Mandatory-Access-Control (MAC) mit SELinux

An die **Kette** gelegt

Gelangen Eindringlinge erst einmal auf den eigenen Server, steht das angeblich so sichere Linux schnell nicht mehr so gut da. Mit den Root-Rechten erlangen sie gleich uneingeschränkten Zugang zu allen Systembereichen. SELinux hilft das Schlimmste verhindern. Thorsten Scherf

Linux hat den Ruf, ein besonders sicheres Betriebssystem zu sein. Jedoch schützen die klassischen Zugriffsrechte nicht vor Fehlkonfigurationen oder schlecht programmierter Software. Laufen Programm aus dem Ruder, weil der Admin vergessen hat, das letzte Update einzuspielen, oder hat ein Benutzer durch eine falsche Konfiguration erweiterte Rechte erlangt, ist guter Rat meist teuer. SELinux verspricht hier Abhilfe, denn es begrenzt zumindest den potenziellen Schaden. Es führt eine zusätzliche Zugriffskontrolle ein, die sogenannte Mandatory-Access-Control (MAC).

Gut sieben Jahre ist es nun her, seitdem die National Security Agency (NSA) [1] die erste Version von SELinux auf den Markt brachte. Damals noch als Erweiterung für Kernel 2.4 gedacht, haben diese Kernel-Patches mittlerweile Einzug in den offiziellen 2.6-Linux Kernel erhalten. Bei vielen Distributionen gehört SELinux mittlerweile zum Lieferumfang. Die in diesem Artikel vorgestellten Beispiele basieren alle auf Red Hats Community-Distribution Fedora Core 8, sind aber generell auf jeder anderen Plattform, die Unterstützung für SELinux anbietet, ebenfalls nachvollziehbar. Wichtig ist der entsprechende Support im Kernel (»CONFIG_SECURITY_SELINUX«) sowie die Pakete »libselinux«, »policycoreutils« und »selinux-policy-targeted«. Einige Standard-Pakete (»SysV-Init«, »pam«, »util-linux«, »coreutils« und andere) müssen ebenfalls für den Betrieb unter SELinux vorbereitet sein.

Kontroll-Listen

Auf klassischen Linux-Systeme kommt üblicherweise die so genannte Discretionary Access Control (DAC) zum Einsatz. Das bedeutet, dass der Eigentümer einer Datei beziehungsweise eines Objekts absolute Kontrolle über das von ihm erzeugte Objekt besitzt. Gestattet ein Besitzer versehentlich jedermann den Schreibzugriff auf eine von ihm erzeugten Datei, gibt es keine weitere Instanz, die diesen Vorgang überprüft. Oder schafft es beispielsweise ein Angreifer beliebigen Programmcode auf einem Webserver auszuführen, indem er eine Schwachstelle in der Webserver-Software ausnutzt, so läuft dieser Programmcode, beispielsweise eine Shell, mit den Rechten des Benutzers unter dem auch der Webserver läuft. Oft ist dies der Benutzer »apache«. Das heißt, der Angreifer hat Zugriff auf sämtliche Dateien, auf die auch »apache« zugreifen darf.

Auch hier gibt es keine weitere Instanz, die überprüft, ob der Webserver auf die eine oder andere Datei wirklich zugreifen muss, um seine Arbeit korrekt verrichten zu können. Da der Angreifer ja nun über einen Zugang zum System

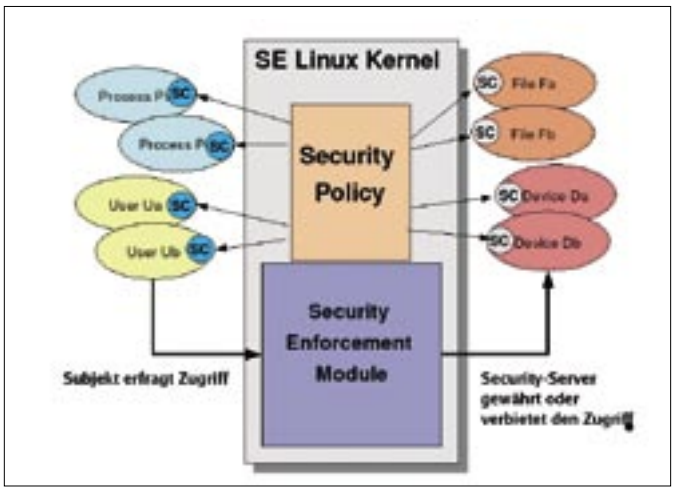

Abbildung 1: **Der Kernel-interne Security Server entscheidet über erlaubte Zugriffe.**

verfügt, kann er sich unter Umständen sogar erweiterte Rechte auf der Maschine verschaffen. Vor kurzem erst wurden viele Linux-Systeme über einen Bug im Kernel kompromittiert, den Angreifer über den Systemaufruf »vmsplice()« ausnutzen konnten. Als Folge erhielten die Angreifer komplette Kontrolle über das gesamte System. Notwendig hierfür war lediglich der Zugang zu einem unprivilegierten Konto auf der betroffenen Maschine.

Auf Systemen mit aktiviertem SELinux existiert neben der klassischen Zugangskontrolle auf Basis der DAC eine weitere Kontrollschicht, die sogenannte Mandatory Access Control (MAC). Jede Datei beziehungsweise jedes Objekt erhält ein Security-Label. Bei Datei-Objekten speichert Linux dieses Label in den erweiterten Attributen (Extended Attributes). Ebenso erhält jeder Prozess beziehungsweise jedes Subjekt ein solches Label. Ein Label, auch Security-Context genannt, besteht üblicherweise aus drei Komponenten: User:Rolle:Type/Domain. Im Folgenden zwei Beispiele für die Datei »/usr/bin/httpd« und den Prozess »httpd«:

```
# ls -lZ /usr/sbin/http
-rwxr-xr-x  root    root system_u:?
object_r:httpd_exec_t   /usr/sbin/?
httpd

# ps -AZ|grep httpd
user_u:system_r:httpd_t      2571 ?
   ?        00:00:01 httpd
```

Wie das Security-Label sind die Posix-ACLs in den erweiterten Attributen gespeichert. Im Beispiel der Datei »/usr/sbin/httpd« sehen sie so aus:

```
# getfattr -d -m security /usr/
sbin/httpd
...
security.selinux="system_u:object_?
r:httpd_exec_t\000"
```

SELinux-Funktionen

SELinux kennt drei unterschiedliche Implementierungen:

- Type-Enforcement (TE)
- Role-based Access Control (RBAC)
- Multi-Level Security (MLS)

Beim TE wird festgelegt, welches Subjekt auf welche Objekte zugreifen darf, zum Beispiel welcher Prozess auf welche Dateien. Allerdings gibt es eine Vielzahl unterschiedlicher Objekte, hierzu zählen beispielsweise auch Netzwerkports oder Speicherbereiche. SELinux prdnet jedem Subjekt eine Domäne und jedem Objekt einen Typ. Allgemein ausgedrückt regelt das Type-Enforcement also, welche Domäne auf welche Typen zugreifen darf. Es gibt bei der Darstellung keine Unterscheidung zwischen Typen und Domänen, beide enden immer auf»_t«, zum Beispiel »httpd_t«.

RBAC arbeitet mit einem abstrahierten User-Model und weist jedem Benutzer genau eine Rolle zu. Diesen Rolle lassen sich Berechtigungen zuweisen, beispielsweise auf welche Programme die Rolle Zugriff hat. Benutzer erben nun die Berechtigungen der Rolle. So ist es beispielsweise auch möglich, dem Benutzer Root eine Rolle zuzuweisen, die keine administrativen Rechte besitzt, weil sie den Zugang zu den entsprechenden Programmen verbietet. Um in eine andere Rolle mit erweiterten Rechten zu wechseln, muss der Benutzer sich erst mit Hilfe seines Passwortes authentifizieren. Das ist ein interessantes Feature, wenn man Maschinen ohne den allmächtigen Root-Benutzer aufsetzen möchte. Ein Benutzer kann sich zu einem Zeitpunkt immer nur in einer Rolle befinden, jedoch mit dem Kommando »newrole« (ähnlich »su«) die Rolle wechseln, wenn die Policy diesen Wechsel gestattet. Ein typischer Rollenname ist beispielsweise »user_r« – alle Rollen enden immer auf »_r«.

Russell Coker bietet zu Demo-Zwecken einige SELinux Play-Maschinen im Netz an [4], die die Funktion von RBAC demonstrieren. Er stellt den Root-Account für diese Maschinen öffentlich zur Verfügung, sodass sich jeder Interessierte als Administrator einloggen kann. Jeder Tester wird dann schnell feststellen, dass er einen sehr eingeschränkten Befehlsumfung zur Verfügung hat, da der Benutzer »root« sich auf diesen Play-Maschinen in einer nicht administrativen Rolle befindet.

Militärisch sicher

MLS schließlich definiert unterschiedliche Sicherheitsstufen und wird primär in extrem sicherheitskritischen Umgebungen, beispielsweise dem Militär, eingesetzt. Objekten werden hier Geheimhaltungsstufen (Vertraulich, Streng Vertraulich, Geheim und so weiter) zugewiesen und Subjekte erhalten Freigaben für diese unterschiedlichen Geheimhaltungsstufen. Kommt die MLS zum Einsatz, so wird das Security-Label um eine vierte und fünfte Komponente erweitert und hat somit folgenden Aufbau:
User:Role:Type/Domäne:Sensitivität:Kategorie

SELinux arbeitet zwar auch mit Filesystemen, die erweiterte Attribute nicht unterstützen, beispielsweise NFS oder ISO9660, hierfür muss der Admin aber in Trickkiste greifen. Das Mount-Kommando kennt für solche Fälle eine Option »context = *Security-Label*«. Hiermit lässt sich ein Security-Label für das komplette Filesystem angeben.

Policies regeln den Zugriff

Eine wichtige Komponente ist die so genannten Policy, die erlaubte Zugriffe zwischen den einzelnen Objekten und Subjekten definiert. So ist innerhalb der Policy genau festgelegt, auf welche Objekte beispielsweise ein HTTPD-Prozess mit einer bestimmten Rolle zugreifen darf. Findet ein nicht explizit erlaubter Zugriff statt, wird dieser zuerst geloggt und schließlich – zumindest im Enforcing-Mode – verboten.

Verantwortlich dafür, dass keine Verletzungen der Policy auftreten, ist der sogenannte Security-Server im Kernel. Hierbei handelt es sich um eine Instanz, die ausgehend von der eingesetzten Policy und den Security-Labeln entscheidet, ob ein Zugriff erlaubt ist. Um nicht zu starke Performance-Verluste in Kauf nehmen zu müssen, arbeitet dieser Kernel-interne Server mit einem Cache, dem sogenannten Access Vector Cache (AVC). Dank dieses Zwischenspeichers beschränkt sich die Leistungseinbuße bei SELinux-Systemen auf ungefähr sieben Prozent im Vergleich zu klassischen Systemen.

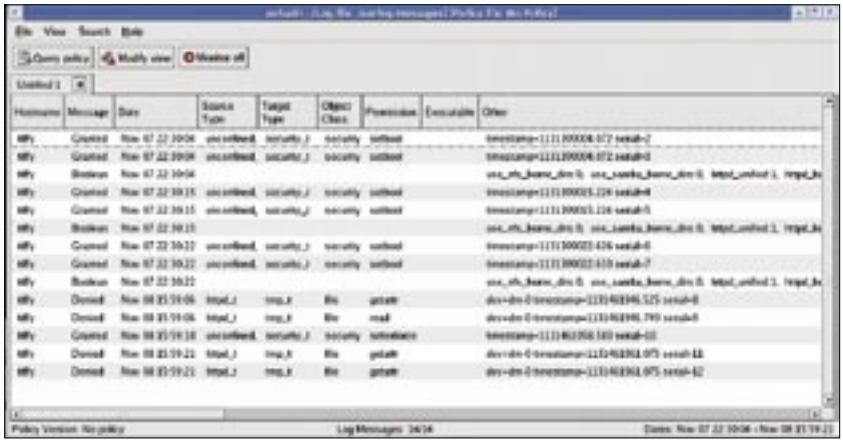

Abbildung 2: Das Tool »sedaudit« zeigt alle Log-Einträge von SELinux sortiert an.

Als kleine Demonstration zur Funktionsweise des Type-Enforcements soll folgendes Szenario dienen: Der Admin legt im Ordner »/tmp« die Datei »index.html« an. Anschließend verschiebt (nicht kopiert!) er diese Datei in das Document-Root des Webservers, bei Fedora »/var/www/html/«. Anchließlich startet er den Webserver (»/etc/init.d/httpd start«) und ruft die eben erzeugte Seite im Webbrowser auf (»http://localhost/index.html«). Ist der Enforcing-Mode von SELinux (default) aktiv, sollte der Webbrowser eine Fehlermeldung anzeigen. Zur Erklärung: Die Datei »index.html« wurde in »/tmp« angelegt, dadurch hat sie das Security-Label des Ordners »/tmp« geerbt:

```
# ls -lZ /tmp/index.html
-rw-r--r--  root     root     root:object_r:➥
tmp_t   /tmp/index.html
```

Der Webserver-Prozess läuft in einer Domäne mit dem Namen »httpd_t«:

```
# ps -AZ|grep httpd
user_u:system_r:httpd_t          2571 ?➥
        00:00:02  httpd
```

Es müsste nun nun also eine Allow-Regel in der Policy existieren, die der Domäne »httpd_t« den Zugriff auf Dateien vom Typ »tmp_t« gestattet. Um es vorwegzunehmen: Eine solche Regel existiert nicht. Warum sollte sie auch? Der Webserver muss auf seine Konfigurationsdateien zugreifen dürfen, auf seine Logs, auf CGI-Skripte und anderen Content innerhalb des Webserver-Verzeichnisses. Für all diese Dateien existieren bestimmte Typen, zum Beispiel »httpd_config_t«, »httpd_log_t«, »httpd_sys_script_exec_t« und »httpd_sys_content_t«. Dateien im Order »/tmp« zählen aber üblicherweise nicht zu den Objekten, auf die ein Webserver zugreifen muss, insofern fehlt die entsprechende Allow-Anweisung in der Policy-Datei für den Typ »tmp_t«.

Im Log »/var/log/audit/audit.log« findet sich ein Hinweis darauf, das ein nicht erlaubter Dateiaufruf stattgefunden hat:

```
... audit(1202241301.521:12): avc: denied
{getattr } for  pid=6608 comm="httpd" name=➥
"index.html" dev=dm-0 ino=179881
scontext=user_u:system_r:httpd_t tcontext=➥
root:object_r:tmp_t tclass=file
```

Im Klartext bedeutet der Log-Eintrag, das der Prozess mit der PID 6608 und dem Namen »httpd« versucht, hat den Syscall »getattr« (also das Aufrufen der Attribute) auf eine Datei mit der Inode-Nummer 179881 und dem Namen »index.html« anzuwenden. »scontext« und »tcontext« sind die Bezeichnungen der Security-Label des Source-Prozesses (»apache«) und der Target-Datei (»index.html«). Die Ausgabe »avc:denied« zeigt

Listing 1: »sealert -l 5e982b3b-3ae7-4848-b8bd-7d8553a07732«

```
Summary
    SELinux is preventing the /usr/sbin/httpd from using potentially
    mislabeled files (/var/www/html/index.html).

Detailed Description
    SELinux has denied /usr/sbin/httpd access to potentially mislabeled
    file(s) (/var/www/html/index.html).  This means that SELinux will not allow
    /usr/sbin/httpd to use these files.  It is common for users to edit files
    in their home directory or tmp directories and then move (mv) them to system
    directories.  The problem is that the files end up with the wrong file
    context which confined applications are not allowed to access.

Allowing Access
    If you want /usr/sbin/httpd to access this files, you need to relabel them
    using restorecon -v /var/www/html/index.html.  You might want to relabel
    the entire directory using restorecon -R -v /var/www/html.
...
```

an, dass der Security-Server im Kernel diese Aktion untersagt hat. Wer es gerne etwas aufgeräumter hat, der greift zum Anzeigen der Log-Einträge auf das Tool »seaudit« zurück (Abbildung 2), das die einzelnen Logeinträge grafisch dargestellt.

Neu: Verständlichere Reports

Mit dem Aufruf »seaudit-report --html *Logdatei*« ist es sogar möglich, eine HTML-Seite zu generieren, die neben den Logs auch diverse Statistiken über das SELinux-System anzeigt (Abbildung 3). Läuft auf der Maschine »setroubleshootd«, erscheint zusätzlich folgender Eintrag in der Logdatei »/var/log/messages«:

```
setroubleshoot: #012    SELinux is ➥
preventing the /usr/sbin/httpd from using ➥
potentially mislabeled files (/var/www/html/➥
index.html).#012 For complete SELinux messages➥
run sealert -l 5e982b3b-3ae7-4848-b8bd-➥
7d8553a07732
```

Der Daemon »setroubleshoot« wurde vor einiger Zeit entwickelt, um solche, etwas kryptischen Meldungen des Audit-Daemons menschenlesbar darzustellen und Tipps zur Problembeseitigung zu geben. Führt der Anwender den im Log-Eintrag angegebenen Befehl aus, erhält er die in Listing 1 sichtbare Erklärung.

Der Gnome-Desktop blendet bei jedem neu erzeugten SELinux-Log-Eintrag ein kleines Icon (ein gelbes Schutzschild) in der Taskbar ein. Klickt der Anwender auf dieses Icon, erscheint der grafische SELinux-Troubleshoot-Browser, mit dem der Admin grafisch durch die aufbereiteten Meldungen blättern kann (Abbildung 4).

Mit dieser Hilfestellung ist es nun auch ungeübten Administratoren möglich, das Problem zu beseitigen:

```
restorecon -v /var/www/html/index.html
```

Dieser Befehl setzt mit Hilfe der Policy das korrekte Label der Datei »index.html«. Alternativ kann der Admin den korrekten Datei-Typ auch manuell angeben:

```
chcon -t httpd_sys_content_t /var/www/html/➥
index.html
```

In beiden Fällen sollte das Ergebnis wie folgt aussehen:

```
# ls -lZ /var/www/html/index.html
-rw-r--r--  root root system_u:object_r:➥
httpd_sys_content_t  /var/www/html/index.html
```

Beim erneuten Versuch, die Datei im Webbrowser anzuzeigen, sollte es nun keine Sicherheitshindernisse mehr geben.

Vorher nachdenken

Anhand dieses kleinen Beispiels lässt sich sehr schön die Funktionsweise von SELinux erkennen. Unabhängig von den klassischen Berechtigungen gestattet Linux den Zugriff nur, wenn ein entsprechender Eintrag in der SELinux-Policy existiert (Mandatory Access Control). Ein solcher Eintrag werden der sicherheitsbewusste Distributor oder Admin aber nur anlegen, wenn der Zugriff wirklich notwendig ist (Principle of least Privilege).

Administration

Für die Administration eines SELinux-Systems stehen diverse Tools zur Verfügung. Beispielsweise zeigt das kleine Tool »getenforce« den aktuellen SELinux-Mode an. Mittels »setenforce 0|1« lässt sich der Mode nun natürlich auch verändern, wobei die 0 für den Permissive Mode steht und 1 für den Enforcing Mode. Permissive Mode bedeutet, dass nicht authorisierte Aktionen zwar geloggt, jedoch nicht verboten werden. Das ist beispielsweise zum Entwickeln neuer Policy-Module hilfreich, bringt aber keine zusätzliche Sicherheit. Anders sieht es beim Enforcing-Mode aus, der zusätzlich zu den Logeinträgen die nicht erlaubten Aktionen. Was erlaubt ist

Listing 2: »/etc/selinux/config«

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of
enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Abbildung 3: »seaudit-report« kann Statistiken in HTML generieren.

und was nicht, entscheidet der Security-Server anhand der Policy-Einträge. Soll der Mode dauerhaft verändern werden, so ist hierfür ein Eintrag in der Datei »/etc/selinux/config« nötig (Listing 2).

Booleans

Das interessanteste Tool in Hinblick auf SELinux ist wohl »system-config-selinux« (Abbildung 5). Hiermit lassen sich sowohl ganz grundlegende Einstellungen vornehmen, wie beispielsweise der zu verwendene SELinux-Mode, wie auch sehr umfangreichere Arbeiten durchführen, wie beispielsweise das Erstellen von neuen Policy-Modulen. Dazu später noch mehr. Desweiteren lassen sich auch so genannte Booleans konfigurieren. Booleans sind nichts anderes als Anweisungen mit denen vorbereitete Policy-Regeln aktiviert werden können, ohne das man sich hierfür mit der Makrosprache M4 (auf dieser Sprache basiert die ganze Policy) auseinanderzusetzen hat. Es gibt eine Vielzahl vordefinierter Booleans, so kann man beispielsweise dem Webserver erlauben auf Daten in den Ordnern der User zuzugreifen (UserDir) oder dem Nameserver gestatten, Änderungen an seiner Zonendatei (DDNS) durchzuführen. Auf der Kommandozeile lassen sich diese Booleans mittels getsebool anzeigen. Der Befehl »getsebool -a | grep httpd« beispielsweise listet alle Booleans für den Webserver auf (Listing 3).

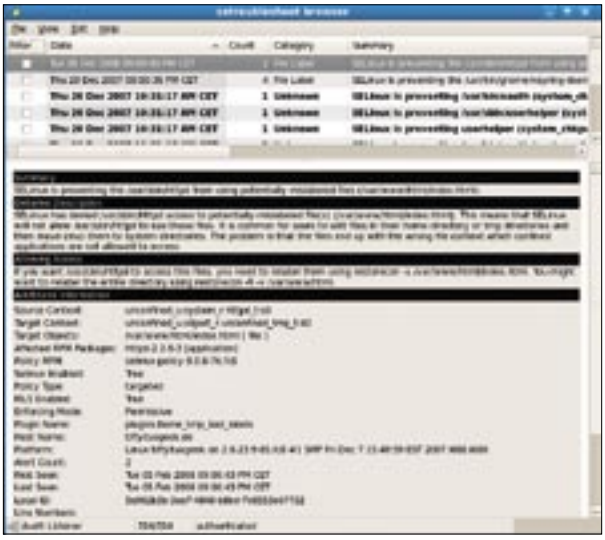

Abbildung 4: Das »setroubleshootd« kann die Logs auch grafisch anzeigen.

Listing 3: »getsebool -a | grep httpd«

```
allow_httpd_anon_write --> off
allow_httpd_dbus_avahi --> off
allow_httpd_mod_auth_pam --> off
allow_httpd_sys_script_anon_write --> off
httpd_builtin_scripting --> on
httpd_can_network_connect --> off
httpd_can_network_connect_db --> off
httpd_can_network_relay --> off
httpd_can_sendmail --> off
httpd_enable_cgi --> on
httpd_enable_ftp_server --> off
httpd_enable_homedirs --> on
httpd_ssi_exec --> off
httpd_tty_comm --> on
httpd_unified --> on
httpd_use_cifs --> off
httpd_use_nfs --> off
```

Listing 4: »sestatus -b«

```
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   permissive
Mode from config file:          permissive
Policy version:                 21
Policy from config file:        targeted

Policy booleans:
allow_console_login               off
allow_cvs_read_shadow             off
allow_daemons_dump_core           on
allow_daemons_use_tty             on
allow_execheap                    off
allow_execmem                     on
...
```

Es gibt eine Menge Man-Pages die die Booleans der gängisten Netzwerkdienste beschreiben. Die passende Man-Page für den Webserver heißt »httpd_selinux«.
Natürlich lassen sich Änderungen an diesen Booleans ebenfalls auf der Kommandozeile durchführen. Das Tool hierfür ist »setsebool«. Folgender Aufruf gestattet dem Webserver das Ausführen von CGI-Skripten:

```
setsebool -P httpd_enable_cgi 1
```

Ein weiteres interessantes Kommandozeilen-Tool heißt »sestatus«. Es liefert einen Überblick über die aktuelle SELinux-Konfiguration. So zeigt es, welche Policy gerade aktiv ist, welcher SELinux-Mode aktiv ist, und die Einstellungen der Booleans (Listing 4).

Modulare Policy

Auf aktuellen Linux-Systemem kommt mittlerweile eine Policy zum Einsatz, die sich stark von älteren Versionen unterscheidet. Hat beispielsweise RHEL 4 noch eine rein monolithische Policy verwendet, so kommt heute ausschließlich eine modulare Variante der Policy zum Einsatz. Das bringt eine Menge Vorteile mich sich. So muss sich ein Policy-Entwickler nun nicht mehr mit den kompletten Policy-Sourcen des ganzen SELinux-Systems herumschlagen, es genügt, ein einzelnes Modul für die zu schützende Applikation zu entwickeln und es dem System hinzuzufügen. Die Standard-Policy, die Teil der Fedora-Distribution ist (Targeted Policy), stellt bereits eine ganze Menge Applikationen unter das Schutzschild von SELinux. Man spricht hier von den sogenannten Targeted-Programmen (daher auch der Policy-Name). Eine Übersicht aller verfügbaren Policy-Module liefert der Befehl »semodule« (Listing 5).
Möchte der Admin ein Modul entfernen und so den SELinux-Schutz für dieses Programm aufheben, übergibt er einfach die Option »-r« und den Modul-Namen:

```
semodule -r amavis
```

Das hebt den Schutz für die angegebene Applikation dauerhaft auf. Natürlich lässt sich das Modul später auch wieder zum System hinzufügen. Das Policy-RPM speichert alle verfügbaren Standard-Module im Verzeichnis »/usr/share/selinux/targeted«. Möchte der Administrator also beispielsweise das Amavis-Modul erneut laden, ruft er »semodule« wie folgt auf:

```
semodule -i /usr/share/selinux/targeted/❯
amavis.pp
```

Dieser Aufruf lädt das Amavis-Modul wieder in den Security-Server des Kernels. Der ist dann dafür verantwortlich, anhand der Regelsätze des Moduls, Aktionen die die Amavis-Software betreffen, zu verbieten oder zuzulassen. Da

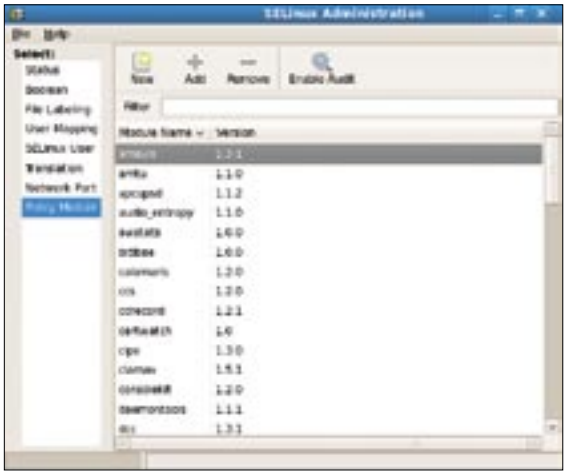

Abbildung 5: **Über das grafische Konfigurationswerkzeug »system-config-selinux« lässt sich ein SELinux-System sehr komfortabel administrieren.**

die Module im Binärformat vorliegen, lassen sich die einzelnen Anweisungen natürlich nicht nachverfolgen. Wer eine genaue Übersicht haben möchte, welche Regeln die einzelnen Module mit sich bringen, kann das SRPM (Source-RPM) der eingesetzten Policy installieren oder einfach das grafische Tool »apol« verwenden. Es ist in der Lage, die binäre Policy-Datei im Klartext darzustellen. Hiermit lässt sich die eingesetzte Policy sehr schön erforschen.

Haufenweise Regeln

Wem das zu aufwändig ist oder wer nur eine allgemeine Übersicht über die eingesetzte Policy haben möchte, dem hilft »seinfo« weiter (Listing 6). Hier lässt sich gut erkennen, wie umfangreich das gesamte Regelwerk doch ist. So kommen bei der Standard-Policy bereits mehr als 17 000 Allow-Regeln zum Einsatz.

Eine weitere neue Eigenschaft der SELinux-Policy unter Fedora 8 ist, dass sie nun auch Eigenschaften der alten Strict-Policy enthält. Genauer gesagt, ist es mit Targeted-Policy nun auch möglich, Benutzer-Konten einzuschränken, also auf die Role-based-Access-Control (RBAC) zurückzugreifen. Dan Walsh hat hierfür beispielhaft ein Policy-Modul mit dem Namen »xguest« entwickelt [5]. Hiermit kann der Administrator im Handumdrehen aus einem Gnome-Desktop ein Kiosk-Systemmachen. Der einzige User, der sich einloggen darf, ist »xguest«. Auf dem Gnome-Desktop stehen diesem Benutzer nur sehr eingeschränkte Funktionen und eine kleine Auswahl von Programmen zur Verfügung. So ist es beispielsweise nur mit Hilfe des Firefox-Webbrowsers möglich, auf das Netzwerk zuzugreifen, jeder anderen Applikation bleibt das verwehrt.

Auch Modifikationen an den Gnome-Einstellungen selbst (»gconf«) sind untersagt. Also eine ideale Umgebung für Kiosk-Systeme, wie sie oft an Flughäfen und Lobbys zu finden sind. Das Policy-Modul »xguest« kann natürlich auch als Ausgangspunkt für eigene Entwicklung dienen. Beispielsweise könnte man die bestehenden Regelsätze um weitere Anweisungen erweitern, sodass auch auch ein Zugriff via SSH möglich ist.

Entwicklung

Wer sich nicht nur passiv mit vorhandenen SELinux-Policies auseinandersetzten, sondern aktiv ins Geschehen eingreifen möchte, dem stehen hierfür unter Fedora 8 einige Tools zur Verfügung. So lässt sich beispielsweise die binäre Policy mittels »semanage« „on the Fly“ modifizieren, ohne das hierfür die Sourcen notwendig wären. Natürlich sind nicht alle Eigenschaften auf diese Weise veränderbar, aber für einfache Modifikationen reicht es allemal. Ein weiteres praktisches Beispiel soll im Folgenden die Anwendung des Management-Tools demonstrieren.

Listing 5: »semodule -l«

```
amavis          1.3.1
amtu            1.1.0
apcupsd         1.1.2
audio_entropy   1.1.0
awstats         1.0.0
bitlbee         1.0.0
calamaris       1.2.0
ccs             1.2.0
cdrecord        1.2.1
certwatch       1.0
cipe            1.3.0
clamav          1.5.1
...
```

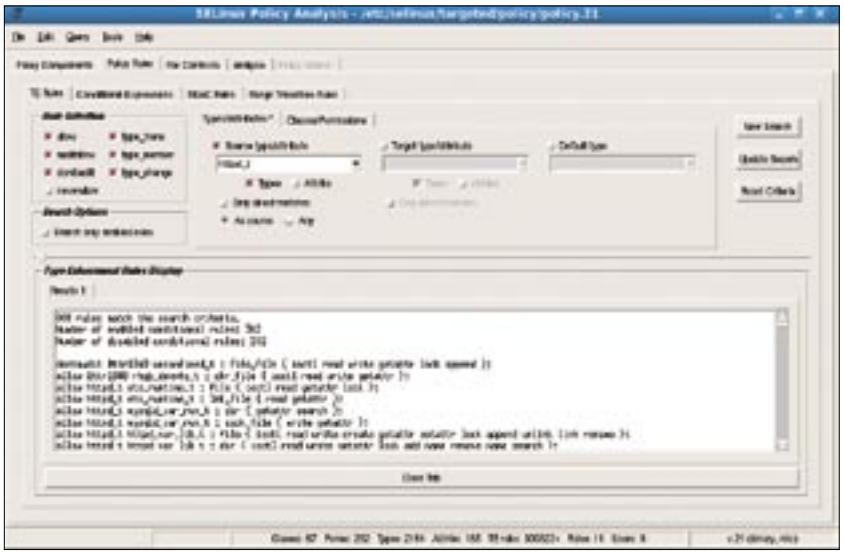

Abbildung 6: **Die grafische Anwendung »apol« ist in der Lage, die binäre Policy im Klartext anzuzeigen.**

Listing 6: »seinfo«

```
Statistics for policy file: /etc/selinux/targeted/
policy/policy.21
Policy Version & Type: v.21 (binary, mls)

   Classes:            67    Permissions:       232
   Sensitivities:       1    Categories:       1024
   Types:            2166    Attributes:        185
   Users:               8    Roles:              11
   Booleans:          115    Cond. Expr.:       144
   Allow:          171807    Neverallow:          0
   Auditallow:         28    Dontaudit:      134379
   Type_trans:       3286    Type_change:        88
   Type_member:        14    Role allow:         16
   Role_trans:          3    Range_trans:       158
   Constraints:        59    Validatetrans:       0
   Initial SIDs:       27    Fs_use:             17
   Genfscon:           66    Portcon:           292
   Netifcon:            0    Nodecon:             8
```

Abbildung 7: Mit »system-config-selinux« wird das Erzeugen neuer Policy-Module zum Kinderspiel.

Die SELinux-Policy erlaubt dem Apache-Webserver nur, sich an bestimmte Netzwerkports zu binden. Diese Ports werden in der Policy mit dem SELinux Typen »http_port_t« versehen. Ein Aufruf von »semanage« zeigt an, welche Ports über ein solches Label verfügen:

```
# semanage port -l | grep http_port_t
tcp      80, 443, 488, 8008, 8009
```

Möchte der Administrator nun einen neuen Port mit diesem Label versehen, so muss er »semanage« wie folgt aufrufen:

```
semanage port -a -t http_port_t -p tcp 777
```

Dieser Befehl versieht den TCP-Port 777 mit dem Label »httpd_port_t«. Das bestätigt der erneute Aufruf von »semanage«:

```
# semanage port -l |grep http_port_t
tcp      777, 80, 443, 488, 8008, 8009
```

Sucht der Admin mittels »apol« die Policy nach einer passenden Regel ab, wird er schnell fündig:

```
allow httpd_t http_port_t : tcp_socket?
 { name_bind name_connect };
```

Die Regel gestattet Prozessen in der »httpd_t«-Domäne den Zugriff auf alle Netzworkports, die über das Label »http_port_t« verfügen, jetzt also 777, 80, 443, 488, 8008, und 8009.

Wer noch einen Schritt weiter gehen möchte und komplett neue Module erzeugen möchte, dem stehen mit »system-config-selinux« und »policygentool« zwei Tools zur Verfügung. »policygentool« ist Teil des RPM »selinux-policy-devel« und befindet sich im Ordner »/usr/share/selinux/devel«. Es hilft beim Erzeugen der notwendigen Dateien, aus denen sich dann schließlich ein binäres Policy-Modul wird. Da eine Beschreibung aller einzelnen Schritte in dickes Buch füllen würde, hier nur ein grober Überblick der notwendigen Anweisungen. Ein ausführliches Tutorial zur SELinux-Policy-Entwicklung bietet [2].

Modul selbstgemacht

Zuerst ruft der Admin das Tool mit dem Namen der zu schützenden Applikation und dem Namen des neu zu erstellenden Policy-Moduls auf:

```
./policygentool foo /usr/bin/foo
```

Er bekommt daraufhin einige Fragen bezüglich der Applikation gestellt, beispielsweise ob ein Init-Skript zur Verfügung steht, wo die Log-Dateien liegen und so weiter. Hat er alle Fragen beantwortet, erzeugt »policygentool« die Dateien »foo.fc«, »foo.if« und »foo.te«. Hierbei handelt es sich um die File-Context-, eine Type-Enforcement- und Interface-Datei. In der File-Context-Datei kann der Admin nun alle Dateien, die zur Applikation gehören, mit einem SELinux-Label verknüpfen. In der Type-Enforcement-Datei sind die entsprechnenden Regel-Anweisungen aufzuführen, also was die Applikation alles darf. Die Interface-Datei stellt anderen Policy-Modulen Makros zur Verfügung.

Hat der Administrator die Dateien entsprechend angepasst, kann er im Anschluss das eigentliche Policy-Modul wie folgt erzeugen:

```
make -f /usr/share/selinux/devel/Makefile
```

Anschließend findet er im aktuellen Verzeichnis die neue Datei »foo.pp«, die er mit »semodule -i foo.pp« in den Security-Server des Kernels lädt.

Wem dieser ganzer Prozess zu aufwendig ist, der kann natürlich auch mit dem grafischen Frontend »system-config-selinux« neue Policy-Module erzeugen. Unter [3] findet sich auch hierfür ein sehr ausfürliches Tutorial.

Einfacher als vorher

SELinux stellt eine sinnvolle Sicherheitserweiterung aktueller Linux-Installationen dar. Einmal aktiviert, arbeitet SELinux fast transparent im Hintergrund und überwacht das laufende System – sofern der Distributor entsprechende Vorarbeit geleistet hat und eine praxistaugliche Policy mitliefert. Fedora ist hier im Moment federführend.

Auch im Bereich Usability hat sich einiges getan. So sind SELinux-Logs dank dem »setroubleshootd« mittlerweile besser lesbar als früher. Selbst ungeübte Anwender können eigene Policy-Module, mit denen sich neue Programme unter das Schutzschild von SELinux stellen lassen, mit Hilfe von »system-config-selinux« erstellen. Sie sollten trotzdem verstehen, was sie dabei machen. *(ofr)* ■

Infos

[1] NSA SELinux Website: [http://www.nsa.gov/selinux]

[2] Thorsten Scherf, SELinux-Tutorial I-III, Gut bewacht: Mandatory Access Control für Linux-Systeme, iX 8/2006-10/2006

[3] Dan Walsh, A step-by-step guide to building a new Policy Module: [http://www.redhatmagazine.com/2007/08/21/a-step-by-step-guide-to-building-a-new-selinux-policy-module]

[4] Russell Coker, SELinux Play-Machines: [http://www.coker.com.au/selinux/play.html]

[5] Dan Walsh, Creating a Kiosk Account: [http://danwalsh.livejournal.com/13376.html]