Security Enhanced Linux

Thanks to David Quigley

History

SELinux Timeline

1985: LOCK (early Type Enforcement)

1990:

DTMach / DTOS

1995:

Utah Fluke / Flask

1999: 2.2 Linux Kernel (patch)

2000:

2001: 2.4 Linux Kernel (patch)

2002: LSM

2003: 2.6 Linux Kernel (mainline)

2006: Full network labeling

Present

Concepts

Type Enforcement

- *Object(s):* items in a system that are acted upon (files, IPC, sockets, etc….)
- *Subject(s):* process that are requesting access to an object
- All Objects and Subjects contain a security context
- *Security Context(s)* are composed of four parts
- All Security Context components are checked against the policy to see if access is allowed.
- Type is the base component while role and user are used to further restrict type enforcement

Security Contexts

system_u:object_r:passwd_exec_t:s0:c0.c2-s2:c0.c1

user:role:type:sensitivity[:category,…][-sensitivity[:category,…]]

TE Access Control

allow user_t bin_t : file {read execute write getattr setattr}

- *Source type(s):* The domain type of the process accessing the object
- *Target type(s):* The type of the object being accessed by the process
- *Object class(es):* The class of object to permit access to
- *Permission(s):* The kind of access permitted for the indicated object class

Domain Transitions

- Analogous to SetUID programs
- Joe running as user_t (untrusted user) needs to change his password. How does Joe change his password?
- allow user_t passwd_exec_t : file {getattr execute}
- allow passwd_t passwd_exec_t : file entrypoint

(A process in one domain transitions to another domain by executing an application that has the entrypoint type for the new domain)

- allow user_t passwd_t : process transition
- Main idea: restricts trusted domain passwd_t and allows user_t to transition to it.
- Implicit domain transitions provided via type_transition.

Domain Transitions (explained)

A user wants to change their password. /usr/bin/passwd is labeled with the passwd_exec_t type:

```
~]$ ls -Z /usr/bin/passwd
-rwsr-xr-x root root system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd
```

/usr/bin/passwd accesses /etc/shadow, which is labeled with the shadow_t type:

```
~]$ ls -Z /etc/shadow
-r--------. root root system_u:object_r:shadow_t:s0 /etc/shadow
```

A policy rule states that processes running in the passwd_t domain are allowed to read and write to files labeled with the shadow_t type. The shadow_t type is only applied to files that are required for a password change: /etc/gshadow, /etc/shadow.

A policy rule states that the passwd_t domain has entrypoint permission to the passwd_exec_t type. When a user runs the passwd application, the user's shell process transitions to the passwd_t domain.

A rule exists that allows (among other things) applications running in the passwd_t domain to access files labeled with the shadow_t type. /usr/bin/passwd is allowed to access /etc/shadow, and update the user's password.

Users & Roles

- First and second component of a security context
- SELinux usernames and DAC usernames are not synonymous
- Semanage is used to maintain mappings of DAC to SELinux usernames.
- Roles are collections of types geared towards a purpose
- Roles can be used to further restrict actions on the system
- SELinux usernames are granted roles in the system

MLS

- MLS portion of Security Context is composed of 4 parts
  - Low/High
  - Sensitivity/Category
- Includes syntax to define dominance of security levels
- Subjects with range of levels considered *trusted subjects*
- Implements a variation of Bell-La Padula

Architecture

LSM

- Kernel framework for security modules
- Provides a set of hooks to implement further security checks
- Usually placed after existing DAC checks and before resource access
- Implications? SELinux check is not called if the DAC fails
- Makes auditing difficult at times.

SELinux LSM Module

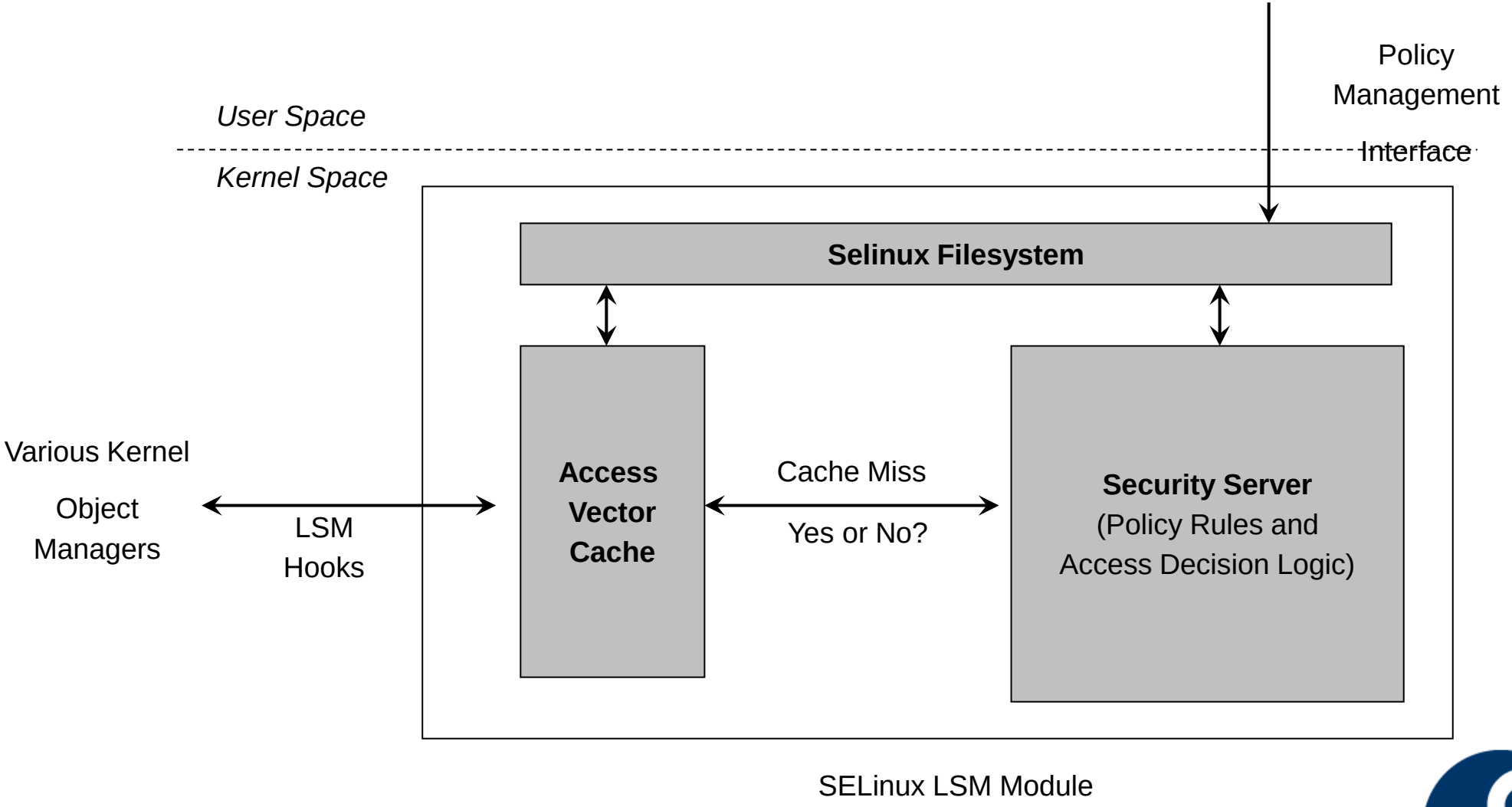

Figure taken from *SELinux by Example*

Userspace Object Managers

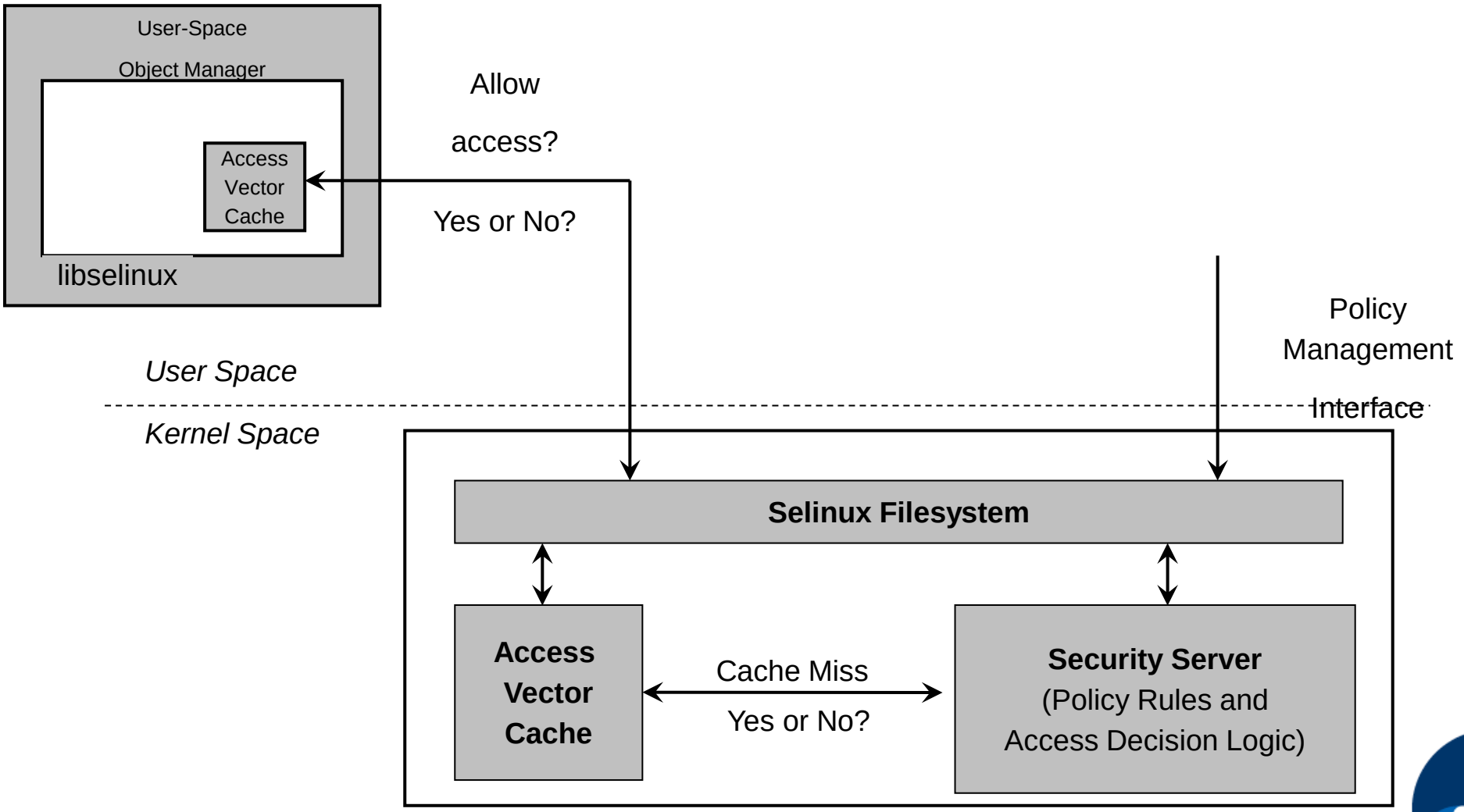

Figure taken from *SELinux by Example*

Policy Server

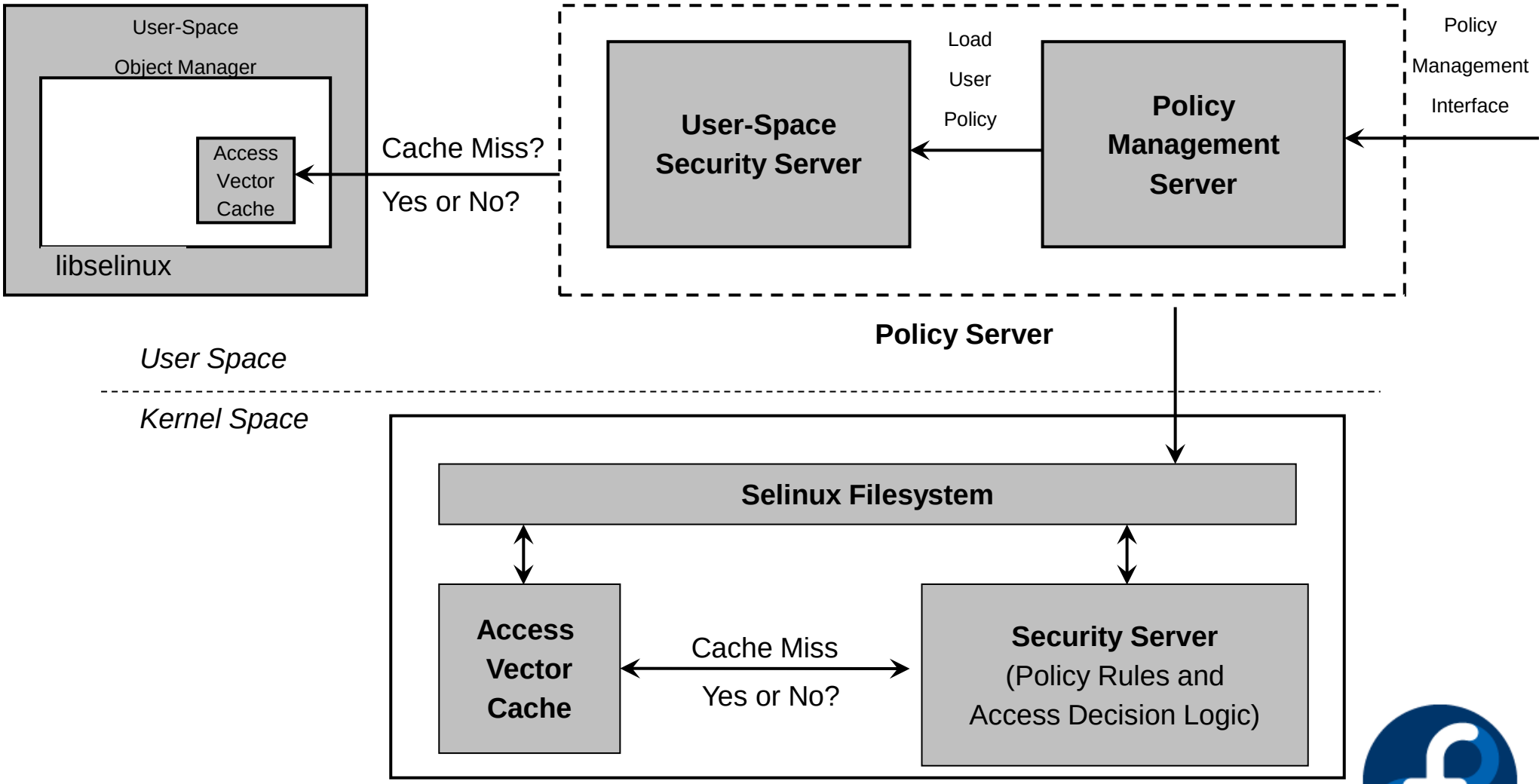

Figure taken from *SELinux by Example*

Policy Language

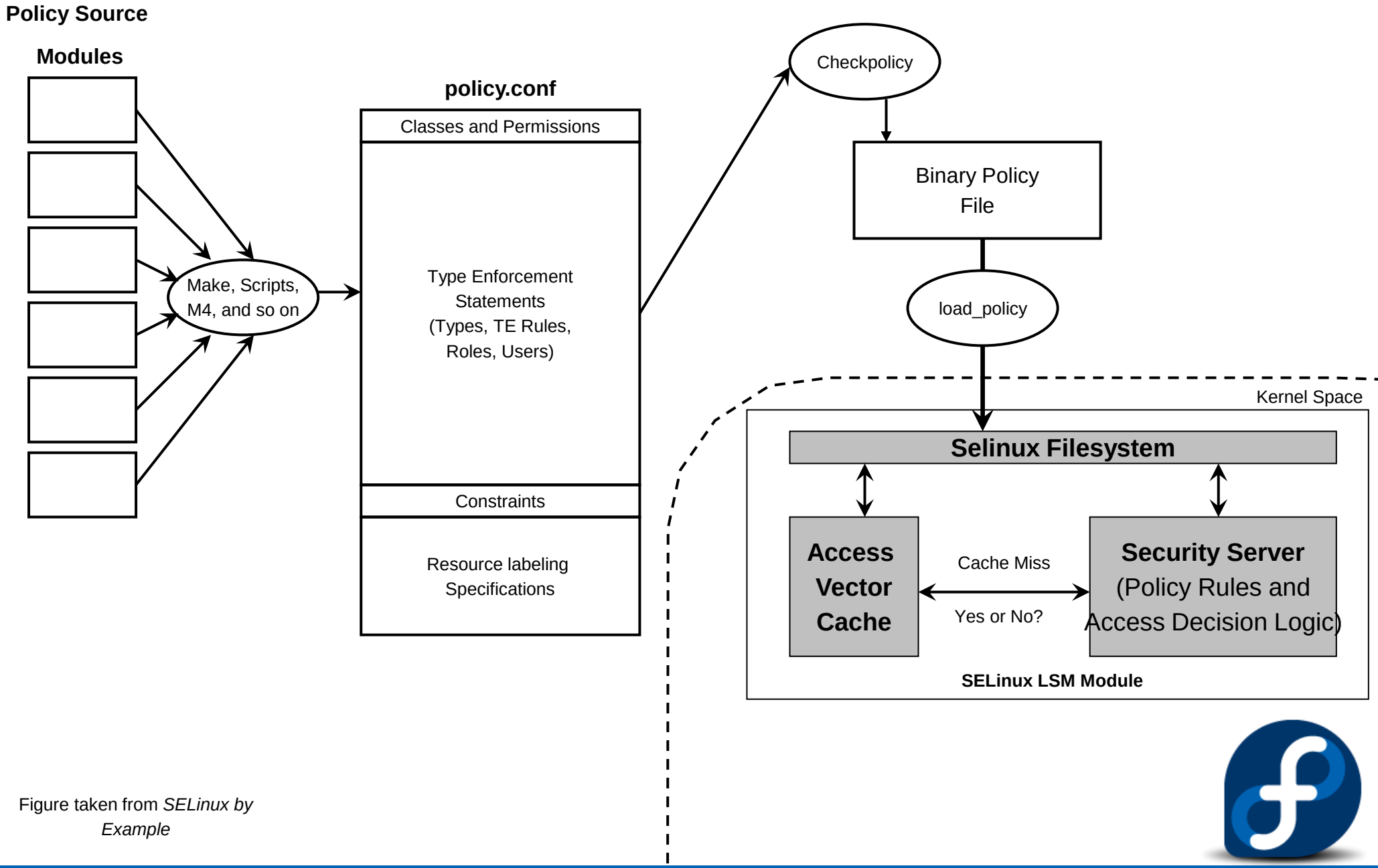

Figure taken from *SELinux by Example*

Networking

Network Labeling

- Three methods of labeling
  - netifcon (interface)
  - nodecon (host)
  - portcon (port)
- Object classes for interfaces, sockets, nodes etc.

Network Labeling: IPSEC/xfrm

- Implicit packet labeling via IPSEC/xfrm.
  - NETLINK_XFRM (xfrm = “transform”) provides an interface to manage the IPsec security association and security policy databases. It is mostly used by Key Manager daemons when they are used in Internet Key Exchange protocol.
- Security context stored in xfrm policy rules and states.
- Authorize socket's use of policy based on context.
- Build SAs with context of policy.
- Included in Linux 2.6.16.

Network Control: SECMARK

- Motivation: Existing SELinux network controls very limited in expressiveness and coverage.
- Solution: Separate labeling from enforcement.
  - Use iptables to select and label packets.
  - Use SELinux to enforce policy based on those labels.
- SECMARK and CONNSECMARK targets added.
- http://james-morris.livejournal.com/11010.html
- For 2.6.18.

Network Labeling: MLS enhancements

- Granular IPSEC associations
  - Allow a single xfrm policy rule to cover a MLS range.
  - Instantiate individual SAs for individual levels within the range.
- Flow labeling outside of socket context
  - Label based on origin when no socket involved (e.g. forward)
- Label socket IPSEC policy from socket.
- Label TCP child sockets from peer.
- In progress, see redhat-lspp and netdev lists.

Network Labeling: NetLabel

- Explicit packet labeling via IP option.
- Motivation: Compatibility with other trusted OSes.
  - Also avoids requiring use of iPSEC for labeling.
  - Also enables packet filtering based on the explicit labels.
- Presently limited to CIPSO, MLS labels.
- Code and info at http://free.linux.hp.com/~pmoore/projects/linux_cipso/

SELinux Policy Language

Object Classes

- Represents resources of a certain kind
- Policy must include declarations for all object classes
- Classes
  - File related (blk_file,chr_file,dir,fd …)
  - Network related (socket, packet_socket, rawip_socket, …)
  - IPC related (ipc, msg, msgq, sem, shm)
  - Misc Classes (capability, process, security, system)

Permissions

- Specific to a particular Object Class
- Includes traditional Linux permissions
- Extends existing permissions to be finer grained
- Includes SELinux specific permissions for labeling

Type Enforcement

- Several major keywords
  - **`type`**
  - **`attribute`**
  - **`typeattribute`**
  - **`typealias`**
  - **`allow`**
  - **`dontaudit`**
  - **`auditallow`**
  - **`neverallow`**
  - **`type_transition`**
  - **`type_change`**

RBAC

- Adds 2 components to security context
  - **`user`**
  - **`role`**
- Adds 3 policy language keywords
  - **`allow`** (different than AVC allow)
  - **`role_transition`** (similar to type_transition)
  - **`dominance`**

Multilevel Security

- Policy Declares Levels and categories
- applies constraints on objects and permissions with MLS dominance keywords
  - `==, !=, eq, dom, domby, incomp`
  - `mlsconstrain file {create relabelto } { l2 eq h2 }`
- **`mlsvalidatetrans`** transitions between levels
- Still requires a lot of work

Conditional Policies

- Allows enabling/disabling portions of policy
- Booleans define in policy
- Logical operations allowed
  - **&&**
  - **||**
  - **^**
  - **!**
  - **==**
  - **!=**
- Does not support nested conditionals
- Booleans modified through special applications or SELinuxfs

Reference Policy

- Maintained by NSA and FC Mailing Lists
- Compiles into three versions
  - Strict, Targeted, MLS
- Stats
  - Version .18
  - Object Classes 55
  - Common Permissions 3, Permission 205
  - Types 1589
  - allow 372755, auditallow 12, dontaudit 238663
  - type_transition 2657, type_change 68
  - roles 6, RBAC allow 6, role_transition 97, users 3
  - bools 70

Userspace

Components

- checkpolicy
- libselinux
- libsemanage
- libsepol
- policycoreutils

libselinux

- Used by SELinux aware applications
- Houses user space AVC
- Contains functions to
  - calculate AVCs
  - get/set/create contexts
  - query policy engine

libsemanage

- Used to query and configure state of a running system
- Provides functions to query/modify
  - login names
  - users
  - network ports/interfaces
  - file contexts
  - level translations
  - roles
  - etc.

SELinuxfs

- Interface between userspace and kernel
- Used by libselinux and libsemanage to communicate requests with the kernel
- Provides a quick and easy interface for humans
- Usually not used directly from programs

policycoreutils

- SELinux Management and policy analysis tools
  - audit2allow
  - audit2why
  - load_policy
  - newrole
  - restorecon
  - semanage
  - semodule
  - sestatus
  - setbool
  - etc...

Distributions

- Fedora Core 3 and later
- Debian
- Gentoo
- SuSe
- SE-BSD
- SE-MACH

More Information

- SELinux Homepage: www.nsa.gov/selinux
- SELinux Mailing list: http://www.nsa.gov/selinux/info/list.cfm?MenuID=41.1.1.9
- Redhat SELinux Mailing List: http://www.redhat.com/mailman/listinfo/fedora-selinux-list
- Fedora SELinux Wiki: http://fedoraproject.org/wiki/SELinux

Type Enforcement

```
attribute file_type;
attribute httpdcontent;

#These two statements...
type httpd_user_content_t;
typeattribute httpd_user_content_t file_type, httpdcontent;

#are equivalent to this one
type httpd_user_content_t, file_type, httpdcontent;

#These two statements...
type mozilla_t, domain;
typealias mozilla_t alias netscape_t;

#are equivalent to this one
type mozilla_t alias netscape_t, domain;
```

Type Enforcement

```
rule_name src_type_set target_type_set : class_set perm_set;
#valid
allow user_t bin_t : file { read getattr } ;
allow user_t bin_t : dir { read getattr search } ;

#invalid since file does not have a search permission
allow user_t bin_t { file dir } {read getattr search } ;

#dontaudit when this access is denied
dontaudit httpd_t etc_t : dir search ;

#audit when this access is allowed
#by default allowed access is not audited
auditallow domain shadow_t : file write ;

#This statement may never be allowed by any rule
neverallow user_t shadow_t : file write

allow user_t bin_t : { file dir } * ;
allow user_t bin_t : file ~{ write setattr ioctl };
```

Type Enforcement

- Type Transitions
  - type_transition
  - type_change

```
#These two statements...
type_transition user_t passwd_exec_t : process passwd_t;
type_transition sysadm_t passwd_exec_t : process passwd_t;

#are equivalent to this one
type_transition { user_t sysadm_t } : process passwd_t;

#This domain transition rule…
type_transition init_t apache_exec_t : process apache_t ;

#would require atleast the follow 3 allow rules to succeed
allow init_t apache_exec_t : file execute ;
allow init_t apache_t : process transition;
allow apache_t apache_exec_t : file entrypoint ;
```

RBAC Example

```
#valid security context

joe:user_r:passwd_t

#role user_r assigned to user joe

user joe roles { user_r };

#equivalent to this one

role user_r types { user_t passwd_t };

allow staff_r sysadm_r;

role_transition sysadm_r http_exec_t system_r;

#super_r inherits all types from sysadm_r and secadm_r

dominance { role super_r { role sysadm_r; role secadm_r; }}
```