Configuring SELinux by Simplified Policy ver 1.2

Yuichi Nakamura *

September 1, 2005

Contents

*The George Washington University, Hitachi Software, ynakam@gwu.edu

1 About this document

This document is a reference manual for simplified policy(version 1.2.0). About install, see How to install. Note that this document is under construction, the contents will change.

2 Overview

SELinux[1] implements fine-grained Mandatory Access Control on Linux. However, the access control is too fine-grained and its policy tends to be too complicated. So it is very difficult to understand and configure policy. "Simplified policy" is a policy written in simplified policy language. Simplified policy language reduces the number of policy description by hiding type label from users and integrating object classes and access vectors. User can use SELinux system easily by using this. For example, if httpd_t domain want to read /var/www and use tcp port 80, the configuration is like below.

```
{
domain httpd_t;
allow /var/www r,s;
allownet -tcp -port 80;
```

Simplified policy was originally developed as a part of SELinux Policy Editor[2][3] by Hitachi Software[4]. Now it is maintained in SELinux Policy Editor Project[5]. Fedora Core4 and 3 are supported. Simplified Policy does not affect existing SELinux, you can go back default SELinux easily. Feel free to try!

3 Components of Simplified policy

Simplified Policy is composed of two *must* components, Converter and sample policy. In addition, there is an optional component, GUI(SELinux Policy Editor).

1. converter
   A compiler for simplified policy. This reads simplified policy and generates SELinux policy understandable by m4 and checkpolicy.

2. sample policy
   A sample simplified policy.

3. GUI(SELinux Policy Editor)(optional)
   A Web-based GUI to edit simplified policy. By using GUI, SELinux becomes much easier. Simplified policy with GUI is called *SELinux Policy Editor*.

4 Notice

- Syntax will change in next version. We are reviewing the security of simplified policy. As a result, syntax may change.
- Some permissions are not supported. See 5.2.3.
- Some filesystems are not supported. It is treated as unlabeled_t. To access unsupported file system, use allowadm unlabel.

5 Sample Simplified Policy

This section describes information about sample simplified policy. If you are in a hurry, read 5.1, 5.2.1 and 5.3.

5.1 Default configuration in simplified policy

- Supported services in version 1.0.0
  Supported services in sample policy are "auditd, syslogd, httpd, webmin, iptables, network". They run as *servicename*_t domain.
- RBAC configuration
  Three roles sysadm_r, staff_r, user_r are prepared by default.
  - sysadm_r
    Can do everything. only user root can use this.
  - staff_r
    Have limited access rights. Can use su command. can read /root. only root can use by default.
  - user_r
    Have limited access rights. Can not use su command. Default role for every user.

5.2 Contents of directory for simplified policy

Sample simplified policy is located at "/etc/selinux/seedit/src/policy/". Some directories and files are located there.

5.2.1 simplified_policy

This is the most important. At the directory sample simplified policy is stored. For detail of syntax for simplified policy, see 8.
Sample simplified policy are described in *global*, and *domain*.te

- global
  This is a configuration commonly used by all domains. Be careful that some access rights(For example, tmpfs usage, tty device access) are granted by default to show that SELinux can become easy.

- *domain*.a
  Here configuration for domains are described. For example,in file *httpd_t.a*, configuration for *httpd_t* domain is described.

- all
  global, and *.a are jointed. converter reads this file. This file is automatically generated. Do not edit.

5.2.2 Makefile

This is Makefile to compile simplified policy and to load policy to kernel. See 5.3.

5.2.3 base_policy

Files in this directory is used by converter to generate SELinux policy.

- default.te
  This file is useful. Statements described in this file is included in policy to be generated by converter. In this file, allow statements allowed by default in simplified policy is described. This means you can know which access vectors and object classes are not supported by simplified policy. For example, look at next line.

  ```
  allow global self:capability ~{ net_raw net_bind_service
  net_admin sys_boot sys_module sys_rawio sys_ptrace sys_chroot };
  ```

  global attribute is used in generated SELinux policy. global attribute is attached to every domain. So this means, every domain is allowed to use capability other than net_raw net_bind_service net_admin sys_boot etc. This means, simplified policy language does not support those access vectors.

  In addition, you can write original SELinux's rules here. To write auditallow rule is a good idea. But allow rules must not be written here, because it can break the security of generated policy.

- attribute.te
  In this file, attributes used in policy to be generated by converter is described. Do not edit.

- types.te
  In this file, types used in policy generated by converter is described. Do not edit.

The meaning of below files are the same as original SELinux's policy. Do not edit them.

- genfs_contexts
  File system that are unlabeled are not supported.
- security_classes
- access_vectors
- initial_sid_contexts
- fs_use
- initial_sids

5.2.4 macros

Macros to generate SELinux policy is stored. Converter generates policy including macros. In make, macros are processed by m4 and policy.conf is generated.

5.2.5 sepolicy

Generated SELinux policy is written in this directory.

- test.conf
  Policy that includes macros.
- policy.conf Policy file that is processed by m4. This is understandable by checkpolicy.
- file_contexts
  This will be installed in /etc/selinux/seedit/contexts/files/file_contexts.

5.3 To load policy

If you modify policy in simplified_policy, you must load the modified policy to kernel. There are some targets in Makefile.
First, you have to go /etc/selinux/seedit/src/policy.

- make reload
  This converts simplified policy in simplified_policy dir into SELinux policy.
  (1) policy.conf and file_contexts are generated in sepolicy dir.
  (2) binary policy is created by checkpolicy from generated policy.conf.
  (3) binary policy is installed in /etc/seedit/policy, file_contexts is installed in /etc/seedit/contexts/policy. And contents of base_policy/contexts is installed in /etc/seedit/contexts.
  (4) binary policy is loaded by load_policy
- make relabel
  After *make reload*, *fixfiles restore* runs.

- make diffrelabel
  This is very useful. *make relabel* relabels all files, and it takes a lot of time. On the other hand, *make diffrelabel* relabels only files whose label have changed. So it takes only some seconds. When you load policy, *make diffrelabel* is recommended.

6 Option of converter

converter is usually run by make, you do not have to use converter directly. Following is only for reference.
Usage : converter -i *infile* -b *base policy dir* -o *policyfile* -f *file_context*

- *infile*
  Simplified policy file. In make this is all.
- *base policy dir* Directory for base_policy.
- *policyfile* Generated policy file name.
- *file_contexts* Generated file_contexts fine name.

7 Example

7.1 Configuring vsftpd

As an example of configuration of daemon, let's configure vsftpd using simplified policy. In this example, configure Anonymous ftp access.
By default, the domain of vsftpd is *initrc_t* . *initrc_t* is a domain for scripts under /etc/rc.d . vsftpd is executed by /etc/rc.d/init.d/vsftpd(the domain is initrc_t) and inherits the domain.
However, this is not secure. Because initrc_t has a lots of access rights(see /etc/selinux/seedit/src/policy/simplified_policy/initrc_t.a).
In following current directory is */etc/selinux/seedit/src/policy*, and permissive mode

```
login: root
....
# newrole -r sysadm_r
# id -Z
root:sysadm_r:sysadm_t
# cd  /etc/selinux/seedit/src/policy
# setenforce 0
```

And for detail of syntax, see 8 .

7.1.1 Create domain for vsftpd

Let's give vftpd *vsftpd_t* domain.

1. Create configuration file
   Create *simplified_policy/vsftpd_t.a* .

2. Configure domain transition
   In *simplified_policy/vsftpd_t.a* write following.

   ```
   # simplified_policy/vsftpd_t.a
   {
   domain vsftpd_t;
   domain_trans initrc_t /usr/sbin/vsftpd;
   }
   ```

   In line 2, you've defined domain vsftpd_t. In line 3, you've configured domain transition, parent domain is initrc_t, entry point is /usr/sbin/vsftpd.

7.1.2 Test domain transition

When you edit configuration, you must use make command to indicate kernel change of configuration(See 5.3). In this case, type as below.

```
# make diffrelabel
```

Usually *make diffrelabel* is enough.
Restart vsftpd and check the domain of vsftpd.

```
# /etc/init.d/vsftpd restart
# ps -eZ
...
root:system_r:vsftpd_t          13621 pts/1    00:00:00 vsftpd
...
```

You can see that the domain of vsftpd is *vsftpd_t*. Domain transition is successful.

7.1.3 Protect files related to vsftpd

Protect files related to vsftpd
If you want to protect files related to domain, the best way is *deny* in global. In this case, let's protect /etc/vsftpd and /var/ftp. Add following in simplified_policy global. Note you have to add between { and }.

```
# In simplifed_policy/global
deny /etc/vsftpd;
deny /var/ftp;
```

And

```
# make diffrelabel
```

As a result, if some domain want to access /etc/vsftpd and /var/ftp, it must be allowed explicitly. e.g: If httpd_t want to read /etc/vsftpd, *allow /etc/vsftpd r;* must be described in httpd_t, if *allow /etc r;* is described, access to /etc/vsftpd is not allowed. deny is useful to mark important files.

7.1.4 Give access rights to vsftpd_t

The default access right for vsftpd_t is inherited from simplified_policy/global. It is not enough, you have to add configuration. The best way to know what is necessary is to test vsftpd on permissive mode and see SELinux log(The process is skipped in this document..). Below is a policy for vsftpd_t.

```
# simplifed_policy/vsftpd_t.a
     1  {
     2  domain vsftpd_t;
     3  domain_trans initrc_t /usr/sbin/vsftpd;
     4  # access to files related to vsftpd
     5  allow /etc/vsftpd r,s;
     6  allow /var/ftp r,s;
     7  allowonly /var/log r,w,s;
     8  # allow to communicate with syslog
     9  allow dev_log_t r,w,s;
    10  allowcom -unix syslogd_t;
    11  # allow to use tcp 20 and 21
    12  allownet;
    13  allownet -connect;
    14  allownet -tcp -port 20;
    15  allownet -tcp -port 21;
    16  #
    17  allowadm chroot;
    18  }
```

After writing this,

```
# make diffrelabel
```

Let's review the file.

- Line 5 to 7
  These are configuration to access files related to vsftpd. In line 5 and 6, giving access rights to read vsftpd configuration files and ftp public directory.
  Pay attention to line 7.

  ```
  allowonly /var/log r,w,s;
  ```

In this, we want to allow to write /var/log/xferlog. If we could configure,

```
allow /var/log/xferlog r,w,s;
```

this would be the best. However, /var/log/xferlog may be deleted by administrater, and when re-created the SELinux label information is lost. So we can not control access to /var/log/xferlog. So we used *allowonly /var/log r,w,s.* In this vsftpd_t can write all files on /var/log/, but can not write files on child directories. This is better than *allow /var/log r,w,s;*(This allows write access to all files under /var/log including child directories). Similally, for /tmp, /var/run, you can not controll access per-file, in those directories, files are deleted and re-created, SElinux label information may be lost.
If you have a enough knowledge of SELinux, you can use *allow file exclusive label;* This configures SELinux's file type transition. You can configure access control to files that are deleted and re-created. For detail see 8.8.1.

- Line 9,10
  Those are to communicate syslogd. If you want to communicate with syslogd, always include these two lines.
- Line 12-15
  These are to communicate via tcp 20 and 21.

7.1.5 Give access rigths to initrc_t

initrc_t is a type for start up script(/etc/init.d/vsftpd). This requires read access to /etc/vsftpd. But access to this file is denied in global, so you have to allow explicitly.

```
#add to simplified_policy/initrc_t.a
allow /etc/vsftpd r,s;
```

Then,

```
# make diffrelabel
```

7.1.6 Test again

Test in permissive mode and see access log. If no deny is outputted, then test in enforcing mode.

8 Specification of simplified policy language

8.1 Terms

8.1.1 Domain/Role/Global domain

- Domain
  Domain is the same as domain in SELinux. It is attached to process by domain transition.

- Role
  Role in simplified policy language is simplified. Role is identified with a domain for user shell. In simplified language, we describe access rights for role. In fact, it is giving access rights for user shell of the role. For example, when you give access right for *sysadm_r*, access right is given to *sysadm_t*(Domain for user shell of *sysadm_r*).
  Note that in generated SELinux policy, all roles can type every types. There is no syntax corresponding to *role:x:types:y* in simplified language.

- *global* domain
  Domain that is named *global* is special. Configuration described in global domain is inherited by all domains. For example, if you allow to read /etc in global domain, httpd_t, sendmail_t and all other domains can read /etc.

8.2 Default deny rule

Domain and roles are denied all permissions unless allowed by simplified policy language.

8.3 Structure of configuration by simplified language

Configuration is composed of sections. In each section, access control for domains/roles are described. Section begins with { and ends with }.

8.4 Syntax of section

{ (begin of section)
domain/role (declare domain or role, one domain or role can be declared in one section)
users (This can be used only for role)
domain_trans (Configure domain transition)
allow/deny (Describe access control for files)
allowxxx (Describe access control for resources other than file)
}

8.5 Declaring domain and role

8.5.1 declare domain

1. Syntax
   domain *domainname* ;

2. Meaning
   Declare domain. All configuration in a section is done for the domain declared by this statement. If *domainname* is *global* , configurations in the section is inherited by all other domains.

3. Constraints
   Domain name must end with *_t*(excepting global).
   This statement can not be used twice in one section.

8.5.2 declare role

1. Syntax
   role *rolename* ;

2. Meaning
   Declare role. *rolename* is associated to user by using *user* statement as shown below.

3. Constraints
   rolename must end with *_r*.

8.6 Configuring RBAC

8.6.1 user

1. Syntax
   user *user name*;

2. Meaning
   Define users who can use the role.

3. Example
   {
   role user_r;
   user root;
   user ynakam;
   ....
   Above means user root and ynakam can use user_r.

4. Constraints
   It can be used only section where role is declared.

8.7 Configuring domain transition

8.7.1 domain_trans

1. Syntax
   domain_trans *parentdomain filename-of-entrypoint*;

2. Meaning
   This defines how domain is attached to process.

3. Example
   {
   domain httpd_t;
   domain_trans initrc_t /sbin/httpd;
   ....
   Above means that when process(domain: initrc_t) executes /sbin/httpd, /sbin/httpd runs as httpd_t domain.

8.8 Configuring access control to normal files

8.8.1 allow

1. Syntax

   (a) allow *filename* | *label* [r],[w],[x],[s];

   (b) allow *directoryname* exclusive *label*;

   (c) allow *directoryname* exclusive -all [r],[w],[x],[s];

2. Meaning

   (a) Allow operation to file specified by permissions.

   (b) This is used things corresponding to SELinux's *file_type_auto_trans*. The file created under *directoryname* is identified by the *label*. To allow access to such file, use allow *label* [r],[w],[x],[s]. And if you want to protect file that is deleted and re-created(such as /etc/mtab), you have to use this. The *label* is the same as type in SELinux.
   When converter finds a file(assume the filename is A) has a type that is named *label*, it includes

   ```
   A   system_u:object_r:label
   ```

   in generated file_contexts.
   When file does not exist at the time of configuration, and want to protect file that is deleted/re-created, this is useful. For example, files under /var/run, /tmp, /var/log.

   (c) This allows to access all files on *directoryname* that are labeled by allow *directoryname* exclusive *label*;.

3. Meaning of permissions [1]
   - r: read and get attribute
   - w: write
   - x: execute
   - s: For directory, get contents of directory. For file, get attribute.
4. Example
   {
   domain httpd_t;
   ...
   allow /var/www r,s;
   ....
   httpd_t is allowed to read all files and directories under /var/www.

8.8.2 deny

1. Syntax
   deny *filename*;
2. Meaning
   In normal domain, this is used to cancel allow. For global domain This is used to explicitly deny the access. See examples below.
3. Example
   (a) Example 1
   {
   domain httpd_t;
   ...
   allow /var r,s;
   deny /var/named; ....
   httpd_t is allowed to read /var, but denied to read /var/named.
   (b) Example 2
   {
   domain global;
   deny /etc/shadow
   ...
   {
   domain httpd_t;
   ...
   allow /etc r,s;
   ...
   httpd_t is allowed to read under /etc, but denied to access /etc/shadow, because access to /etc/shadow is denied in global. If you have a important file, it is a good idea to describe deny in global.

[1] In [6], problem was pointed out. We are going to re-visit the permission.

8.8.3 allowonly

1. Syntax
   allowonly *directory name* [r],[w],[x],[s];

2. Meaning
   In *allow* access right is inherited by all sub directories. On the other hand, in allowonly access is granted for files in the directory, not granted for sub directories.

3. Example
   {
   domain httpd_t;
   ...
   allowonly /etc r,s;
   ....
   httpd_t is allowed to read under /etc, but not allowed to access sub directories such as /etc/httpd.

8.8.4 denyonly

1. Syntax
   denyonly *directory name*;

2. Meaning
   Deny access is granted for files in the directory, not deny access for sub directories.

8.8.5 Priority of allow, allowonly, deny, denyonly

1. For the same directory *allow(deny),allowonly(denyonly)* in global domain is overwritten by *allow(deny),allowonly(denyonly) in normal domain.*

   - Ex1)
     In global:allow /usr/ r;
     In a_t domain:allowonly /usr/ w;
     a_t can write to under /usr.

2. When allow or deny exists for the child directory, it overwrites allow for parent directory.

   - Ex)
     In a_t: allow /usr r; allow /usr/local w;
     a_t can read under /usr including sub directories. But it can write under /usr/local.

3. allow or deny for the same directory in the same domain
   OR operation is processed.

- Ex1)
  {
  domain httpd_t;
  allow /var/www r;
  allow /var/www w;
  httpd_t can read write under /var/www.
- Ex2)
  {
  domain httpd_t;
  allow /var/www r;
  deny /var/www;
  httpd_t can not access /var/www.

4. More notes about global domain.
   To cancel allow/deny in global domain, allow/deny must be explicitly specified.

   - Ex1)
     In global: deny /etc/shadow;
     If you want a_t domain to read /etc/shadow, you must specify *allow a_t /etc/shadow r;*.
   - Ex2)
     In global:allow /usr/local r;
     In a_t: allow /usr w;
     a_t can not write under /usr/local. If you want write access to /usr/local, you must specify *allow /usr/local w;* in a_t..
   - Ex3)
     global: allowonly /usr/local r;
     a_t: allow /usr w;
     a_t is allowonly read for /usr/local.

5. GUI will be useful to know which files a domain can access.

8.8.6 Special files

Access to following files are special.

1. /dev/tty* /dev/pts /dev/ptmx
   If you write allow for those file, this does nothing. Access control to these files must be done by allowtty and allowpts.

2. /proc, /sysfs, /selinux, /dev/tmpfs
   Allow to these files do nothing, because these files are mounted on filesystems that do not support xattr. See allowfs. For /selinux see allowadm getsecurity.

8.9 Configuring access control to network

8.9.1 allownet

1. syntax
   (a) allownet;
   (b) allownet -connect;
   (c) allownet -raw;
   (d) allownet (-tcp|-udp) -port *port number*;
   (e) allownet (-tcp|-udp) -allport;
2. meaning
   They are related to usage of network.
   (a) Allow to use tcp/ip network. This includes usage of tcp, udp socket, ports more than 1024. Note that to initiate network connection is not allowed. To allow network connection, use allownet -connect. And usage of well-known ports is not allowed.
   (b) Allow to connect network. This means to use *name_connect* and *connect* permission in SELinux.
   (c) Allow to use raw socket. Usage of raw socket is necessary for such as ICMP.
   (d) When you want to use well-known port, you have to reserve port by this.
   - Ex)
     {
     domain httpd_t;
     allownet -tcp 80;
     ...
     httpd_t has reserved tcp 80 port, and can use it.

   (e) Allow to use all unreserved well-known ports.
3. Constraints
   These can not be canceled once declared. Be careful using in global domain. If you use them in global domain, all domain has specified access right and can not be denied in individual domain.

8.10 Configuring access control to process communication

8.10.1 allowcom (network socket)

1. Syntax
   allowcom -tcp|-udp|-unix *todomain*;

2. Meaning
   Controls usage of socket in process communication. If *todomain is global* the domain can communicate with every domain.

3. Example
   {
   domain httpd_t;
   allowcom -unix syslogd_t;
   ...
   This means httpd_t can communicate with process that has syslogd_t by unix domain socket.

4. Constraints
   -tcp and -udp can not be used in kernel 2.6 based SELinux. They do nothing if specified.

8.10.2 allowcom (IPC)

1. Syntax
   allowcom -sem|-msg|-msgq|-shm|-pipe *to domain* [r],[w];

2. Meaning
   Allow to communicate with *to domain* by specified IPC.
   If *to domain* is *self*, this means IPC within domain. If *to domain* is *global* the domain can IPC to every domain.

8.10.3 allowcom(Signal)

1. Syntax
   allowcom -sig *to domain* [c],[k],[s],[n],[o];

2. Meaning
   Allow to send signal to *to domain*. [c] is sigchld, [k] is sigkill, [s] is sigstop, [n] is signull, [o] is other signals. signull is not supported, this means all domains are allowed to use signull.

8.11 Configuring access control to tty/pts devices

8.11.1 allowtty

allowtty is used to control access to tty device files(/dev/tty*). In SELinux environment, tty device files are given label according to login user's role. So tty device files should be treated differently in simplified language.

1. Syntax
   (a) allowtty -create;
   (b) allowtty *role* [r],[w];

(c) allowtty -change *role*;

2. Meaning

   (a) This is usually used in role section. Allow role to have its own tty device. At the time of login, by login program, role's tty device file is given type *role prefix*_tty_device_t.

   (b) Allow to read/write *role*'s tty device.

   (c) Allow to change label of tty device, and rename, unlink.

3. Special role
   If *role* is *general*, this means tty before labeled(The type is devtty_t and tty_device_t). If *role* is *global*, this means all tty devices.

8.11.2 allowpts

allowpts is used to control access to pseudo tty device files(/dev/pts). Device under /dev/pts is terminal for remote login and login from gdm.

1. Syntax

   (a) allowpts -create;

   (b) allowpts *role* [r],[w];

   (c) allowpts -change *role*;

2. Meaning
   The meaning the same as allowtty except that target is pseudo tty device.

8.12 Configuring access control to process information

8.12.1 allowproc

You can describe access control to process information. Process information is stored in /proc/*pid*. This can be used to hide process in ps command.

1. Syntax
   allowproc -self|-other;

2. Meaning

   (a) -self
   Access control to /proc/*pid*. /proc/*pid* is process information of program running its domain.

   (b) -other
   This means /proc/*pid* for other domain's process.

8.13 Configuring access control to files on misc file systems

SELinux can do fine-grained access control to files on filesystems that suport extended-attributes, such as ext3, ext2 and xfs. For such files, you configure access control using *allow* statement. In other filesystems, you should configure *allowfs* described in this section.

8.14 allowfs

- Syntax
  1. allowfs *name_of_filesystem* [s],[r],[x],[w];
     For *name_of_filesystem tmpfs sysfs autofs usbfs cdfs romfs ramfs dosfs smbfs nfs proc proc_kmsg proc_kcore xattrfs* can be used.
  2. allowfs *name_of_filesystem* exclusive *label*;
  3. allowfs *name_of_filesystem label* [s],[r],[x],[w];
  4. allowfs *name_of_filesystem* -all [s],[r],[x],[w];
- Meaning
  1. Allow access to files in specified system. For example, *allowfs proc s,r;* means to grant s,r access to files on proc filesystem(/proc). When you see logs whose types are *filesystem_t* , you may have to use allowfs. This means, if you find log about *read access to sysfs_t is denied*, you may add *allowfs sysfs s,r;*.
  2. In current SELinux, only tmpfs is allowed for *name_of_filesystem*. This is used things corresponding to SELinux's *file_type_auto_trans*. Files created in *name_of_filesystem* is given *label* type.
  3. In current SELinux, only tmpfs is allowed for *name_of_filesystem*. This gives access rights to all files that are labeled by file type transition.
- Notice
  In allowfs *name_of_filesystem* exclusive *label*;, *label* must be *domain prefix_name_of_filesystem_t*. For example, in httpd_t domain, *allowfs tmpfs exclusive httpd_tmpfs_t*.
- Notice about *name_of_filesystem*
  - proc filesystem
    Access control to proc file system is a little fine-grained. proc_kmsg means, /proc/kmsg, proc_kcore means /proc/kcore. proc means other files on /proc.

– xattrfs
This means filesystem that supports extended-attribute, but not configured to use SELinux's label. For example, if you format USB memory as ext3 on non-SELinux machine. Next you mount the USB memory in SELinux machine, the files on it are recognized as xattrfs. You have to use *allowfs xattrfs permissions* in such case.

– cdfs
This corresponds to iso9660 and udf filesystem.

– dosfs
This corresponds to fat, vfat, ntfs.

– smbfs
This corresponds to cifs and smbfs.

8.15 Configuring access control other administrative access rights

8.16 allowkernel

Configures priviledges to communicate and administrate kernel. For detail of what is granted see allow_admin_xxxx in macros/seedit_macros.te. For example, to analyze what is allowed in *allowkernel klog_adm* see allow_admin_klog_adm macro.

- Syntax
  allowkernel netlink|klog_read|klog_write|klog_adm|insmod;
- Meaning

1. netlink
   Allows to communicate with kernel by netlink socket.
2. klog_read
   Allows to read kernel messages by syslog(2) call. Usually it is required to use dmesg command.
3. klog_write
   Allows to send log message to audit subsystem in kernel.This is the same as capability audit_write.
4. klog_adm
   Allows to change configuration of audit in kernel. The same as capability audit_control,sys_pacct.
5. insmod
   Allows to install kernel module.

8.17 allowseop

- Syntax
  allowseop load_policy|setenforce|relabel|part_relabel|getsecurity;
- Meaning
  Allow priviledges to administrate SELinux.
  1. relabel
     Allow to relabel all files. You must also allow getsecurity and allowpriv search.
  2. part_relabel
     Allow to relabel files that the domain can write. You must also allow getsecurity.
  3. getsecurity
     Allow to get security policy decisions, by accessing /selinux.
  4. setenforce
     Allow to toggle enforcing/permissive mode.
  5. load_policy
     Allow to load policy to kernel.

8.18 allowpriv

- Syntax
  allowpriv net|boot|quotaon| swapon|mount |rawio|ptrace*mid*chroot|unlabel |memlock|nice|resource| time|devcreate|setattr|search|read |write|all
- Meaning
  Allow other priviledges.
  1. net
     Allow capability *CAP_NET_ADMIN*(Such as administrate NIC, route table).
  2. boot
     Allow capability*CAP_SYS_BOOT*. This means allow the usage of reboot system call.
  3. insmod
     Allow capability*CAP_SYS_MODULE*. This means allow to install kernel module.
  4. quotaon
     Allow to quotaon.
  5. swapon
     Allow to swapon.
  6. mount
     Allow to mount device.

7. rawio
   Allow capability *CAP_SYS_RAWIO*.This means usage of ioperm, iopl system call and access to /dev/mem.
8. ptrace
   Allow to use ptrace.
9. chroot
   Allow to use chroot.
10. unlabel
    Allow full access to unlabeled files(Files labeled as unlabeled_t).
11. memlock
    Allow capability *CAP_IPC_LOCK*. This means to lock memory.
12. nice
    Allow capability *CAP_SYS_NICE*. This means process scheduling.
13. resource
    Allow capability *CAP_SYS_RESOURCE*. This means usage of rlimit etc.
14. time
    Allow capability *CAP_SYS_TIME*. Thie means modify system clock.
15. devcreate
    Allow to create device files in directory that the domain can write. Without this, a process can not create device file on a directory even it is configured writable.
16. setattr
    Allow to setattr to files that the domain can s access. Without this setattr permission is granted in w permission.
17. search
    Allow s permission to all files.
18. read
    Allow r permission to all files.
19. write
    Allow w permission to all files.
20. all

8.19 conflict in global and domain

For allow excepting allow for file, allow can not be canceled once declared. Be careful using in global domain. If you use them in global domain, all domain has specified access right and can not denied in individual domain.

9 Contact us

For comment, suggestion, feedback, please send e-mail to seedit-admin@lists.sourceforge.net

10 TODO

- Review security of simplified policy language, especially for network and file permissions.
- Extend language to be able to include raw SELinux policy
- Support deny syntax for access control other than file.
- Prepare tutorial, more documents.

References

[1] Security-Enhanced Linux URL=http://www.nsa.gov/selinux

[2] Yuichi Nakamura and Yoshiki Sameshima, Configuration system for access control policy of Security-Enhanced Linux, Proc. of the 2003 Symposium on Cryptography and Information Security (SCIS 2003), Shizuoka, Japan, 2003, Vol. II, 831.836. URL=http://seedit.sourceforge.net/papers/scis2003paper.pdf (Japanese)

[3] Yuichi Nakamura "Simplifying Policy Management with SELinux Policy Editor", presentation on SELinux Symposium 2005, URL=http://www.selinux-symposium.org/2005/presentations/session4/4-2-nakamura.pdf

[4] Hitachi Software URL=http://www.selinux.hitachi-sk.co.jp/

[5] SELinux Policy Editor Project URL=http://seedit.sourceforge.net/

[6] Katsuya SUEYASU, Toshihiro TABATA, Kouichi SAKURAI, "On the Security of SELinux with a Simplified Policy," Proc. of the IASTED International Conference on Communication, Network, and Information Security (CNIS 2003), pp.79-84, Dec. 2003. URL=http://www.swlab.it.okayama-u.ac.jp/~tabata/research/CNIS2003_sueyasu.pdf