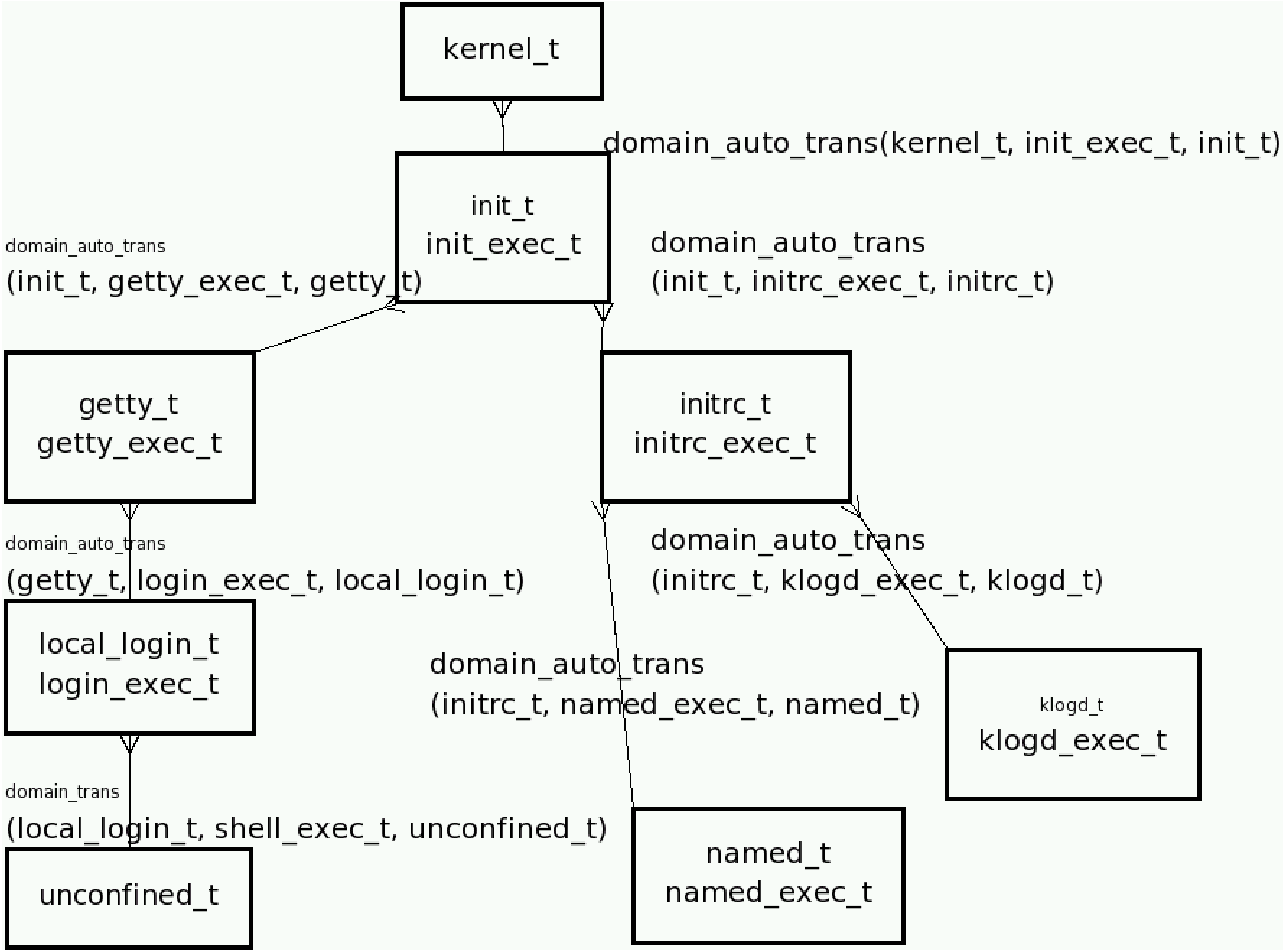
- Each role has a list of domains that may exist in it
- At login time the security context is changed (identity, role, and domain), also the newrole program may be used to change roles (comparable to an su operation)
- A role doesn’t often change, unlike the domain which may change often automatically without the user noticing
- The role determines which domains are permitted
- Roles may also be changed through role_transition rules in some situations

Identities

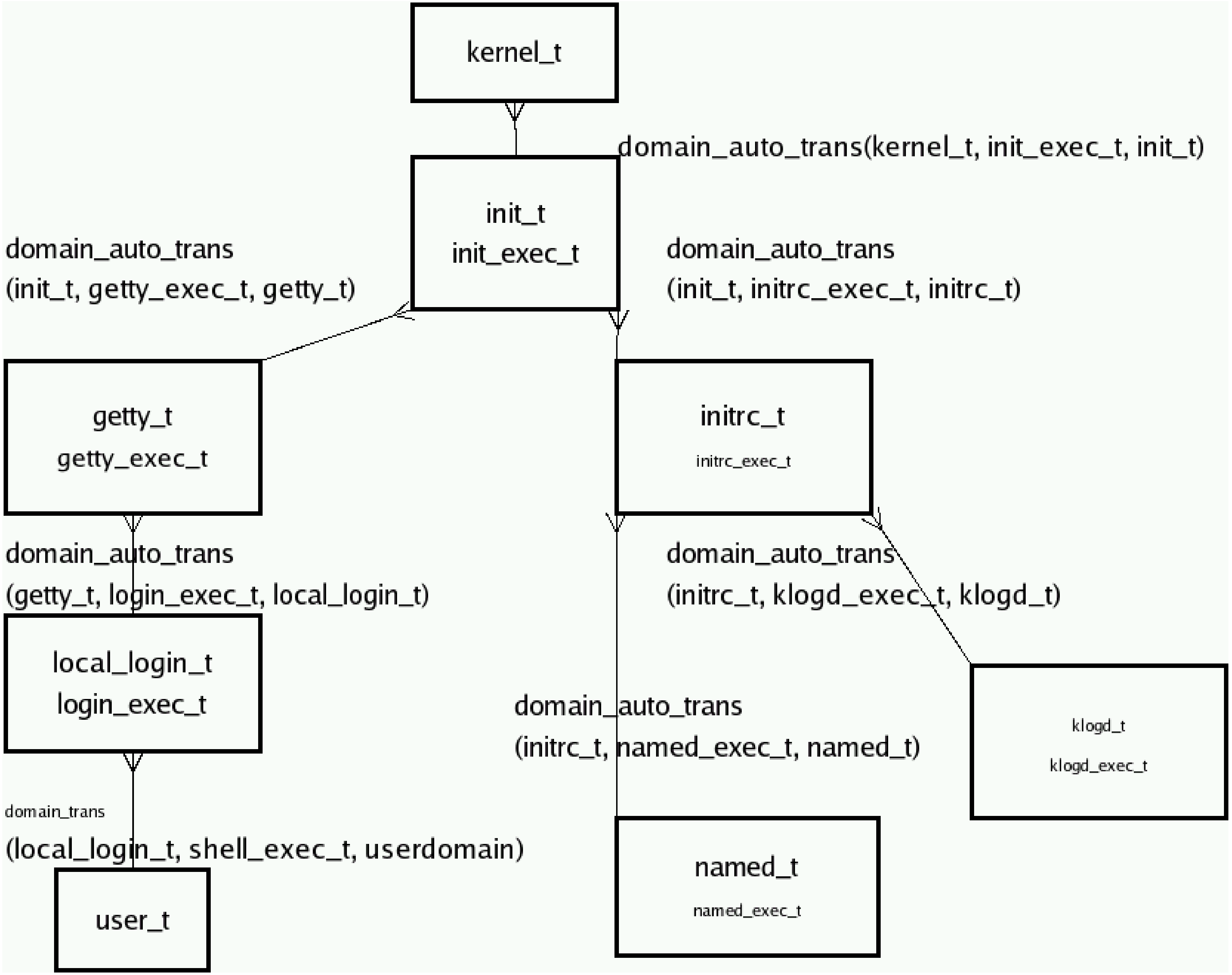
- The Identity is usually the Unix account name and is compiled into the policy database
- Identity controls the available roles which controls the available domains - but this level of control is not used in the Targeted policy



Transitions in the Targeted Policy



Transitions in the Strict Policy



Policy

- Written at a high level with M4 macros
- Compiled into a binary form that is understood by the kernel
- May have multiple binary modules of policy that are linked as part of the load process
- Loaded by /sbin/init at the start of the boot process before any other programs are executed
- A modified policy can be loaded at any time by the administrator

SE Linux Aware Applications

- An application can request a specific context for a file or directory that it creates. The policy determines which contexts the application may request.

- An application can specify the context for a program that it executes. The policy determines which contexts the application may request and which types of file may be executed to enter the domains in question. All login programs and crond use this interface.

Kernel interfaces 1/3

- selinuxfs file system (almost always mounted on /selinux) used for loading policy, enabling/disabling SE Linux, and querying the kernel policy database

- Display whether SE Linux is in enforcing mode:
  - $ /usr/sbin/getenforce
    Enforcing
  - $ cat /selinux/enforce
    1

- /proc/PID/attr/* files are used for discovering the current and previous contexts of a process and for a process to request that a non-default context be used for a program it executes or a file it creates

Kernel interfaces 2/3

- Querying the context of a file system object requires reading the security.selinux XATTR. The XATTR interface is supported by devpts and all other pseudo file systems.
- $ ls -Z foo
  - -rw-rw-r-- rjc   rjc   user_u:object_r:tmp_t   foo
- $ getfattr -n security.selinux foo
  - # file: foo
  - security.selinux="user_u:object_r:tmp_t:s0\000"

- Setting the context of a file system object involves setting the value of the security.selinux XATTR
- $ setfattr -n security.selinux -v "rjc:object_r:tmp_t:s0\000" foo
- $ ls -Z foo
  - -rw-rw-r-- rjc   rjc   rjc:object_r:tmp_t   foo

Kernel interfaces 3/3

- Specifying part of a context is possible with the chcon command:
- $ chcon -u user_u foo
- $ ls -Z foo
  - -rw-rw-r-- rjc rjc user_u:object_r:tmp_t foo

- Can specify that a file should have the same context as another file:
- $ chcon --reference /var/www/html foo
- $ ls -Z foo
  - -rw-rw-r-- rjc rjc system_u:object_r:httpd_sys_content_t foo

- To fix contexts for a directory tree:
- # restorecon -R /etc

Booleans

- # ps xZ|grep http
  - user_u:system_r:httpd_t 9096 ? Ss 0:00 /usr/sbin/httpd
- # setsebool httpd_disable_trans 1
- # /etc/init.d/httpd restart
- # ps xZ|grep http
  - user_u:system_r:initrc_t 9603 ? Ss 0:00 /usr/sbin/httpd
- # getsebool httpd_disable_trans
  - httpd_disable_trans --> on
- # echo 0 > /selinux/booleans/httpd_disable_trans
- # getsebool httpd_disable_trans
  - httpd_disable_trans --> on pending: off
- # echo 1 > /selinux/commit_pending_bools
- # getsebool httpd_disable_trans
  - httpd_disable_trans --> off

Distinguishing characteristics of SE Linux

- In SE Linux access is based on file context not name. Creating a hard-link to a file will not give a different level of access to the file.
- File context will be assigned to a file via an XATTR (not specified in configuration) or at mount time all files on a file system can be assigned a file type.
- Requires modified utilities and applications to use full functionality
- A simpler policy could be written which does not require modified utilities, but no-one has published such a policy yet
- Is fully configurable by policy, changes to security requirements do not require recompiling applications

Status of SE Linux in Distributions

- Red Hat Enterprise Linux 4 has targeted policy (no official support for strict and no real possibility of MLS). Before reference-policy so no MCS either.
- RHEL 5 will ship with targeted/MCS and will have MLS as an option for government use
- Fedore Core 5 has targeted, strict, and MLS
- Gentoo has strict
- Debian will have targeted, strict, and MLS soon

End of Part 1 - Q/A

- #selinux on irc.freenode.net
- http://www.nsa.gov/selinux/   Official SE Linux web site
- http://www.coker.com.au/selinux/   My SE Linux web pages

- Russell Coker <russell@coker.com.au>