



16 SELinux-Anpassungen

Die von den Distributionen mitgelieferte SELinux-Richtlinie ist meist sehr universell einsetzbar und für die meisten Fälle ausreichend. Einfache Anpassungen können sehr leicht über die booleschen Variablen (siehe Kapitel 15) erfolgen. Ab und zu reicht das aber nicht.

Entweder möchten Sie, dass die Dateien in einem bestimmten Verzeichnis einen besonderen *Security-Context* erhalten, oder Sie möchten den *Apache* Webserver auf einem anderen zusätzlichen *Port* laufen lassen oder einer Applikation mehr Rechte einräumen, als sie aktuell besitzt.

Ich werde diese Aufgaben hier recht allgemein darstellen. Ausführlichere Beispiele der Anpassung finden Sie in Kapitel 19.

16.1 Verwaltung der SELinux-Benutzer

Mit dem Befehl `semanage` können Sie die wichtigsten Anpassungen mit einigen Einschränkungen vornehmen. Sie administrieren zum Beispiel die SELinux-Benutzer und ihre Rollen mit diesem Befehl. Wenn Sie möchten, dass ein bestimmter Benutzer besonders behandelt wird, können Sie ihm zunächst einen eigenen *SELinux!-User* zuordnen. Stellen Sie sich vor, der Benutzer *ralf* soll auch die neue Rolle *webadm_r* wahrnehmen dürfen. Dann können Sie das folgendermaßen erreichen:

```
[root@supergrobi policy]# semanage user --roles 'user_r webadm_r' ◀  
--prefix user --add ralf_u
```

Hiermit erzeugen Sie zunächst einen SELinux-Benutzer *ralf_u*. Der Befehl weist dem Benutzer die beiden Rollen *user_r* und *webadm_r* zu. Natürlich müssen diese beiden Rollen auch existieren. Sie können es alternativ auch mit der Rolle *sysadm_r* testen.

Anschließend weisen Sie bei dem Login dem Linux-Benutzer *ralf* den SELinux-Benutzer *ralf_u* zu.

```
[root@supergrobi policy]# semanage login --add --seuser ralf_u ralf
```

Mit dem Befehl `semanage user -l` bzw. `semanage login -l` sehen Sie die definierten SELinux-Benutzer und ihre Zuordnung zu echten Linux-Benutzern. Natürlich können Sie auch Benutzer löschen. Das funktioniert jedoch nur bei den Benutzern, die Sie selbst erzeugt haben. Benutzer, die in der Policy definiert wurden, können nicht gelöscht werden:

```
[root@supergrobi policy]# semanage user -d root /usr/sbin/semanage:
SELinux user root is defined in policy, cannot be deleted
```

Das ist allgemein der Fall auch bei Ports und Security-Contexts.

16.2 Verwaltung der Ports

Die Netzwerk-Ports können nun genauso verwaltet werden wie die Benutzer. Auch hier wird der Befehl `semanage` verwendet, und auch hier können Sie die durch die Policy definierten Ports nicht ändern oder löschen. Wenn Sie aber möchten, dass der *Apache* Webserver auf einem zusätzlichen *Port* horchen soll, können Sie das hiermit einfach erreichen.

Um dem Apache Webserver in der Policy auch den Port 81 zu erlauben, müssen Sie lediglich den folgenden Befehl benutzen:

```
[root@supergrobi policy]# semanage port --add --proto tcp --type http_port_t 81
[root@supergrobi policy]# semanage port -l | grep http_port_t
http_port_t          tcp          81, 80, 443, 488, 8008, 9050
```

Wichtig bei den Ports ist die Angabe des Protokolls. Sie müssen mit `-p` oder `--proto` das Transportprotokoll (`udp` oder `tcp`) angeben.

16.3 Verwaltung der Security-Contexts der Dateien

Schließlich können Sie mit dem Befehl `semanage` auch beeinflussen, welchen *Security-Context* welche Datei erhält (File-Context). Der zu verwendende Security-Context wird von der Policy definiert. Sie können sich alle definierten Security-Contexts mit `semanage fcontext -l` anzeigen lassen. Es ist wiederum nicht möglich, in der Policy vordefinierte File-Contexts zu ändern oder zu löschen. Sie können aber zusätzliche Definitionen hinzufügen.

Ein häufiges Problem stellt zum Beispiel beim Betrieb eines Webserver die Auslagerung der Webseiten dar. Häufig erzeugt der Administrator ein eigenes Verzeichnis als *DocumentRoot* (z.B. `/web`) in dem die Webseiten gespeichert werden. Dieses Verzeichnis erhält per Definition zunächst den File-*Security-Context* `system_u:object_r:default_t:s0`. Die Policy für den Apache erlaubt es ihm nicht, auf diese Dateien zuzugreifen. Anstatt nun dem Webserver zu erlauben, auf diese zusätzlichen Security-Contexts zuzugreifen, sollten diese Dateien so gelabelt werden, dass die vorhandene Policy den Zugriff erlaubt.

Das gelingt Ihnen mit dem Kommando `semanage`. Das Kommando unterstützt hierbei reguläre Ausdrücke. Damit nun jede Datei in diesem Verzeichnis und weiteren Unterverzeichnissen und das Verzeichnis `/web` selbst den richtigen Context erhalten, setzen Sie den folgenden Befehl ab:

```
[root@supergrobi policy]# semanage fcontext --add --type ←
    httpd_sys_content_t '/web(/.*)?'
```

Dies setzt sowohl für das Verzeichnis als auch für seinen Inhalt den Context. Wenn Sie nun mit `restorecon` den *Security-Context* reparieren, erhält das Verzeichnis den richtigen Security-Context¹. Jede weitere in dem Verzeichnis angelegte Datei erhält nun auch automatisch den richtigen Security-Context, da sich der Security-Context des Verzeichnisses vererbt.

```
[root@supergrobi policy]# restorecon -R /web
[root@supergrobi policy]# ls -Zd /web
drwxr-xr-x root root system_u:object_r:httpd_sys_content_t /web
```

16.4 Customizable Types

Es gibt *Customizable Types*. Hierbei handelt es sich um eine Liste von Typen, die bei einem *Relabeling* des Systems oder bei Verwendung des Befehls `restorecon` nicht zurückgesetzt werden. Diese Typen werden in der Datei `contexts/customizable_types` definiert. Bei der *Fedora Core 6*-Distribution enthält diese Datei bei der *Targeted-Policy* die folgenden Typen:

```
cvs_data_t
httpd_bugzilla_content_t
httpd_bugzilla_htaccess_t
httpd_bugzilla_script_exec_t
httpd_bugzilla_script_ra_t
httpd_bugzilla_script_ro_t
httpd_bugzilla_script_rw_t
httpd_squid_content_t
httpd_squid_htaccess_t
httpd_squid_script_exec_t
httpd_squid_script_ra_t
httpd_squid_script_ro_t
httpd_squid_script_rw_t
httpd_sys_content_t
httpd_sys_htaccess_t
httpd_sys_script_exec_t
httpd_sys_script_ra_t
httpd_sys_script_ro_t
httpd_sys_script_rw_t
```

¹ Wenn dies bei Ihnen nicht funktioniert, lesen Sie bitte auch Abschnitt 16.4!

```
httpd_unconfined_script_exec_t
mount_loopback_t
public_content_rw_t
public_content_t
samba_share_t
swapfile_t
xen_image_t
```

Sobald eine Datei diesen Typ besitzt, wird sie bei einem *Relabeling* oder bei Verwendung der Befehle *restorecon*, *fixfiles* oder *setfiles* nicht modifiziert.

Sie können diese Liste natürlich auch selbst erweitern.

16.5 Erweiterung der Policy

Häufig können die Änderungen mit den Befehlen *setsebool* oder *semanage* so durchgeführt werden, dass eine Erweiterung der Policy nicht nötig ist. Wenn jedoch die Applikation grundsätzlich noch nicht über die benötigten Privilegien verfügt und diese auch über boolesche Variablen nicht anschaltbar sind, dann müssen Sie die Policy erweitern.

Hierzu sollten Sie zunächst die Applikation installieren und prüfen, ob für die Applikation bereits eine Policy existiert, die nicht Ihren Ansprüchen genügt, oder ob eine komplett neue Policy für die Applikation entwickelt werden muss. Im zweiten Fall möchte ich Sie auf Teil IV verweisen, in dem Ihnen genau erklärt wird, wie Sie eine Policy bauen. Kapitel 24 gibt Ihnen dabei eine schnelle Einführung.

Hier wollen wir uns ansehen, wie eine vorhandene Policy erweitert werden kann, um einer Applikation zusätzliche Rechte einzuräumen. Als Beispiel verwende ich hier wieder die Richtlinie für den Webserver und die *PHP*-Applikation *phpSysInfo* (siehe auch Abschnitt 5.6 und Abschnitt 9.2). Hierzu laden Sie die Applikation von ihrer Homepage (<http://phpsysinfo.sf.net>) herunter und entpacken sie im Document-Root des Apache Webservers:

```
# cd /var/www/html
# tar xzf /path/phpsysinfo-<version>.tar.gz
# cd phpsysinfo
# mv config.php.new config.php
```

Testen Sie nun, ob die Applikation sich so verhält, wie Sie es wünschen. Dabei sollte SELinux abgeschaltet sein oder sich mindestens im *Permissive*-Mode befinden. Sie sollten eine Webseite wie in Abbildung 16.1 erhalten. Diese ähnelt der Abbildung 5.10. Sobald Sie nun SELinux wieder mit *setenforce* in den *Enforcing*-Mode versetzen, wird die Applikation nicht mehr alle Informationen anzeigen (siehe Abbildung 16.2). Gleichzeitig können Sie viele Meldungen von SELinux in der Protokolldatei (*/var/log/messages* oder */var/log/audit/audit.log*) verfolgen. Um nun die Richtlinien zu erweitern, müssen Sie aus den Fehlermeldungen entsprechende *allow*-Regeln erzeugen. Am einfachsten erfolgt das mit dem Befehl *audit2allow*. Dieser liest die Protokolldatei und erzeugt daraus entsprechende *Allow*-Meldungen.

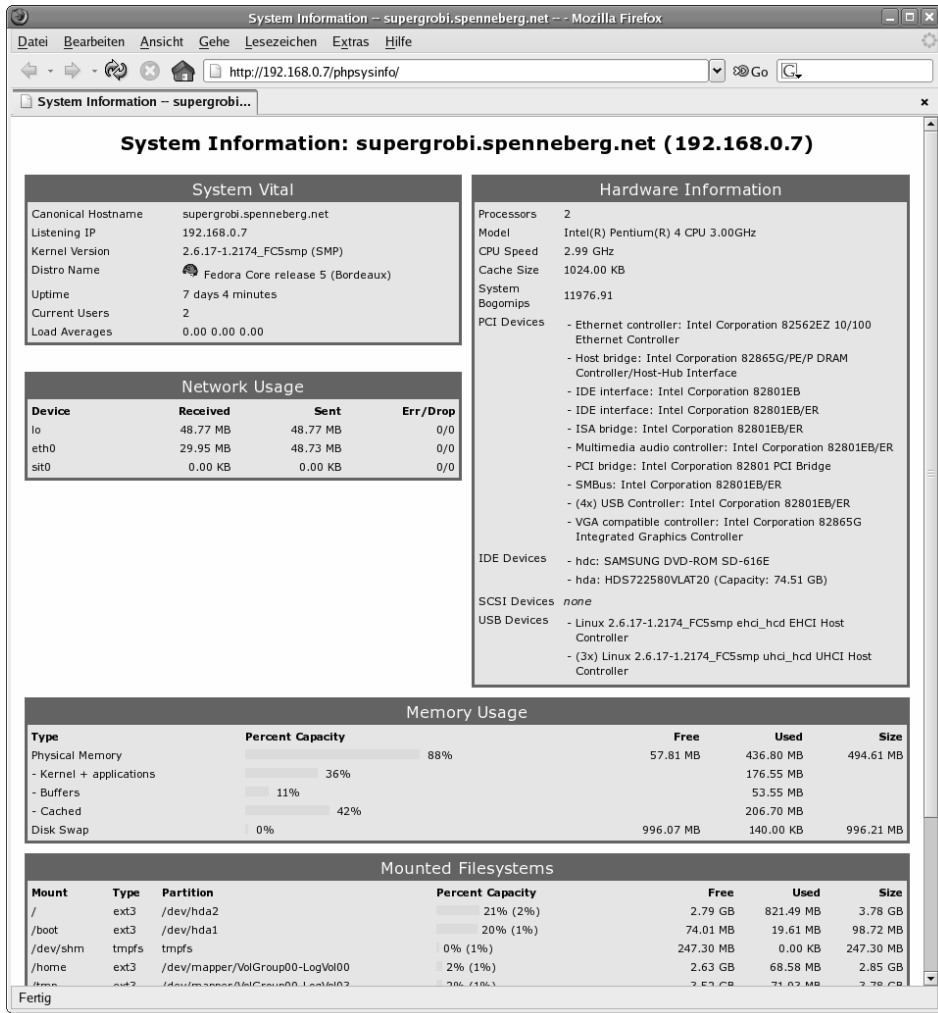


Abbildung 16.1: PHPSysInfo zeigt die Eigenschaften der Fedora Core- Installation an.

Wir werden hier einen schnellen und einfachen Weg zum Ziel nutzen. Später werde ich Ihnen noch eine bessere Alternative zeigen (siehe Kapitel 23).

Damit der Befehl `audit2allow` nicht auch noch Meldungen analysiert, die die *PHP*-Anwendung nicht betreffen, sollten Sie vor dem Einsatz die Policy neu laden. Dies führt zu einem Eintrag in dem Protokoll, und Sie können anschließend dem `audit2allow`-Kommando auftragen, nur die Meldungen zu verarbeiten, die seit dem letzten *Reload* hinzugekommen sind. Natürlich sollten Sie darauf achten, dass Sie nicht gleichzeitig auch noch andere Programme nutzen, damit nicht noch weitere unbeteiligte Meldungen verarbeitet werden.

The screenshot shows a web browser window titled "System Information - supergrob1.spenneberg.net" with the URL "http://192.168.0.7/phpsysinfo/". The main content area is divided into several sections:

- ERRORS:** A table listing four errors from "common_functions.php":

File	Line	Command	Message
common_functions.php	337	file_exists(/proc/net/dev)	the file does not exist on your machine
common_functions.php	337	file_exists(/proc/pci)	the file does not exist on your machine
common_functions.php	337	file_exists(/proc/bus/usb/devices)	the file does not exist on your machine
common_functions.php	157	find_program(mount)	program not found on the machine
- System Information: supergrob1.spenneberg.net (192.168.0.7)**
 - System Vital:** Canonical Hostname: supergrob1.spenneberg.net, Listening IP: 192.168.0.7, Kernel Version: 2.6.17-1.2174_FC5mp (SMP), Distro Name: Fedora Core release 5 (Bordeaux), Uptime: 7 days 7 minutes, Current Users: 0, Load Averages: 0.02-0.01-0.00.
 - Hardware Information:** Processors: 2, Model: Intel(R) Pentium(R) 4 CPU 3.00GHz, CPU Speed: 2.99 GHz, Cache Size: 1024.00 KB, System Bogomips: 11976.91, PCI Devices: none, IDE Devices: hdc: SAMSUNG DVD-ROM SD-616E, - hda: HDS722580VLAT20 (Capacity: 74.51 GB), SCSI Devices: none, USB Devices: none.
 - Network Usage:** Table with columns: Device, Received, Sent, Err/Drop.
 - Memory Usage:** Table with columns: Type, Percent Capacity, Free, Used, Size. Physical Memory is 89% full (55.15 MB free, 439.46 MB used, 494.61 MB size). Other memory types include Kernel + applications (36%), Buffers (11%), Cached (42%), and Disk Swap (0%).
 - Mounted Filesystems:** Table with columns: Mount, Type, Partition, Percent Capacity, Free, Used, Size. Totals: 0%.

At the bottom, there are controls for Template (classic), Language (en), and a Submit button. The footer indicates it was created by phpSysInfo-2.5.2_rc3on Aug 30, 2006 at 12:44 PM and took 0.1349 sec to load.

Abbildung 16.2: SELinux verhindert die Ausführung vieler Befehle.

Bevor wir die SELinux-Policy neu laden, versetzen wir SELinux in den *Permissive*-Mode (`setenforce 0`). In diesem Modus protokolliert SELinux jede Verletzung, verhindert sie aber nicht. Es handelt sich also um eine Art Lernmodus, wenn anschließend die Meldungen mit `audit2allow` ausgewertet werden.



Achtung

Achten Sie darauf, dass Ihr System nun überhaupt nicht von SELinux geschützt wird. Verzichten Sie daher auf Netzwerkaktivitäten, und unterbinden Sie den Zugriff von außen.

Um nun die SELinux-Policy komplett neu zu laden, benutzen Sie den Befehl `semodule`:

```
[root@supergrobi phpsysinfo]# semodule -R
[root@supergrobi phpsysinfo]# tail /var/log/audit/audit.log
...
type=AVC msg=audit(1156940576.026:3334): avc: granted ←
    { load_policy } for pid=29591 comm="load_policy" scontext= ←
    root:sysadm_r:load_policy_t:s0-s0:c0.c255 tcontext= ←
    system_u:object_r:security_t:s0 tclass=security
type=MAC_POLICY_LOAD msg=audit(1156940576.026:3334): policy ←
    loaded aid=0
...
```

Nun benutzen Sie erneut die entsprechende Applikation, deren Policy Sie erweitern möchten. Hier verwenden wir als Beispiel wieder *phpSysInfo*. Wenn Sie dabei gleichzeitig das SELinux-Protokoll beobachten, können Sie die hinzugefügten Meldungen direkt betrachten und analysieren.

Anstatt das jedoch manuell zu tun, werden wir nun den Befehl `audit2allow` verwenden. Die Anwendung ist kinderleicht. Mit der Option `-a` weisen Sie den Befehl an, sowohl die Datei `/var/log/messages` als auch die Datei `/var/log/audit/audit.log` auf SELinux-Meldungen hin zu durchsuchen². Alternativ könnten Sie mit der Option `-i <datei>` auch eine bestimmte Datei auswählen. Wenn Sie weder `-a` noch `-i <file>` angeben, liest der Befehl von der Standardeingabe. Die Option `-l` benötigen Sie, weil der Befehl lediglich die Meldungen seit dem letzten *Reload* der Policy analysieren soll. Mit der Option `-M <modul>` weisen Sie den Befehl `audit2allow` an, direkt ein SELinux Policy-Modul zu bauen, das Sie später mit `semodule` laden können.

Für unseren Anwendungsfall bedeutet das:

```
[root@supergrobi phpsysinfo]# audit2allow -a -l -M phpsysinfo
Generating type enforcement file: phpsysinfo.te
Compiling policy
checkmodule -M -m -o phpsysinfo.mod phpsysinfo.te
semodule_package -o phpsysinfo.pp -m phpsysinfo.mod
```

***** IMPORTANT *****

In order to load this newly created policy package into the kernel, you are required to execute

```
semodule -i phpsysinfo.pp
```

Mit diesen Schritten haben wir nun ein fertiges *Modul* gebaut, das direkt von uns geladen werden kann. Wenn Sie die erzeugten `allow`-Regeln noch einmal kontrollieren

² Unter Debian müssen Sie die Datei `/var/log/syslog` mit der Option `-i` angeben.

möchten, können Sie sich die Datei `phpsysinfo.te` ansehen. Diese Datei enthält die *Type-Enforcement*-Regeln. Sie erkennen das an der Endung `*.te`. Bei diesem Beispiel enthält die Datei die folgenden Regeln:

```
module phpsysinfo 1.0;

require
    class dir getattr search ;
    class file execute execute_no_trans getattr lock read ;
    type etc_runtime_t;
    type httpd_t;
    type hwdata_t;
    type initrc_var_run_t;
    type mount_exec_t;
    type proc_net_t;
    type shell_exec_t;
    type sysctl_fs_t;
    type usbfs_t;
    type var_lib_nfs_t;
    role system_r;

;

allow httpd_t etc_runtime_t:dir search;
allow httpd_t hwdata_t:dir search;
allow httpd_t hwdata_t:file getattr read ;
allow httpd_t initrc_var_run_t:file lock read ;
allow httpd_t mount_exec_t:file execute execute_no_trans read ;
allow httpd_t proc_net_t:dir getattr search ;
allow httpd_t proc_net_t:file getattr read ;
allow httpd_t shell_exec_t:file execute execute_no_trans ◀
    getattr read ;
allow httpd_t sysctl_fs_t:dir search;
allow httpd_t usbfs_t:dir getattr search ;
allow httpd_t usbfs_t:file getattr read ;
allow httpd_t var_lib_nfs_t:dir search;
```

Falls Sie manuelle Anpassungen vornehmen möchten, können Sie das Modul anschließend auch mit `checkmodule` von Hand übersetzen. Sie sollten lediglich darauf achten, dass Sie bei jeder Modifikation die *Versionsnummer* am Anfang inkrementieren.

```
[root@supergrobi phpsysinfo]# checkmodule -M -m -o phpsysinfo.mod ◀
    phpsysinfo.te
checkmodule: loading policy configuration from phpsysinfo.te
checkmodule: policy configuration loaded
checkmodule: writing binary representation (version 5) to ◀
    phpsysinfo.mod
```

```
[root@supergrobi phpsysinfo]# semodule_package -o phpsysinfo.pp ◀  
-m phpsysinfo.mod
```

Um nun dieses Modul zu laden, verwenden Sie den Befehl `semodule`. Ist bisher keine alte Version des Moduls geladen, genügt die Option `-i` für »Installation«. Ist das Modul bereits geladen und Sie möchten es durch ein Modul mit höherer Versionsnummer ersetzen, verwenden Sie die Option `-u`. Mit der Option `-r` entfernen Sie das Modul wieder.

```
[root@supergrobi phpsysinfo]# semodule -i phpsysinfo.pp  
[root@supergrobi phpsysinfo]# semodule -l | grep php  
phpsysinfo      1.0
```

Wenn Sie nun SELinux wieder in den `Enforcing`-Modus versetzen, sollte die Applikation weiterhin funktionieren. Ihr erzeugtes Modul versorgt SELinux nun mit den notwendigen zusätzlichen *Type-Enforcement*-Regeln.

