

Regel-recht

Eine ausgefeilte Zugriffskontrolle ist die Basis eines sicheren Linux. Die National Security Agency (NSA) entwickelt mit SE Linux ein komplexes System unter der GPL, in dem der Admin die Rechte exakt regelt. Dieser Beitrag erläutert Hintergründe, Grundlagen, Installation und Praxis. Konstantin Agouros, Carsten Grohmann, Achim Leitner



Hannes Keller / www.visipix.com

Sie ist eine Quelle zahlreicher Mythen: Gerüchte kursieren über Mathematiker und Computer, die jeden Code knacken. IT-Sicherheit eine Legende? Die National Security Agency (NSA) sieht das anders und kümmert sich um mehr Sicherheit. Ein Ergebnis ist Security Enhanced Linux (SE Linux), das als Forschungsprototyp entstand [1]. SE Linux ergänzt Linux um zusätzliche Zugriffskontrollen. Es entscheidet anhand einer Regelbasis (Policy), wer auf welche Systemteile welchen Zugang hat, also welcher Prozess unter welcher Benutzererkennung welche Dateien öffnen oder welche Netzverbindung aufbauen darf.

Die Policy lässt sich von normalen Systembenutzern nicht beeinflussen, sie wird vom Admin vorgegeben – damit setzt SE Linux MAC um (Mandatory Access Control, siehe **Kasten „Wichtige**

Begriffe“). Dass die Sicherheitseinstellungen sehr detailliert möglich sind, hat seinen Preis in der beachtlichen Komplexität. Um ein Programm auf SE Linux geschützt laufen zu lassen, muss der Admin jede Datei kennen, die der Prozess öffnet, und auch jedes Unterprogramm, das er aufruft. Der Aufwand lohnt sich aber durch den Schutz, den der Admin damit erreicht.

Tragfähige Konzepte, nicht nur für Linux

Die Sicherheitskonzepte von SE Linux wurden ursprünglich nicht für Linux entworfen. Die NSA entwickelte zusammen mit der Firma Secure Computing [13] (bekannt durch das TIS Firewall Toolkit) zunächst zwei Architektur-Prototypen für den Mach-Kernel [14]: DT

Mach (Distributed Trusted Mach) und DTOS (Distributed Trusted Operating System). Erst die Weiterentwicklung Flask (Flux Advanced Security Kernel) wurde dann auf Linux portiert.

Aufgabe des Flask-Systems ist es, die Integrität und Vertraulichkeit der Informationen sicherzustellen. Mit anderen Worten: Es geht um Zugriffskontrolle. Während ein Standard-Linux-Kernel oft die einfache Antwort gibt „Root darf immer, der Rest nie“, bietet Flask feiner abgestufte Sicherheitsvorgaben. Diese Vorgaben betreffen die Zugriffsrechte auf Dateien, aber auch die Kommunikation zwischen Prozessen und vieles mehr. Wie andere Pakete auch kann SE Linux nur bedingt protokoll- oder programm-spezifische Schwächen ausgleichen, es dämmt aber ihre Auswirkung ein.

Die zentrale Komponente der Flask-Architektur ist der Security-Server. Er trifft alle sicherheitsrelevanten Entscheidungen. Der Name stammt noch von den ersten Mach-Implementierungen, bei denen war es tatsächlich ein Userspace-Prozess. Unter Linux ist dieser Server ein Kernel-Subsystem.

Die zweite Komponente sind Objektmanager. Sie verwalten die Sicherheitsattribute, sorgen für korrekte Zuordnung zu den Objekten (Dateien, Prozesse, Sockets ...) und setzen die Entscheidung des Security-Servers um. Bei den Objektmanagern handelt es sich um die bekannten Kernel-Subsysteme wie Prozessmanager, Filesysteme oder Sockets, die entsprechend erweitert wurden.

Als Basis für die Entscheidungen des Security-Servers dienen so genannte Security Contexts, sie fassen verschiedene Sicherheitsattribute zusammen. Ein Context bestehen aus der Benutzeridentität,

seiner Rolle, einem Typ und optional noch einem MLS-Level (Multi Level Security). Dabei sind nur legale, dem Security-Server bekannte Kombinationen zugelassen.

Praktische Abstraktion

Die einzelnen Komponenten eines Security Contexts stammen von den Abstraktionsebenen, die SE Linux einführt. Durch diese Ebenen wird es einfacher, die komplexe Realität der möglichen Zugriffe zu beherrschen. Die Zugriffskontrolle muss für jedes Programm regeln, unter welchen Voraussetzungen es Zugang zu welchen Objekten erhält.

Die erste Abstraktion sind Benutzer. Die Benutzerverwaltung von SE Linux ist aber unabhängig von der User-ID in Linux. Das hat mehrere Vorteile, beispielsweise lässt sich die SE-Linux-Benutzeridentität nach einem Login nicht mehr ändern. Der Wechsel von Berechtigungen gelingt über das Ändern der Rolle (ein weiteres Sicherheitsattribut) oder des Typs (das dritte Attribut).

Mehrere Linux-Benutzer lassen sich zu einem unprivilegierten SE-Linux-Benutzer zusammenzufassen, die Menge und

Komplexität der Regeln bleibt dadurch überschaubar. Ein Beispiel ist der generische SE-Linux-User »user_u«. Nur für Benutzer, die mehr als die Standardrechte des »user_u«-Benutzers benötigen, ist die Policy anzupassen.

Bei SE Linux bezieht sich der Begriff Benutzer in der Regel auf reale Personen, die interaktiv Zugang zum System haben, Ausnahme ist »system_u«. Zusätzliche Pseudo-User für bestimmte Prozesse sind aber nicht mehr erforderlich, die Rechte dieser Prozesse sind durch den jeweiligen Typ festgelegt. Dennoch werden auch weiterhin einige Programme unter ihrer eigenen Benutzererkennung laufen – das ist schon wegen der Rechten des Dateisystems nötig, die SE Linux nicht ersetzt. Durch die getrennte Benutzerverwaltung ist die Rechteverwaltung beider Komponenten unabhängig voneinander, ein Zugriff muss in beiden erlaubt sein um zu gelingen.

Die nächste Abstraktionsebene sind Rollen (RBAC, Role-Based Access Control). Sie sind frei definierbar. Es ist möglich, mehrere Prozesse und Programme unter einer gemeinsamen Rolle laufen zu lassen. Die Rollen orientieren sich an den Aufgaben, für die ein Prozess oder eine

Datei zuständig ist. Die im Folgenden verwendete Beispielkonfiguration nutzt drei Rollen: »system_r«, »sysadm_r« und »user_r«. Alle Systemprozesse laufen in der Rolle »system_r«, die normalen Benutzer sind »user_r« zugeordnet und Administratoren sind Mitglieder der »sysadm_r«-Rolle.

Starke Typen

Als weitere Abstraktion dienen Typen (TE, Type Enforcement). Es sind letztlich die Typen, über die ein Zugriff erlaubt oder verboten wird. Die Regeln sagen aus, welche Typen auf welche anderen Typen zugreifen dürfen. An manchen Stellen findet man noch eine Unterscheidung zwischen Domänen und Typen, diese Trennung ist aber nur sprachlicher Natur. Mit Domänen bezeichnet man Typen, die an Prozesse gebunden sind. Intern gibt es keinen Unterschied.

SE Linux bestimmt den Typ nicht direkt aus der Benutzererkennung oder dem Dateinamen, sondern über die Rollen. Aus den mit der Rolle verknüpften Typen leiten sich deren Rechte ab. Während kein Benutzer ein Kernelmodul in der »user_r«-Rolle laden darf, selbst wenn er Root

Wichtige Begriffe

DAC: Bei Discretionary Access Control (übliches Linux-Verfahren) können die Benutzer nach eigenem Ermessen (Discretion) die Zugriffsrechte auf ihre Objekte ändern. Meist bezeichnet DAC die User-ID-basierte Zugriffskontrolle. Ob eine Aktion erlaubt ist, entscheidet sich dort anhand der User-ID des Subjekts und des Objekt-Eigentümers. Es gibt nur zwei Arten von Usern: normale User und den Superuser.

Domäne: Sicherheitsattribut eines Prozesses innerhalb des TE-Modells (Type Enforcement). SE Linux unterscheidet bei TE nicht zwischen Typ und Domäne. Ein Typ, der sich auf ein Subjekt (einen Prozess) bezieht, wird aber häufig als Domäne bezeichnet.

Label: Symbolische Kennzeichnung für Subjekte und Objekte, anhand derer SE Linux die Erlaubt- oder Verboten-Entscheidung für die Zugriffe fällt. In einem Label stehen die Sicherheitsattribute, vergeben werden sie von der zentralen Policy. Bei SE Linux steht im Label der Security Context.

MAC: Bei Mandatory Access Control gibt ein Policy-Administrator zentral die Zugriffsrechte vor. User und ihre Prozesse können diese Policy nicht ändern, sie gilt für alle Zugriffe. Viele Definitionen gehen bei MAC von der Spezial-

form MLS aus und bezeichnen das allgemeine MAC als Non-Discretionary Access Control.

MLS: Bei Multi Level Security erhalten Subjekte und Objekte eine Sicherheitsstufe, vergleichbar mit einer Geheimhaltungseinstufung: vertraulich, geheim, top secret. Nur wer eine ausreichende Sicherheitsstufe (Freigabe, Clearance) hat, darf auf ein Objekt zugreifen.

Objekt: Alle Komponenten, auf die zugegriffen wird, etwa Dateien, Verzeichnisse oder Netzwerk-Sockets.

Permissions: Abhängig von der Art des Objekts. Bei Dateien etwa Lesen, Schreiben, Anlegen, Umbenennen oder Ausführen, bei Prozessen zum Beispiel Fork, Ptrace oder Signale senden.

PSID: Die Persistent SID ist die permanente Ausführung einer Security-ID. Eine PSID gilt dauerhaft, auch über Reboots hinweg. Sie stellt die Verbindung von Objekten und ihrem Security Context her, etwa bei Dateien.

Policy: Dieses Regelwerk legt fest, welche Zugriffe für wen erlaubt sind.

RBAC: Role Based Access Control beschreibt die Regelung der Zugriffsrechte über so genannte Rollen. Aus den mit einer Rolle verknüpften Typen und Domänen werden bei SE Linux die Rechte abgeleitet.

Rolle: Rollen vereinfachen die Benutzerverwaltung. Je nach Aufgabe treten die User in bestimmten Rollen auf. Die Rechte werden über die jeweilige Rolle zugeteilt, die Zuordnung von Benutzern zu Rollen ist davon unabhängig.

Security Context: Fasst User-Kennung, Rolle und Typ zusammen. Um kompatibel mit beliebigen Sicherheitsmodellen zu sein, ist der Security Context ein reiner Text-String, sein Inhalt wird nur vom Security-Server ausgewertet.

SID: Die Security-ID ist eine Zahl, die einen konkreten Security Context bezeichnet. Diese Zuordnung nimmt der Security-Server zur Laufzeit von SE Linux vor.

Subjekt: Handelnde Komponente im System, also ein Prozess.

TE: Type Enforcement regelt den Zugriff von Domänen (Klassen von Subjekten) auf Typen (Klassen von Objekten) oder andere Domänen anhand einer Zugriffsmatrix. SE Linux vereinfacht dieses Modell und bezeichnet auch die Domänen als Typ. Die Matrix regelt die erlaubten Interaktionen zwischen Typen.

Typ: Sicherheitsattribut eines Objekts innerhalb des TE-Modells (Type Enforcement).

User: Die Benutzerverwaltung von SE Linux ist unabhängig von der Linux-Benutzererkennung.

ist, gelingt ihm das in der »sysadm_r«-Rolle – jedenfalls, wenn er in diese Rolle schlüpfen darf.

Der Rollenwechsel mit »newrole« startet nach einer erfolgreichen Authentifizierung einen Prozess in der neuen Rolle. Damit dies möglich ist, muss der Wechsel zwischen beiden Rollen prinzipiell zugelassen sein und der Benutzer muss auch beide Rollen benutzen dürfen.

Der Domänenwechsel als spezieller Typwechsel ist wie beim Rollenwechsel nur über einen neuen Prozess möglich, bei dem von einer Domäne in die nächste gewechselt wird. Um die Übersicht bei Benutzer, Typen, Rollen und Domänen zu behalten, werden üblicherweise folgende Endungen verwendet:

- Benutzer: »_u«
- Rolle: »_r«
- Typ oder Domäne: »_t«

Für Benutzer des Linux-Systems wird die Endung »_u« nicht verwendet, damit sind auch diese beiden Konzepte einfach zu unterscheiden.

Entscheidungen gekonnt durchsetzen

Um mit den Sicherheitsattributen arbeiten zu können, werden sie zu einem Security Context zusammengefasst. Der Security Context eines Subjekts (Prozess) oder eines Objekts (Datei, Socket, IPC-Objekt ...) ist dreiteilig aufgebaut, als Trennzeichen im Textstring dient der Doppelpunkt »:«. Die Attribute sind der Benutzer, seine Rolle und der Typ, etwa »system_u:object_r:inetd_exec_t«.

Der Security-Server weist jedem Prozess einen Security Context zu. Der ergibt sich aus dem Regelwerk, dem Vaterprozess und der Benutzer-ID, unter der der Prozess läuft. Das Regelwerk muss festlegen, welcher Prozess welche Kindprozesse starten darf. So wird ein kompromittierter Sendmail-Prozess davon abgehalten, »/bin/tcsh« zu starten.

Über das Regelwerk erhält auch jedes Objekt ein Label, in dem sein Security Context vermerkt ist. Die Zugriffsrechte (Permissions) sind dabei wesentlich feiner abgestuft, als die von Linux normalerweise bereitgestellten Rechte. Bei Dateien unterscheidet SE Linux unter anderem das Lesen, Schreiben, Anlegen, Umbenennen und Ausführen, bei Prozessen können die Permissions zum Beispiel einen Fork, Ptrace oder das Senden von Signalen erlauben.

Während seiner Laufzeit benutzt SE Linux nicht immer die ausführliche String-Darstellung, sondern gibt den Strings Nummern (SID, Security Identifier). Diese Integer-Nummern sind nur lokal gültig und nicht von Dauer, für Dateisystemobjekte kommen daher persistente SIDs (PSID) zum Einsatz. Deren Zuordnung zum Security Context wird im jeweiligen Dateisystem gespeichert. Ein interessanter Plan [12] der SE-Linux-Entwickler ist es, die SIDs an IPSEC-Security-Assoziationen zu koppeln. Damit würden vernetzte Applikationen auf verschiedenen SE-Linux-Rechnern im gleichen Security-Level laufen.

Vor jedem Zugriff sendet der Objektmanager die Security Contexts von Subjekt

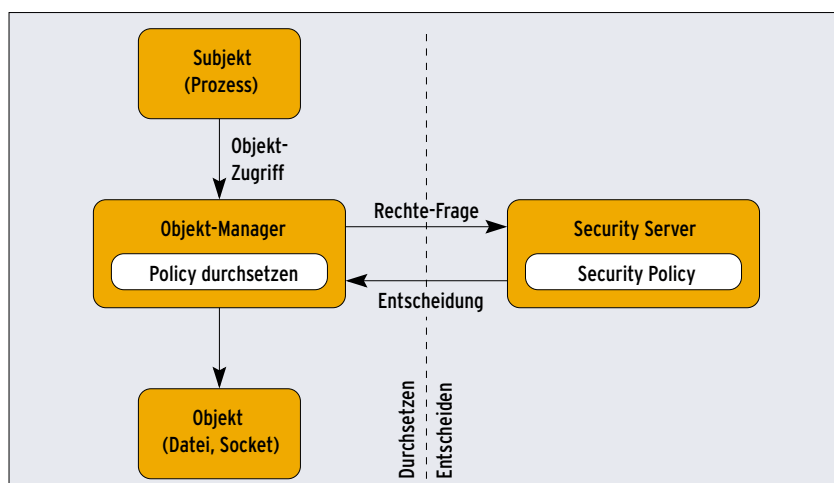


Abbildung 1: Der Zugriff eines Subjekts auf ein Objekt wird vom jeweiligen Objektmanager geregelt. Er konsultiert den Security-Server und fragt, ob der Zugriff erlaubt ist.

und Objekt an den Security-Server, der anhand seiner Regelbasis eine Entscheidung fällt. Wenn ein Prozess auf eine Datei zugreifen will, fragt der zuständige Objektmanager bereits beim »open()«-Call den Security-Server, ob der Zugriff erlaubt ist. Im Gegensatz zum normalen Linux fragt der Manager aber bei jedem Schreib- und Lesezugriff erneut, ob diese Aktion noch erlaubt ist. Ein Standardkernel trifft die Entscheidung dagegen nur einmal zu Beginn. Wenn sich die Rechte der geöffneten Datei ändern, kann der Prozess weiterlesen – SE Linux verhindert dies.

Damit die Performance des Systems nicht zu sehr unter den häufigen Nachfragen leidet, gibt es den Access Vector Cache (AVC). Er speichert die Antworten des Security-Servers und liefert sie bei wiederholten Anfragen deutlich schneller, die Gesamtleistung leidet damit kaum unter den häufigen Überprüfungen. Sollten sich die Rechte in der SE-Linux-Policy ändern, markiert der Security-Server die geänderten Einträge im AVC als ungültig. **Abbildung 2** verdeutlicht den Zugriff.

Jeder Prozess läuft in seinem eigenen geschützten Kontext. Die User-ID »0« hat dabei keine besondere Auswirkung, wenn das Regelwerk nicht ausdrücklich etwas anderes festlegt. Prozesse lassen sich damit einsperren, Systemaufrufe an- und abstellen und es lässt sich genau festlegen, welche Dateien geschrieben, gelesen oder erzeugt werden dürfen. Einem Prozess, der besondere Rechte benötigt (etwa das Binden auf Ports unter 1024), können alle anderen Rechte genommen werden, mit denen er Schaden am System anrichten könnte. Selbst

SuSE-Pakete für SE Linux

Um SE Linux kompilieren zu können, ist eine Reihe von Tools nötig. Bei SuSE Linux 7.3 sind diese Werkzeuge in folgenden RPM-Paketen zu finden:

- Serie »d«: bison, gettext, flex, pam_devel, openssl-devel, patch, slang (für slcurses.h) und yacc
 - Serie »a«: diffutils, ncurses (für libncurses), texinfo (für makeinfo), base (für m4) und util-linux (für more)
 - Serie »ap«: sharutils
- Je nach eigener Konfiguration können auch noch weitere Pakete hinzukommen.

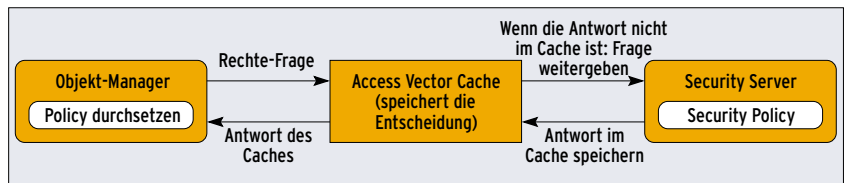


Abbildung 2: Die Entscheidung des Security-Servers, ob ein Zugriff erlaubt ist oder nicht, wird im Access Vector Cache gespeichert. Das beschleunigt weitere gleich lautende Fragen des Objektmanagers.

wenn ein Angreifer zu Root-Rechten kommt, bleibt er im Käfig gefangen. Die Quellen von SE Linux unterliegen der GPL. Sie stehen auf den Webseiten der NSA [1] und auf Sourceforge [4] zum Download bereit. Im Test kam das Komplettpaket [2] zum Einsatz, installiert auf einem Minimalsystem mit SuSE 7.3. Da SE Linux auf Red Hat basiert, sind einige kleinere, aber überschaubare Änderungen an der Installation nötig. Eine Einschränkung ist zu beachten: SE Linux unterstützt zurzeit nur die Dateisysteme Ext 2, Ext 3 und ReiserFS.

SE Linux installieren

Für die Installation sind Root-Rechte notwendig. Beim Auspacken des Archivs entstehen die beiden Unterverzeichnisse »lsm-2.4« mit dem angepassten Kernel sowie »selinux« mit den benötigten Programmen und Regeln. Der Kernel enthält die LSM-Patches (Linux Security Modules) [5], die dem Kernel die passenden Haken und Ösen einfügen, um die Zugriffskontrollen von SE Linux als Kernelmodul implementieren zu können. Der SuSE-Kernel ist für SE Linux nicht geeignet, wegen der umfangreichen Änderungen lassen sich die LSM- und SE-Linux-Patches darauf nicht anwenden. Das neue Unterverzeichnis »selinux« dient, sofern nicht anderes angegeben,

als Basis für alle weiteren Schritte. Ganz Eilige könnten die komplette Installation einem Makefile überlassen: »make quickinstall«. Der ausführliche Weg zeigt aber interessante Zusammenhänge. Dafür sind einige Tools erforderlich, die in dem Kasten »SuSE-Pakete für SE Linux« genannt sind.

Für SE Linux sind noch einige Änderungen am LSM-Kernel nötig. Die Patches hierfür liegen im Unterverzeichnis »selinux/module«, dort genügt ein »make insert«. Damit der Kernel automatisch nach »/boot« installiert wird, muss im Kernel-Makefile das Kommentarzeichen vor »export INSTALL_PATH= /boot« entfernt werden – ein Patchfile kann diesen Schritt übernehmen.

Ein weiteres Patch ändert den Dateinamen des Kernelabbilds auf »/boot/vmlinuz-selinux«. Damit bleibt der ursprüngliche Kernel erhalten, was während der Tests nützlich ist und auch später bei Problemfällen hilft. Die beiden zuletzt genannten Patches stehen unter [7] zur Verfügung:

```

cd ../lsm-2.4
patch -p0 < ../kernel_install_path.diff
patch -p0 < ../kernel_vmlinuz-selinux.diff
  
```

Der neue Kernel ist noch für den eigenen Rechner passend zu konfigurieren, eine vorhandene »config« kann als Grundlage dienen. SE Linux benötigt das »Net-



Abbildung 3: Beim Konfigurieren mit »make xconfig« müssen innerhalb der Security-Optionen einige Module noch in den Kernel eingebunden werden: »Capabilities Support« und »NSA SELinux Support« sind Pflicht, der Development-Support ist praktisch.

work Packet Filtering« aus den »Networking Options« sowie einige Einstellungen in den »Security Options«, siehe [Abbildung 3](#). Das Modul »NSA SELinux Development Support« erleichtert das Entwickeln eigener Regelsätze. Es startet SE Linux im Permissive Mode und nicht im Enforcing Mode: Regelverstöße führen dabei nur zu einem Protokolleintrag, werden aber nicht verhindert. Anschließend wird der Kernel erstellt:

```
make dep && make bzImage && make modules 2
&& make modules_install && make bzlilo 2
&& make clean
```

Der Bootloader muss nun noch den neuen Kernel und dessen Möglichkeiten kennen. Mit dem einkompilierten Development-Support startet SE Linux automatisch im Permissive Mode, die Boot-Option »enforcing= 1« ändert dies. Es hat sich bewährt, zu den vorhandenen Bootkonfigurationen noch zwei weitere hinzuzufügen: Der erste Eintrag bootet SE Linux mit »enforcing= 1«, der zweite bootet ohne zusätzliche Parameter, sodass SE Linux im liberalen Modus startet. Der strikte Modus sollte der Defaultfall sein, damit ein Neustart die Regeln nicht außer Kraft setzt.

Grafischen Login abschalten oder ersetzen

Vor der Installation ist zu prüfen, ob der Rechner im Runlevel 3 (ohne grafischen Login) hochfährt. Die Displaymanager sind noch nicht an das neue Login-Verhalten angepasst, funktionieren also nicht. Statt »/etc/inittab« zu ändern, genügt es, die SE-Linux-Einträge in »/etc/lilo.conf« um »append= 3« zu ergänzen. Ein abschließender Aufruf von »lilo« aktiviert die neue Konfiguration.

Wer unbedingt einen grafischen Login benötigt, kann einen modifizierten GDM für Red Hat von [\[4\]](#) oder ein Patch für KDM von [\[9\]](#) verwenden. Ohne Displaymanager lässt sich X11 auch manuell per »startx« aufrufen, damit laufen auch KDE und Gnome unter SE Linux. Leider sind noch keine Regelsätze für die Desktop-Umgebungen verfügbar.

Das SE-Linux-Paket enthält eine Reihe modifizierter Userspace-Programme. Die meisten werden unter »/usr/local/selinux« installiert, Ausnahmen sind etwa

OpenSSH, Login und der Cron-Daemon. Die übliche Make-Kombination »make && make install« ist in »selinux/module« und »selinux/libsecure« auszuführen. Die weiteren Werkzeuge benötigen noch zwei Patches, die in [\[7\]](#) enthalten sind:

```
patch -p0<utils_makefile_uselargefile.diff
patch -p0<utils_libncurses.diff
```

Ohne diese Patches lässt sich der Quellcode unter SuSE Linux nicht übersetzen. In »selinux/utils/Makefile« sind für die meisten Tools »./configure«-Aufrufe enthalten, die Optionen lassen sich an dieser Stelle anpassen. »make && make install« im »utils«-Unterverzeichnis erstellt und installiert diese Pakete, das gilt auch für das Setfiles-Tool im Verzeichnis »selinux/setfiles«.

Weiche Landung ohne harte Links

Vor der Konfiguration von SE Linux ist noch eine Hürde aus dem Weg zu schaffen: Die Datei »/etc/localtime« ist ein Hardlink (siehe auch [Kasten „Allgemeine Tipps“](#)). SE Linux hat damit Probleme, weil zwei unterschiedliche Sicherheitseinträge auf dasselbe Inode verweisen würden.

```
cd /etc
cp localtime localtime.hl
rm localtime
mv localtime.hl localtime
```

Damit »SuSEconfig« diesen Hardlink nicht wieder anlegt, sollte der Eintrag »TIMEZONE= "Zeitzone"« in »/etc/rc.config« auf den Wert »YAST_ASK« gesetzt werden. Ab SuSE 8.0 ist dieser Eintrag in »/etc/sysconfig/clock« zu finden. Sollten noch andere Hardlinks im System existieren, warnt SE Linux beim Anlegen der Security Contexts.

Fast alle Dateien, die das Verhalten von SE Linux steuern, liegen unter »selinux/

policy«, damit ist dieses Verzeichnis die Basis für die nächsten Schritte. Kommentarzeilen beginnen in den Konfigurationsdateien wie üblich mit einer Raute »#«. Der Inhalt dieser Files wird noch vom Makro-Präprozessor »m4« interpretiert, viele Admins kennen das Tool von der Sendmail-Konfiguration. Die Makros vereinfachen komplexere Konfigurationen – für die ersten Schritte und die normale Anwendung muss man die »m4«-Sprache aber nicht verstehen.

SE Linux konfigurieren

Die Datei »users« weist jedem Benutzer die nutzbaren Rollen zu. Nur Benutzerkennungen, die hier aufgelistet sind, lassen sich auch in der System-Policy nutzen. Als Erstes sollte man die Beispielbenutzer »jdoe« und »jadmin« löschen. Die Einträge haben das Format »user *Benutzername* roles *Rolle*;« oder, wenn mehrere Rollen erlaubt sind, »user *Benutzername* roles { *Rolle1* *Rolle2* };«.

Drei weitere Benutzer sind schon vorgegeben. »system_u« ist der Systembenutzer, daneben gibt es »root« und »user_u«. Nicht aufgeführte Benutzer werden automatisch dem Standardbenutzer »user_u« und dessen Rolle »user_r« zugeordnet, dadurch ist es nicht notwendig, jeden einzelnen Linux-User zu dieser Datei hinzuzufügen. Ein Beispieleintrag ohne besondere Privilegien könnte so aussehen:

```
user foo roles { user_r bar_r };
```

Soll ein User mehr Rechte erhalten, kann man ihm Zugang zur »sysadm_r«-Rolle gewähren. Sind mehrere Rollen eingetragen

Listing 1: Datei-Kontexte

```
/home/[^/] * -d system_u:object_r:user_home_dir_t
/var/run(/.*) system_u:object_r:var_run_t
/var/run/./*.*pid <<none>
```

Listing 2: Minigetty-Fehlermeldungen

```
1 Jul 5 17:36:36 max kernel: avc: denied { read } for pid=616 exe=/sbin/minigetty path=/2/fd/10 dev=00:03
ino=163850 scontext=system_u:system_r:getty_t tcontext=system_u:system_r:init_t tclass=lnk_file
2 Jul 5 17:36:36 max kernel:
3 Jul 5 17:36:36 max kernel: avc: denied { read } for pid=616 exe=/sbin/minigetty path=/450/maps dev=00:03
ino=29491213 scontext=system_u:system_r:getty_t tcontext=system_u:system_r:postfix_master_t tclass=file
4 Jul 5 17:36:36 max kernel:
5 Jul 5 17:36:36 max kernel: avc: denied { getattr } for pid=616 exe=/sbin/minigetty path=/450/maps dev=00:03
ino=29491213 scontext=system_u:system_r:getty_t tcontext=system_u:system_r:postfix_master_t tclass=file
```

gen, kann der Benutzer jederzeit wechseln. Die »sysadm_r«-Rolle gibt dem User dieselben SE-Linux-Rechte wie Root, dennoch hat er keine Root-Rechte auf dem zugrunde liegenden Linux-System, weil SE Linux zusätzlich zu den Standard-Linux-Rechten greift. Daher kann dieser Benutzer auch nicht auf das Verzeichnis »/root« zugreifen.

In der Praxis werden mit den Rollen Zugriffe auf verschiedene Programme gewährt oder abgelehnt. Ein Beispiel dafür ist das Programm »insmod«. Die Mitglieder der »sysadm_r«-Rolle sowie das System selbst dürfen das Programm benutzen, da beide Rollen die Domäne (den Typ) »insmod_t« enthalten. Den normalen Benutzern in der »user_r«-Rolle fehlt dagegen die notwendige Berechtigung. Daher können diese Benutzer und ihre

Programme keine Kernelmodule nachladen. Auch wenn ein Benutzer Root-Rechte (User-ID »0«) erlangen sollte, ist ihm das Nachladen von Modulen unmöglich, wenn er sich nicht in der »sysadm_r«-Rolle befindet.

Security Context für Dateien und Prozesse

Die Dateien unter »file_contexts« ordnen den Dateisystemeinträgen ihren Security Context zu. »types.fc« enthält die allgemeinen programmunabhängigen Zuordnungen, die programmspezifischen stehen unter »program«. Ein Blick in diese Dateien hilft die Funktionsweise von SE Linux besser zu verstehen (Listing 1): Jede Zeile beginnt mit dem Dateisystemeintrag, der als Regular Expression recht bequem definiert werden kann. Das Verankern mit »^« am Anfang und »\$« am Ende kann entfallen, SE Linux fügt diese Zeichen automatisch ein.

Auf die Dateiangebe folgt eventuell die Dateiart als Option mit führendem Minuszeichen. Ein »-« steht für einen Verzeichniseintrag, die Regel gilt dann nur für Verzeichnisse. Sind nur gewöhnliche Files gemeint, ist »-« einzutragen. Am Ende der Zeile ist der Security Context

aufgeführt. Er besteht stets aus dem User »system_u«, der Rolle »object_r« und einem passenden Typ. Die während der Laufzeit von SE Linux neu erzeugten Files erhalten den Benutzer, die Rolle und den Typ entsprechend dem erzeugenden Prozess. Um keinen Security Context anzulegen, lässt sich hier auch »< < none« angeben.

Passt eine Datei zu mehreren Regeln, verwendet SE Linux die letzte zutreffende Zeile. Die zweite Zeile in Listing 1 passt auf alle Dateisystemeinträge unterhalb »/var/run« und kennzeichnet sie mit dem Security Context »system_u:object_r:var_run_t«. Die nächste Zeile löscht die Kennzeichnung für alle Einträge, die auf ».pid« enden.

Hier zeigt sich, wie wichtig die Reihenfolge ist: Wären Zeile 2 und 3 vertauscht, würden letztlich auch die PID-Dateien den Security Context erhalten. Die Einträge sollten also vom Allgemeinen zum Speziellen geordnet sein. Eine falsche Reihenfolge ist oft Ursache für unerwartete Ergebnisse.

Die sprachliche Trennung beim Type Enforcement zwischen Typ für Dateien und Domäne für Prozesse findet sich in den Namen der Konfigurationsdateien wieder: Im Verzeichnis »selinux/policy/do-

Listing 3: Pfade ergänzen

```
#
# add SE Linux utilities and man pages to path and manpath
#
uname -r | grep --silent selinux
if [ "$?" = "0" ]; then
    PATH=/usr/local/selinux/bin:/usr/local/selinux/sbin:$PATH
    MANPATH=/usr/local/selinux/man:$MANPATH
    export PATH MANPATH
fi
```

Listing 4: Prozesse

PID	SID	CONTEXT	COMMAND	279	172	system_u:system_r:atd_t	/usr/sbin/atd
1	7	system_u:system_r:init_t	init [489	176	system_u:system_r:inetd_t	/usr/sbin/xinetd -reuse
2	7	system_u:system_r:init_t	[keventd]	550	180	system_u:system_r:crond_t	/usr/sbin/crond
3	7	system_u:system_r:init_t	[kapmd]	574	182	system_u:system_r:getty_t	/sbin/mingetty --noclear t
4	1	system_u:system_r:kernel_t	[ksoftirqd_CPU0]	575	186	system_u:system_r:local_login_t	login -- root
5	1	system_u:system_r:kernel_t	[kswapd]	576	182	system_u:system_r:getty_t	/sbin/mingetty tty3
6	1	system_u:system_r:kernel_t	[bdflush]	577	182	system_u:system_r:getty_t	/sbin/mingetty tty4
7	1	system_u:system_r:kernel_t	[kupdated]	578	182	system_u:system_r:getty_t	/sbin/mingetty tty5
8	7	system_u:system_r:init_t	[kreiserfsd]	579	182	system_u:system_r:getty_t	/sbin/mingetty tty6
214	169	system_u:system_r:syslogd_t	/sbin/syslogd	603	187	root:sysadm_r:sysadm_t	-bash
217	166	system_u:system_r:klogd_t	/sbin/klogd -c 1	622	187	root:sysadm_r:sysadm_t	ps ax --context

Listing 5: Dateisystem

drwxr-xr-x	root	root	system_u:object_r:root_t	./	drwxr-xr-x	root	root	system_u:object_r:lib_t	lib/
drwxr-xr-x	root	root	system_u:object_r:root_t	../	drwxr-xr-x	root	root	system_u:object_r:lost_found_t	lost+found/
drwx-----	root	root	system_u:object_r:file_labels_t	...security/	drwxr-xr-x	root	root	system_u:object_r:root_t	media/
drwxr-xr-x	root	root	system_u:object_r:bin_t	bin/	drwxr-xr-x	root	root	system_u:object_r:root_t	mnt/
drwxr-xr-x	root	root	system_u:object_r:boot_t	boot/	drwxr-xr-x	root	root	system_u:object_r:root_t	opt/
lrwxrwxrwx	root	root	system_u:object_r:root_t	cdrom	dr-xr-xr-x	root	root	system_u:object_r:proc_t	proc/
lrwxrwxrwx	root	root	system_u:object_r:root_t	cdrw	drwx-----	root	root	system_u:object_r:sysadm_home_t	root/
drwxr-xr-x	root	root	system_u:object_r:device_t	dev/	drwxr-xr-x	root	root	system_u:object_r:sbin_t	sbin/
drwxr-xr-x	root	root	system_u:object_r:etc_t	etc/	drwxrwxrwt	root	root	system_u:object_r:tmp_t	tmp/
lrwxrwxrwx	root	root	system_u:object_r:root_t	floppy	drwxr-xr-x	root	root	system_u:object_r:usr_t	usr/
drwxr-xr-x	root	root	system_u:object_r:user_home_t	home/	drwxr-xr-x	root	root	system_u:object_r:var_t	var/

mains/program« sind Files mit Namen »*.te« (Type Enforcement) enthalten. Sie legen fest, welche Zugriffe erlaubt sind (welche Domäne darf auf welchen Typ wie zugreifen). In diesen »*.te«-Dateien sind nur die Typ-Angaben ausschlaggebend, nicht etwa Dateinamen, Benutzerkennungen oder Rollen.

Damit die Zuweisung der Rechte auch wie geplant funktioniert, müssen die Prozesse einen Security Context erhalten, das erfolgt über Dateien mit der Endung »*.fc« (File Context) im Unterverzeichnis »selinux/policy/file_contexts/program«. Der Security Context des Prozesses ergibt sich unter anderem aus dem Security Context der Programmdatei – und den legen die »*.fc«-Files fest.

Angepasste Einstellung

Nur wenn alle Einträge korrekt an die jeweilige Distribution angepasst wurden, kann SE Linux fehlerfrei funktionieren. Da die Vorgaben Red-Hat-spezifisch sind, sind sie für SuSE-Systeme entsprechend zu ändern. Das betrifft vor allem die Dateisystemeinträge in der linken Spalte der »*.fc«-Files, da beide Distributionen unterschiedliche Pfade für einzelne Programme benutzen.

Wer sich die zeitraubende Arbeit der manuellen Anpassung ersparen möchte, findet die zusätzlichen Regeln für SuSE unter [8]. Um spätere Regeländerungen von SE Linux einfacher einspielen zu können, wurden alle spezifischen Regeln in »suse.fc« und »suse.te« zusammengefasst. »suse.fc« sollte in das Verzeichnis »selinux/policy/file_contexts/program« kopiert werden, für »suse.te« ist »selinux/policy/domains/program« der richtige Ort.

Unter »selinux/utils/appconfig« befinden sich einige Dateien, die die für SE Linux modifizierten Programme konfigurieren. Alle Files werden zuerst nach »/etc/security« kopiert. Die Datei »default_contexts« legt für den lokalen Login, für Login via SSH sowie für Cronjobs fest, welche Rolle und welcher Typ per Default zum Zuge kommt. Die Einträge in »default_type« ordnen jeder Rolle einen Typ als Standard zu. Die Einträge haben die Form »Rolle:Domäne«. Änderungen in dieser Datei sind nur nötig, wenn neue Rollen definiert werden.

Die restlichen Dateien benötigen keine Änderungen. »initrc_context« enthält in der einzigen Zeile den Security Context, in dem die Init-Skripte ablaufen, wenn sie nachträglich mit »run_init« aufgerufen werden. »passwd_context« und »shadow_context« enthalten den Security Context für »/etc/passwd« und »/etc/shadow«. Damit können diverse Wrapper-Programme wie etwa »spasswd« den Kontext erneut herstellen, nachdem »passwd« und andere in diese Dateien geschrieben haben.

Die Datei »policy/rbac« konfiguriert den RBAC-Mechanismus (Role Based Access Control) und sollte nicht geändert werden. Es sei denn, eine neue Rolle erfordert diesen Schritt. Die vorhandenen Regeln benötigen keinen zusätzlichen Rollenwechsel, ein unbedachter Rollenwechsel könnte die Sicherheit des ganzen Systems gefährden. Die Datei ist zeilenbasiert und hat folgende Syntax:

```
allow alte_Rolle neue_Rolle;
```

Im Gegensatz zum Typwechsel, für den eigene Regelsätze zuständig sind, erlaubt jede Zeile dieser Datei einen klar definierten Rollenwechsel.

Mingetty mit Maxi-Rechten

Wer SuSEs Mingetty benutzt, wird auf viele Meldungen über fehlende Rechte stoßen (Listing 2). Diese Mingetty-Variante möchte die PID-Verzeichnisse unter »/proc« auslesen. Da diese Verzeichnisse aber mit dem Security Context des zugehörigen Prozesses ausgestattet sind, würde »mingetty« Leserechte für zu viele Typen benötigen und somit den eigentlichen Bestrebungen von SE Linux entgegenwirken.

Eine Lösung wäre, das Mingetty-Paket von Red Hat zu nutzen [10]. Das Binär-RPM entsteht aus dem Source-RPM mit »rpm --rebuild mingetty-1.00-1.src.rpm« und wird mit »rpm -ihv --force mingetty-1.00-1.rpm« installiert. Diese Aktion überschreibt das SuSE-Pendant. Der genaue Pfad zum neu erstellten Mingetty-Paket ist abhängig von der jeweiligen Distribution, üblicherweise lautet er »/usr/src/packages/RPMS/i386«.

Nachdem alle Anpassungen abgeschlossen sind, kann man die Policy im Verzeichnis »selinux/policy« mit »make &&

make install« erstellen. Damit werden die Änderungen nach dem Neustart des Systems wirksam. Zur Laufzeit von SE Linux steht noch »make load« zur Verfügung, das die neue Policy sofort lädt.

Die Policy regelt alles

Im nächsten Schritt verknüpft der Aufruf »make reset« die Security Contexts mit den Dateisystemeinträgen. In den Wurzelverzeichnissen aller eingebundenen und unterstützten Dateisysteme entsteht dabei ein Verzeichnis »...security«, das eine Datenbank mit PSIDs enthält (Persistent Security Identifier). Sie ordnen den einzelnen Dateien und Verzeichnissen über ihren Inode den passenden Security Context zu.

Die Änderungen werden erst nach einem Neustart wirksam. Sollen sie in laufendem SE Linux sofort aktiv werden, hilft »make relabel«. Der Aufruf ist auch nötig, wenn ein Nicht-SE-Linux-Kernel ausgeführt wurde. Disketten- und CD-Laufwerke erhalten nach dem Mounten dynamisch ihren Security Context.

Beim Einpflegen neuer Regeln in das System ist zuerst die Policy zu aktualisieren, bevor man den Security Context der Dateien ändert. Sonst könnte ein neuer Typ dem System noch unbekannt sein, während er schon für Dateien und Verzeichnisse Verwendung findet. SE Linux würde dann Zugriffe verhindern, die vielleicht dringend nötig sind.

Vor dem ersten Booten von SE Linux sollte man noch die Pfade für die Manpages und die Programme erweitern. Am einfachsten geht das mit Hilfe von [Listing 3](#), das als »/etc/profile.local« seinen Dienst verrichtet.

Booten und einloggen

Um bei unvollständigen oder fehlerhaften Regeln handlungsfähig zu bleiben, ist das Booten im Permissive Mode empfehlenswert. Der erste Login sollte als Root erfolgen. Dabei ist es – wie bei jedem anderen Login – möglich, in die »sysadm_r«-Rolle zu wechseln. Wenn SE Linux fehlerfrei gestartet ist, führt »ps ax --context« zu einer ähnlichen Ausgabe wie [Listing 4](#): Es zeigt die Prozesse mit den richtigen Security Contexts. Die dritte Spalte dieser Ausgabe enthält den Security Context mit dem schon bekannten Aufbau »Benutzer:Rolle:Typ«.

Alle Prozesse eines Benutzers laufen unter seiner Identität und mit seiner Rolle. Beide Eigenschaften erben auch die Kindprozesse. Für Systemprozesse ist der Security Context nicht frei wählbar, sie laufen immer als »system_u«-Benutzer und in der »system_r«-Rolle. Nur die Domäne (der Typ) ist vom jeweiligen Prozess abhängig. Sollten alle Prozesse in derselben Domäne laufen, dann wurden vermutlich die Dateien unter »selinux/policy/file_contexts« nur unvollständig angepasst.

Durch die Vererbung der Domäne an Kindprozesse kann es auch passieren, dass sich nach dem Booten noch Prozesse in der »initrc_t«-Domäne befinden. Diese Domäne dient aber nur zum Start der Skripte unter »/etc/init.d/«. Der Admin kann dem auf zwei Wegen entgegen: Entweder er verhindert, dass die RC-Skripte diese Programme starten, oder er definiert eine eigene Domäne für die aufgerufenen Programme.

Die Security Contexts der Dateien zeigt das Kommando »ls / --context« ([Listing 5](#)). Die Ausgabe ähnelt der von »ps«. Wurden beim Shutdown oder Booten Dateien neu angelegt, dann haben diese noch kein Label und damit keinen Security Context, auch wenn eine Policy-Regel auf das File zutrifft. Der Aufruf »make relabel« im Policy-Verzeichnis ergänzt die fehlenden Labels. Abschließend empfiehlt es sich, die Installation im Permissive Mode noch eine Weile zu beobachten, um gegebenenfalls die Regeln weiter anzupassen.

Kommandozeilen-Tools für Admin und User

SE Linux enthält auch einige Userspace-Tools. Das Programm »avc_enforcing« gibt den aktuellen Modus von SE Linux aus, also »enforcing« oder »permissive«. Zwischen beiden Modi wechselt das Tool »avc_toggle«, es benötigt dazu keine Parameter.

Tipps zu SE Linux

Keine Hardlinks: Durch den gemeinsamen Inode kann eine Datei unter verschiedenen Namen keine unterschiedlichen Rechte haben. Der Security Context bezieht sich auf den Inode, daher erhalten beide Namen den gleichen Context, auch wenn die File-Context-Konfiguration versucht verschiedene Contexts zu vergeben.

Ext 2 nicht in Ext 3 umwandeln: Falls ein Ext-3-Dateisystem eingesetzt wird, sollte man es nicht nachträglich aus einem Ext 2 erzeugen. Die dabei angelegte Datei »journal« wirkt in manchen Fällen störend, für sie ist noch kein Dateityp deklariert.

Keine RAM-Disk: SE Linux sollte ohne initiale RAM-Disk gestartet werden.

Mischen von SE Linux und Standard-Linux: Wenn zwischenzeitlich ein Nicht-SE-Linux-Kernel gebootet wurde, sollten vor dem erneuten Start von SE Linux alle Verzeichnisse »...se-

linux« gelöscht und der Security Context für die Dateien neu angelegt werden.

Mailinglisten-Archiv: Das Archiv der SE-Linux-Mailingliste wird auf der Homepage [\[1\]](#) nur unregelmäßig aktualisiert. Es empfiehlt sich daher, andere Archive [\[6\]](#) zu nutzen.

Backup der Konfiguration: Um das Sichern der Konfigurationsdateien zu erleichtern, kann das Verzeichnis »selinux/policy« nach »/etc/selinux/policy« kopiert werden.

Schnelleres Setfiles: Ein Durchlauf von »setfiles« dauert bei Verzeichnissen mit vielen Dateien recht lange. Eine Alternative wäre, »setfiles« manuell aufzurufen und nur die geänderten Dateisysteme zu bearbeiten. Das Arbeitsverzeichnis muss dazu »selinux/policy« sein. Das Kommando lautet »setfiles file_contexts/file_contexts *Wurzel_der_Partition*«. Falls die Datei »file_contexts/file_contexts« nicht existiert oder zu alt ist, hilft der Aufruf

»make filecontext/filecontext«. Kleinere Änderungen sind mit »chcon« generell effektiver als mit dem umfassenden »setfiles«.

X-Server: Wer einen X-Server nutzen möchte, sollte die in der Installationsanleitung »selinux/README« in Punkt 4 beschriebenen Änderungen vornehmen und anschließend »startx« ausführen. Einige Versuche mit KDE 2 meldeten nur wenige Regelverstöße. Ein eigener Regelsatz für KDE könnte auch diese beheben.

Root-Login: Im aktuellen Regelsatz hat »sshd« keinen Zugriff mehr auf das Homeverzeichnis von Root »/root«. Deshalb sollte kein direkter Root-Login via SSH erfolgen. Der nachträgliche Wechsel mit »su -« und »newrole« ist weiterhin möglich und auch empfohlen.

Bootmeldungen: Ein Blick in die Bootmeldungen gibt interessante Einblicke in SE Linux. So erkennt man zum Beispiel, in welchem der beiden Modi SE Linux gestartet ist.

Ähnlich dem »su«-Kommando im Standard-Linux startet »newrole« eine neue Shell in einer anderen Rolle. Der Aufruf »newrole -r sysadm_r« vollzieht nach der Eingabe des Benutzerpassworts einen Wechsel in die »sysadm_r«-Rolle. Die Passworтеingabe stellt sicher, dass nur der Benutzer selbst seine Rolle wechselt, nicht etwa ein Shellskript. Für einen Rollenwechsel müssen folgende Bedingungen erfüllt sein:

- Der Benutzer muss als Mitglied beider Rollen in »users« aufgeführt sein.
- Der Rollenwechsel muss in der Datei »rbac« erlaubt sein.

Ein wichtiges Skript ist »newrules.pl«. Es befindet sich unter »selinux/scripts/« und hilft neue Regeln aus den Kernelmeldungen zu erzeugen. Der Aufruf von »newrules.pl --help« verrät die genaue Syntax. Interessant ist die Option »-v«: Sie ergänzt die Regel um einen Kommentar mit zusätzlichen Informationen über den Prozess, der die Regelverletzung auslöste. Um Init-Skripte nachträglich zu starten, steht »run_init« zur Verfügung. Das Programm wechselt nach der Passwortabfrage in den in »/etc/security/initrc_context« festgelegten Security Context und startet das Init-Skript.

Geänderte Programme für SE Linux

Um die zusätzlichen Features von SE Linux zu nutzen, enthält das Paket einige modifizierte Programme, etwa »ps«, »ls«, »find«, »id« und »mkdir«. Das geänderte »tar« archiviert auch den Security Context der Files. »runas« startet Programme unter anderen Rahmenbedingungen, zum Beispiel mit einer geänderten Rolle oder in einem anderen Context. »chcon« ändert den Security Context von Dateien und Verzeichnissen, allerdings gehen diese Änderungen beim nächsten »make relabel« verloren.

Ist für ein Filesystem noch kein Security Context definiert, kommt »setfiles« zum Einsatz. Eine neue Policy lädt »load_policy« in den Speicher. »list_sids« zeigt die Security Identifier (SID) an, »sid_to_context« gibt für einen SID den zugehörigen Security Context aus.

Bei Linux ist es üblich, dass Root das Passwort von Benutzern ändern kann, ohne deren altes Passwort zu kennen.

Dieses Sonderrecht ist in »passwd«, »chfn« und »chsh« fest verdrahtet. Alle drei Tools benötigen das Schreibrecht auf die Passwort-Files, daher greift eine einfache Policy hier nicht. Die Lösung sind Wrapper-Programme: »spasswd«, »schfn« und »schsh« stellen sicher, dass alle Benutzer nur ihre eigenen Daten ändern dürfen – es sei denn, sie haben ein Sonderrecht. Dieses Sonderrecht hängt aber nicht an der User-ID, sondern an einer eigenen Domäne.

Wichtige Änderungen betreffen auch »login«, »sshd« und »crond«. Weil Cron viele systemspezifischen Aufgaben erfüllt, ist es schwer, die passende Regeln zu definieren. Es empfiehlt sich, überflüssige Aufgaben auszukommentieren und für die dringend benötigten einen Regelsatz für eine eigene Domäne zu schreiben. Unter SuSE heißt der Crond »cron«, während die geänderte Version den Namen »crond« trägt. Das Init-Skript »/etc/init.d/cron« ist entsprechend anzupassen (die Variable »CRON_BIN« legt den Namen des Binary fest).

Auch für Yast existiert noch kein Regelsatz. Dieses Programm benötigt sehr umfangreiche Rechte. Wer den Aufwand scheut, einen einschränkenden Regelsatz zu schreiben, kann entweder auf Yast verzichten oder es im liberalen Permissive Mode verwenden.

Fazit

Mit SE Linux kann der Admin alle Zugriffe sehr detailliert regeln. Dieser Vorteil kann sich aber auch zum Problem entwickeln: Je zahl- und umfangreicher die Regeln, desto schwieriger wird es, sie zu prüfen und zu kontrollieren. Damit werden Fehler wahrscheinlicher. Diese Gefahr lässt sich aber minimieren. Ein erster Ansatz ist, die Regeln in Gruppen einzuteilen: in originale SE-Linux-Regeln, die nicht verändert werden, in betriebssystemspezifische Regeln, die SE Linux an die eigene Distribution anpassen, sowie in systemspezifische Regeln, die lokale Feinheiten berücksichtigen. So lassen sich neue Regeln leichter und auch sicherer einpflegen.

Alternativ bietet es sich an, die Aufgaben eines Programms zu reduzieren, statt viele neue Regeln zu schreiben. Hier muss man die Ziele abwägen. Zum

einen ist es einfacher, ein paar Cronjobs zu entfernen (etwa das Update der Locate-Datenbank via »updatedb«), als dafür extra Crond-spezifische Regeln zu schreiben. Viele Aufgaben sind aber unverzichtbar und benötigen passende Regeln. (fjl) ■

Die Autoren

Carsten Grohmann ist seit dem KC87 von Computern begeistert und arbeitet seit 1997 mit Linux. Seit 2000 ist er beruflich als Systemadministrator tätig.

Konstantin Agouros beschäftigt sich seit 1989 mit Unix und dem Internet und seit 1994 mit Linux. Er leitet bei der Firma Netage das Competence Center Security.

Infos

- [1] Homepage von SE Linux bei der NSA: [<http://www.nsa.gov/selinux/>]
- [2] Komplettpaket SE Linux (36 MB): [<http://www.nsa.gov/selinux/download2.html>]
- [3] Deutschsprachige Mailingliste: [<mailto:selinux-de@vegaa.de>]
- [4] SE-Linux-Projekthomepage auf Sourceforge: [<http://sourceforge.net/projects/selinux/>]
- [5] LSM-Kernel: [<http://lsm.immunix.org>]
- [6] Alternatives Mailinglisten-Archiv: [<http://marc.theaimsgroup.com/?l=selinux>]
- [7] Patches für SE Linux: [ftp://ftp.linux-magazin.de/pub/listings/magazin/2003/01/SELinux/installpatches_20020930.tar.gz]
- [8] Regelerweiterungen für SuSE: [ftp://ftp.linux-magazin.de/pub/listings/magazin/2003/01/SELinux/suse_rules_20021105.tar.gz]
- [9] SE-Linux-Patch für KDM: [<http://www.coker.com.au/selinux/kdm/>]
- [10] Mingetty-Paket »mingetty-1.00-1.src.rpm«: [<http://www.rpmsfind.net>]
- [11] Stephen Smalley, „Configuring the SELinux Policy“: [<http://www.nsa.gov/selinux/policy2-abs.html>]
- [12] Peter Loscocco und Stephen Smalley, „Integrating Flexibel Support for Security Policies into the Linux Operating System“: [<http://www.nsa.gov/selinux/freenix01-abs.html>]
- [13] Secure Computing Corporation: [<http://www.securecomputing.com>]
- [14] Mach-Kernel: [<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/mach/public/www/mach.html>]