Meaning of permissions in SELinux(Ver 1)

Yuichi Nakamura [*]

January 12, 2006

Contents

[*] The George Washington University, ynakam@gwu.edu

1 Introduction

1.1 About this document

Meaning of SELinux's permissions(operations that are restricted by permissions) are analyzed and shown in this document. For feedback, please send e-mail to ynakam@gwu.edu.

1.2 Terminology and notation

- Notation
  read(2) means, you should refer to Linux man pages. This example means *man 2 read*.

- Access vector permission
  The term *Access vector permission* means, permission defined in SELinux(such as read,write,send_msg). *Access vector permission* is often called simply *permission* in this document. The usage of the term is from [5].

1.3 Motivation

The design SELinux Policy Editor[1] is based on non-LSM based SELinux released at the time of Jan 2003. After that SELinux is re-implemented using LSM. As a result, meaning of access vector permissions had been changed, and many permissions are added. Before re-designing SELinux Policy Editor, we have to understand the meaning of permissions. However, the meaning of them is not well documented. [2] is a good documentation of implementing SELinux, but the meaning of access vectors are not fully covered. [3] is a quick reference of permissions, but the description is short. Therefore, I decided to analyze the meaning of all permissions.

2 Analysis method

The version of SELinux used is that in Linux kernel 2.6.13. Analysis of permissions are based on source code analysis of Linux 2.6.13. The process is following.

(1) Find value corresponding to the permission from security/selinux/include/av_permissions.h
In av_permissions.h, permission is defined as a constant value. For example, when we want to analyze permission *read* for object class *file*. We can find following in av_permissions.h.

```
#define FILE__READ     0x00000002UL
```

(2) Analyze how the constant is used.
In the example above, we analyze how FILE__READ is used in source code. And find out how the permission is checked. lxr [4] is useful.

In some cases above process is not enough.

- Object class capability
  For permissions in object class capability, constants CAPABILITY__* are defined in av_permissions.h. However, we can not find such constants in source code. We tend to think those are not checked, but they are actually checked. They are checked in *capable* Linux function. Let's see permission *cap_sys_admin* as an example. The permission is checked in the statement *capable(CAP_SYS_ADMIN)*, then *selinux_capable* and *task_has_capability* are called, and permission *cap_sys_admin* is checked. In the check, values defined in av_permissions.h do not appear explicitly.

- Object classes related to files and sockets
  In these object classes, some permissions are inherited from object class *file*. We have to pay attention to analyze them. Such permissions use value FILE__*. For example, when we analyze permission *read* in object class *tcp_socket*, *read* permission is inherited from *file*. We can find TCP_SOCKET__READ in av_permissions.h. However, we have to also analyze the behavior of FILE__READ.

3 Meaning of permissions

3.1 permissions related to files

In the following subsection, operations restricted by permissions are described.

3.1.1 Object classes

Object classes related to file are summarized in Table 1.

Table 1: Object classes related to file

Object class	For what kind of file?
file	Normal file
blk_file	Block device file
chr_file	Character device file
fifo_file	Special file for FIFO
lnk_file	Symbolic link
sock_file	Special file for Unix domain socket
dir	Directory

3.1.2 permissions common to object classes related to file

- ioctl
  Control attribute of device. It is checked in ioctl(2).

- read
  Read file. It is checked in read(2).

- write
  Write to file. It is checked in write(2).

- create
  Open and create new file, directory and symbolic link.

- getattr
  Get file attribute (such as last modified). It is checked in stat(2).

- setattr
  Modify file attribute. It is checked in kernel functions that changes file attribute.

- lock
  Lock file. It is checked in flock(2) and fcntl(2)

- relabelfrom, relabelto
  Relabel file. When domain A relabel file whose type is B to type C. A must have relabelfrom to B and relabelto to C.

- append
  Append to file. It is checked when opening file as append mode.

- unlink
  Delete file. It is checked in unlink(2).

- link
  Create hard link. When domain A want to create hard link for file whose type is B, A must have link permission to B.

- rename
  Rename file. It is checked in rename(2). rename(2) is used in such as mv command.

- execute
  Execute file with domain transition. Link shared library.

- swapon
  It is not used. It is defined in SELinux source as FILE__SWAPON but not used. Originally, it controlled swapon system call, but this was dropped when merged into mainline Linux kernel. For domain A to do swapon system call successfully, A need *getattr read and write* permissions. So without *swapon* permission, swapon system call can be restricted.

- quotaon
  Enable quota to disk device file. It is checked in quotactl(2)(Q_QUOTAON flag).

- mounton
  Use directory as a mount point. It is checked in mount(2).

3.1.3 permissions specfic to object class *file, blk_file*

- execute_no_trans
  Execute file without domain transition.

- entrypoint
  Use file as a entry point for domain transition.

- execmod
  Attempt to load executable in specific condition. The condition is quoted from [2] below. *It first checks whether the caller is attempting to make executable a file mapping that has had some copy-on-write done, indicating that it may include modified content. If so, then the hook function performs a file execmod permission check.*

3.1.4 permissions specific to dir

- add_name
  Add entry to directory. It is checked in rename(2) and link(2).

- remove_name
  Remove entry from directory. It is checked in unlink(2).

- reparent
  Change parent directory. It is checked in rename(2).

- search
  When opening file and directory or changing directory, *search* permission is checked. *search* is checked to all ancestor directories. For example, when *cd /etc/selinux/seedit/* command(suppose the domain is foo_t) is run, *search* is checked to /, /etc, /etc/selinux and /etc/selinux/seedit.

- rmdir
  Remove directory. It is checked in rmdir(2).

3.2 permissions related to sockets

3.2.1 Object Classes

In SELinux,object classes are related to sockets. SELinux categorizes sockets by protocol family and type. protocol family is *domain* and type is *type* in socket system call [1]. Table 2 shows relationship between object class, protocol family and type.

The short description of each sockets are below.

[1] See man socket(2)

Table 2: Object classes related to socket, partly quoted from [2]

Object class	Protocol Family	Type
tcp_socket	PF_INET, PF_INET6	SOCK_STREAM
udp_socket	PF_INET, PF_INET6	SOCK_DGRAM
rawip_socket	PF_INET, PF_INET6	SOCK_RAW
unix_stream_socket	PF_UNIX	SOCK_STREAM
unix_dgram_socket	PF_UNIX	SOCK_DGRAM
packet_socket	PF_PACKET	all
key_socket	PF_KEY	all
netlink_route_socket	PF_NETLINK	NETLINK_ROUTE
netlink_firewall_socket		NETLINK_FIREWALL
netlink_tcpdiag_socket		NETLINK_TCPDIAG
netlink_nflog_socket		NETLINK_NFLOG
netlink_xfrm_socket		NETLINK_XFRM
netlink_selinux_socket		NETLINK_SELINUX
netlink_audit_socket		NETLINK_AUDIT
netlink_ip6fw_socket		NETLINK_IP6_FW
netlink_dnrt_socket		NETLINK_DNRTMSG
netlink_kobject_uevent _socket		NETLINK_KOBJECT_UEVENT
netlink_socket		ALL other types
socket	all sockets unmatched above	

- tcp_socket, udp_socket
  These are trivial, TCP and UDP socket.

- rawip_socket, packet_socket
  These are related to socket to send raw packets. These can be used by attacker to create fake packet.

- unix_stream_socket& unix_dgram_socket
  These are unix domain socket, socket to communicate with processes in the same machine.

- netlink*socket
  These are related to netlink socket. Netlink socket is a socket to communicate with kernel.

- key_socket
  This is a socket used for IPSEC.

- socket
  Sockets that does not match all of above. From socket(2), unmatched

sockets will be those whose protocol family are PF_IPX(IPX-Novell protocols), PF_X25(ITU-T X.25 /ISO-8208 protocol), PF_AX25(Amateur radio AX.25 protocol), PF_ATMPVC(Access to raw ATM PVCs) and PF_APPLETAL(Appletalk).

3.2.2 permissions common to sockets

- relabelfrom, relabelto
  These permission is defined, bot not used.
- Target type is domain who created socket
  For following , target type is domain who created socket.
  - read
    Read data from socket. This is checked in system call recvmsg. In other system calls related to socket read(such as recvfrom), recvmsg call are internally used.
  - write
    Write data to socket. This is checked in system call sendmsg. sendmsg is internally used in other system calls related to socket write.
  - create
    Create socket. This is checked when socket is created.
  - getattr
    Get name of socket by getsockname and getpeername system call.
  - bind
    Usage of bind system call. bind system call is to give name to socket.
  - connect
    Usage of connect system call. connect system call is used to initiate network connection
  - listen, accept
    Usage of listen and accept system call. These calls are used to wait network connection.
  - getopt
    Get socket option by getsockopt system call.
  - setopt
    Set socket option by setsockopt system call.
  - shutdown
    Terminate connection by shutdown system call.
  - ioctl
    Set and get attribute of socket by ioctl system call.
  - append
    open socket with O_APPEND option,but it does not make sense for socket.

  - lock
    Lock file descriptor for socket by flock and fcntl system call.
  - setattr
    Set inode attribute of socket.

  As an example, let's see when a_t domain communicate b_t domain by TCP. For convenience of explanation, we see only create and write permissions.

  (1) a_t domain process open TCP socket, and establish connection with process whose domain is b_t
  TCP socket is created, and *create* permission is checked. The created socket has type a_t. *allow a_t a_t:tcp_socket create;* is necessary to allow it.

  (2) a_t domain write data to socket.
  write permission is checked. *allow a_t a_t:tcp_socket write;* is necessary to allow this operation. Note that target type is *not* b_t, because type of socket which a_t is writing is *a_t*.

- Target type is domain of peer socket

  - sendto
    Connect by connect call and send data by sendmsg call. Note that this permission is checked only in unix data gram socket.

  The target type of above permission is domain of peer socket. For example, when domain a_t want to send data to domain b_t, permission check is done using *domain:a_t, type:b_t, objectclass:unix_dgram_socket, permission:sendto* . This means communication between domains are checked. To allow this

  ```
  allow a_t b_t:unix_dgram_socket sendto;
  ```

  must be described in policy.

- Target type is port
  In following permissions, type of port is used as target type. In SELinux, port numbers are labeled.

  - name_bind
    Open port.
  - recv_msg
    Receive data from port. This is checked in kernel function processing incomming data. Target type is type of source port.
  - send_msg
    Send data to port. Target type is type of destination port.

For example, When a_t domain want to receive data from peer whose port is tcp 80(assuming type is http_port_t),

```
allow a_t httpd_port_t:tcp_socket recv_msg;
```

must be specified.

3.2.3 Object class netlink_socket, packet_socket, key_socket ,unix_dgram_socket

For these object classes, all permissions are the same as those specified in section 3.2.2.

3.2.4 Object class unix_stream_socket

In addition to permissions in 3.2.2, following are defined. These permissions are checked between subject domain and domain of peer.

- connectto
  Connect to peer by connect system call
- newconn
  This is not used. Defined as UNIX_STREAM_SOCKET__NEWCONN, but not actually used.
- acceptfrom
  This is not used. Defined as UNIX_STREAM_SOCKET__ACCEPTFROMN, but not actually used.

3.2.5 Object class tcp_socket

In addition to permissions in 3.2.2, following are defined.

- connectto
  Defined as TCP_SOCKET__CONNECTTO, but not used.
- newconn
  Defined as TCP_SOCKET__NEWCONN but not used.
- acceptfrom
  Defined as TCP_SOCKET__ACCEPTFROM but not used.
- node_bind
  Name socket by bind system call. Target type is type of node(Network address).
- name_connect Begin network connetion by connect system call. Target type is port number.

3.2.6 Object class udp_socket, rawip_socket

In addition to permissions in 3.2.2, node_bind is defined. The meaning of node_bind is the same as that of tcp_socket.

3.2.7 Object class netlink_nflog_socket, netlink_selinux_socket and netlink_dnrt_socket

permissions are the same as 3.2.2.

3.2.8 Object class netlink_audit_socket

In addition to permissions in 3.2.2, following permissions are defined. These permissions are checked when sending message to CAPP(Controlled Access Protection Profile)[8] audit system.

- nlmsg_read
  Send message to query the status of LauS.

- nlmsg_write
  Send message to change configuration of LauS.

- nlmsg_relay
  Send user space log message to LauS.

- nlmsg_readpriv
  Send message to obtain configuration of LauS.

3.2.9 netlink_route_socket

netlink_route_socket is used to restrict access to netlink socket that is used to configure kernel routing table. In addition to permissions in 3.2.2, following permissions are defined.

- nlmsg_read
  Send messsage to request to read kernel routing table.

- nlmsg_write
  Send message to request to write kernel routing table.

3.2.10 Object class netlink_firewall_socket

This object class is to control access to IPv4 firewall. In addition to permissions in 3.2.2, following permissions are defined.

- nlmsg_read
  This is defined but not used.

- nlmsg_write
  Send message whose mode is IPQM_VERDICT or IPQM_MODE defined in ip_queue.h.

3.2.11 Object class netlink_tcpdiag_socket

netlink_tcpdiag_socket is used to restrict usage of netlink socket for network monitoring kernel module enabled by CONFIG_IP_TCPDIAG kernel compile option.

In addition to permissions in 3.2.2, following permissions are defined.

- nlmsg_read
  Send message requesting to get infomation about TCP and DCCP protocol.

- nlmsg_write
  This is defined but not used.

3.2.12 netlink_xfrm_socket

netlink_tcpdiag_socket is used to restrict usage of netlink_xfrm_socket to configure IPSEC.In addition to permissions in 3.2.2, following permissions are defined.

- nlmsg_read
  Send message to request to read IPSEC parameter.

- nlmsg_write
  Send message to request to set IPSEC parameter.

3.2.13 Object class netlink_ip6fw_socket

This object class is defined, but not used.

3.3 permissions related to other network elements

3.3.1 Object class netif

Following permissions are defined. In these, target type is network interface[2].

- tcp_recv
  This is checked when tcp socket receives data from network interface.

- tcp_send
  This is checked when tcp socket sends data to network interface.

- udp_recv
  This is checked when udp socket receives data from network interface.

[2] SELinux labels network interface.

- udp_send
  This is checked when udp socket sends data from network interface.

- rawip_recv
  This is checked when raw socket(RAW socket and packet socket) receives data from network interface.

- rawip_send
  This is checked when raw socket sends data from network interface.

3.3.2 Object class node

Following permissions are defined. In these, target type is network node(IP address).

- tcp_recv, tcp_send, udp_recv,udp_send, rawip_recv, rawip_send
  The same as those in class netif except target type is type of node.
- enforce_dest
  Defined as NODE__ENFORCE_DEST but not used.

3.4 permissions related to IPC

3.4.1 Object classes

- ipc
  Defined SECCLASS_IPC,but not used.
- msgq
  IPC message queue. SELinux labels msgq. The type is the same as doamin of creating process.
- sem
  IPC semaphore.SELinux labels semaphore msgq. The type is the same as doamin of creating process.
- shm
  IPC shared memory. SELinux labels shared memory. The type is the same as domain of creating process.
- msg
  Message used in message queue. SELinux labels message. The type is the same as message queue to which a process is going to send to msgq.

3.4.2 permissions common to all ipc object classes

- create
  Create IPC object.

- destroy
  Destroy IPC object by shmctl(option IPC_RMID).

- getattr
  Get information about IPC by shmctl, msgctl and semctl (option IPC_STAT)

- setattr
  Change attributie of IPC object by shmctl,msgctl and semctl(option IPC_SET)

- read
  Meaning of this is different depending on object class.

  - shm
    Attach shared memory to process by using shmat SHM_RDONLY option.
  - msgq
    Read message from message queue.
  - sem
    Get value of semaphore by semctl(GETALL option) and semop.

- write
  Meaning of this is different depending on object class.

  - shm
    Attach shared memory to process by shmat not SHM_RDONLY option.
  - msgq
    Send message to message queue.
  - sem
    Change value of semaphore by semctl(SETALL option) and semop.

- associate

  - sem
    In addition to operations restricted by getattr, get id by semget.
  - shm
    In addition to operations restricted by getattr, get id by shmget
  - msgq
    Get id by msgget

- unix_read
  Operations that read ipc object. This is checked when ipcperms kernel function(with S_IRUGO flag) is called. ipcperms function with S_IRUGO flag is called when ipc object is read.

- unix_write
  Operations that write or modify ipc object. This is checked when ipcperms kernel function(with S_IWUGO flag) is called. ipcperms function with S_IWUGO flag is called when ipc object is written or modified.

3.4.3 Object class msgq

In addition to permissions common to IPC, enqueue is defined.

- enqueue
  This is the same as write.

3.4.4 Object class msg

There are only two permissions in msg. Object classes common to IPC are not used.

- send
  This is the same as write of msgq, except that target type is type of message.

- receive
  This is the same as read of msgq, except that target type is type of message.

As a target type type of message is used above. However, by default, type of message is the same as type of msgq. So, above permissions are same as write and read for msgq.

3.4.5 Object class sem

permissions are the same as those common to IPC.

3.4.6 Object class shm

In addition to permissions common to IPC, lock is defined.

- lock
  Lock shared memory by shmctl with SHM_LOCK or SHM_UNLOCK option.

3.5 Object class capability

- chown
  Change owner of file by chown.

- dac_override
  Skip ordinary Linux's permission check(DAC).

- dac_read_search
  Skip ordinary Linux's permission check about read and directory search.

- fowner

  - Skip permission check in chmod and utime

  - Change acl(Posix ACL)

- fsetid
  Some operations related to setuid.Quoted from capabilities(7): *Don't clear set-user-ID and set-group-ID bits when a file is modified; permit setting of the set-group-ID bit for a file whose GID does not match the file system or any of the supple- mentary GIDs of the calling process.*

- kill
  Skip permission check about kill. The same as CAP_KILL

- setgid
  Change GID for process and socket. Quoted from capabilities(7):*Allow arbitrary manipulations of process GIDs and supplementary GID list; allow forged GID when passing socket credentials via Unix domain sockets.*

- setuid
  Change UID for process and socket. The same as CAP_SETUID. Quoted from capabilities(7):*Allow arbitrary manipulations of process UIDs (setuid(2), etc.); allow forged UID when passing socket credentials via Unix domain sockets.*

- setpcap
  Change capability. The same as CAP_SETPCAP. Quoted from capabilities(7):*Grant or remove any capability in the caller's permitted capa- bility set to or from any other process.*

- linux_immutable
  Set immulable flag on files that support immutable flag. The same as CAP_LINUX_IMMUTABLE.

- net_bind_service
  Bind well known port. The same as CAP_NET_BIND_SERVICE.

- net_broadcast
  Not used.

- net_admin
  The same as CAP_NET_ADMIN. Quoted from capabilities(7):*Allow various network-related operations (e.g., setting privi- leged socket options, enabling multicasting, interface configu- ration, modifying routing tables).*

- net_raw
  Use raw and packet sockets.

- ipc_lock
  Memory lock using mlock, mlockall, shmctl. The same as CAP_IPC_LOCK.

- ipc_owner
  Skip permision check about IPC. The same as CAP_IPC_OWNER.

- sys_module
  Load and unload kernel module. The same as CAP_SYS_MODULE.

- sys_rawio
  Manipulate I/O port by iopl and ioperm. Access /proc/kcore. The same as CAP_SYS_RAWIO.

- sys_chroot
  Use chroot system call.

- sys_ptrace
  Use ptrace to all processes.

- sys_pacct
  Obtain log of process by acct(2).

- sys_admin
  It grants many operations.

  - Usage of following system call: quotactl, mount, umount, swapon, swapoff, sethostname, setdomainname
  - Set attribute to all IPC objects(IPC_SET)
  - Delete all IPC objects(IPC_RMID)
  - Set extended security attibute for file system.
  - Use fake UID as socket credential.
  - Can open more file than limits in /proc/sys/fs/file-max.
  - Allocate memory using space reserved for priviledged process.
    It is checked in security_vm_enough_memory LSM hook function. security_vm_enough_memory LSM hook is called in case such as when process is created. The operation is not audited in SELinux.

  - Get/set xattr trusted attribute
    Xattr trusted attribute is not used for current SELinux.
  - Some ioctl operations
    Developpers of drivers check this capability in some option of ioctl. The check is inserted by developper of driver in place where he thinks important.

- sys_boot
  Reboot by reboot(2). However, it does not restrict reboot by writing /dev/initctl.

- sys_nice
  Increase nice and change nice for other processes.

- sys_resource

  - Ignore hardlimit for resource usage in rlimit
  - Increase hardlimit for resoure usage in rlimit.
  - Use reserved space in ext2 file system
  - Modify journal data flag for ext3 by ioctl
  - Ignore limit related to message queue in /proc/sys/kernel/msgmnb

- sys_time
  Modify system clock.

- sys_tty_config
  Close control terminal by vhangup(2). Change configuration of terminal(such as keycode) by ioctl(such as KDSKBENT, KDSKBSENT option).

- mknod
  Create device file by mknod.

- lease
  Set lease by fcntl system call. Lease is a kind of lock.When a process sets lease to file, not only file is locked but also signal is sent when other process accesses the file. To use lease, file:lock should also be allowed.

- audit_write
  Send user space AVC message to kernel. User space AVC message is not used in currently SELinux.

- audit_control
  Change configuration of Linux Auditing subsystem(LauS)[7] To change /proc/self/loginuid.

3.6 Object class fd

- use

  - Inherit file descriptor when process is executed and domain has been changed.
  - Receive fd from another process by Unix domain socket[3].
  - Get and set attribute of file descriptor,such as owner and flag by fntl and ioctl.

3.7 Object class filesystem

SELinux labels superblock of filesystem. permissions in object class filesystem is used for access control to superblock.

- mount
  Mount filesystem.

- remount
  Remount existing mount by MS_REMOUNT option of mount(2).

- unmount
  Unmount filesystem.

- getattr
  Obtain statistics about filesystem, such as free block by statfs(2).

- associate
  Use type as label for files. A type can not be labeled to file unless the type is not associated to file. For example, when we want to use homepage_t to /var/www, and ext3 filesystem is labeled as fs_t, then, *allow homepage_t fs_t filesystem: associate;* must be described in policy.

- quotaget
  Get quota information .

- quotamod
  Modify quota by quotactl(2).

- relabelfrom,relabelto,transition
  These are defined in source but are not used.

[3] When creating unix domain socket, by setting SCM_RIGHTS flag, file descriptor can be sent, see man unix(7).

3.8 Object class process

permissions in object class are prepared to restrict operations between process. Unless specified, target type is domain of peer process.

- fork
  Create new process by fork(2). Target type is the domain itself.

- transition
  Do domain transition.

- sigchld, sigkill, sigstop, signull, signal
  Send signal. *sigchld* is for SIGCHLD, *sigkill* is for SIGKILL, *sigstop* is for SIGSTOP and *signull* is for signal number zero. *signal* is for other signals.

- ptrace
  Trace process by ptrace(2).

- getsched
  Read scheduling information of process(such as nice value). Session ID is used for job control by shell.

- setsched
  Modify scheduling information of process.

- getsession
  Get session ID of process.

- getpgid
  Get process group ID. Process group ID is used for job control by shell.

- setpgid
  Modify process group ID.

- getcap
  Get capability information of process by capget(2).

- setcap
  Modify capability information of process by capset(2).

- share
  Execute process with domain transition after clone system call.

- getattr
  Read process security information(such as what domain is given) in /proc/pid/attr.

- setexec
  Set security context of executed process by writing /proc/self/attr/exec or by setexecon system call.

- setfscreate
  Set security context of created file by writing /proc/self/attr/fscreate or setfscreatecon system call.

- noatsecure
  This permission is used for glibc's extended mode(secure mode). When this permission is denied, glibc secure mode is enabled(if secure mode exists).

- siginh
  Inherit signal state(such as signal handler) from parent process. This is checked when domain has been changed. The default behavior of Linux is to inherit signal state(signal handler is not inherited in exec), but by denying this permission, we can restrict inheriting signal state. If this is denied, signal state is cleared.

- setrlimit
  Change rlimit information(resource usage limit) by setrlimit(2).

- rlimitinh
  Inherit rlimit information(resource usage limit information) from parent process.This is checked when domain has been changed. The default behavior of Linux is to rlimit information, but by denying this permission, we can restrict inheriting rlimit information. If this is denied, rlimit is cleared.

- dyntransition
  Do dynamic domain transition.

- setcurrent
  Set target domain of dynamic domain transition by writing /proc/self/current.

- execmem, execstack, execheap
  These are useful in combination with Exec Shield[6]. These restrict Exec Shield to be disabled. For more, see Stephen Smalley's post to SELinux Mailing List[4].

3.9 Object class security

Object class security is operations related to query security server [5], changing SELinux internal parameters and managing SELinux. The meaning are found by analyzing selinuxfs.c.

- compute_av
  Query security server about access is denied or granted, by writing /selinux/access.

[4] http://marc.theaimsgroup.com/?l=selinux&m=113440812327410&w=2

[5] Security Server a component of SELinux which makes access control decision based on policy

- compute_create
  Query security server about label transision rule, by writing /selinux/create.

- compute_member
  Query security server about polyinstantiation[9] membership decision, by writing /selinux/member.

- check_context
  Query security server about whether security context is valid, by writing /selinux/context.

- load_policy
  Load policy file to kernel.

- compute_relabel
  Query security server about relabel based on type_change TE rule. *type_change* is a rule to help application to relabel object such as tty device.

- compute_user
  Query security server about users that a context can reach, by writing /selinux/user. Changing user identity is restricted in policy by constraints. This is used programs who change SELinux user identity such as login and ssh.

- setenforce
  Switch enforcing/permissive mode.

- setbool
  Change boolean parameter of policy.

- setsecparam
  Configure avc parameter by writing /selinux/avc.

- setcheckreqprot
  Configure behavior of permission *execmem, execmod and execheap* via /selinux/checkreqprot [6]

3.10 Object class system

In object class system, misc permissions related to system are defined.

- ipc_info
  Get information about IPC object. This is to get system-wide IPC parameter, not information specific to a IPC object. An example of system-wide IPC information is segment size of shared-memory. More precisely, this controls usage of option IPC_INFO, SHM_INFO, SEM_INFO, MSG_INFO in shmctl,semctl,msgctl system call.

[6] By writing 0 or 1 /selinux/checkreqprot, behavior of execmem, execmod and execheap can be configured.

- syslog_read
  Read kernel message by syslog(2)(option 3).

- syslog_console
  Control output of kernel message to console by syslog(2)(option 6,7,8).

- syslog_mod
  Clear kernel message buffer by syslog(2)(option 0,1,2,4,5).

Acknoledgements

Discussion on NSA's SELinux list, especially Stephen Smalley's comment was helpful to analyze what access vectors are unused.

References

[1] SELinux Policy Editor, URL=http://seedit.sourceforge.net/

[2] Stephen Smalley, Implementing SELinux as a Linux Secuity Module, URL=http://www.nsa.gov/selinux/papers

[3] An Overview of Object Classes and Permissions, Tresys Technology, URL=http://tresys.com/selinux/obj_perms_help.shtml

[4] Linux Cross-Reference, URL=http://lxr.linux.no/

[5] Stephen Smalley, Configuring the SELinux Policy, URL=http://www.nsa.gov/selinux/info/docs.cfm

[6] Arjan van de Ven, New Security Enhancements in Redhat Enterprize Linux, URL=http://www.redhat.com/f/pdf/rhel/WHP0006US_Execshield.pdf

[7] Linux manpage auditd, auditctl, ausearch

[8] Controlled Access Protection Profile URL=http://niap.nist.gov/cc-scheme/pp/PP_CAPP_V1.d.pdf

[9] SELinux Mailing List archive, URL=http://www.nsa.gov/selinux/list-archive/0505/11351.cfm