Die Architektur des Linux-Sicherheitssystems Rule Set Based Access Control (RSBAC)

Sicherheits-Architektur

Mehrere Sicherheitsmodelle gleichzeitig in den Kernel integrieren und auch noch alle Zugriffe detailliert protokollieren: Das freie Sicherheitssystem Rule Set Based Access Control (RSBAC) bietet angepassten Schutz für unterschiedliche Anforderungen. Amon Ott

Peter Glogg / www.visipix.com

Sicherheitslücken in Linux finden sich meist in Serverprogrammen und S-Bit-Tools. Der Königsweg wäre: Fehler vermeiden und Programme sofort aktualisieren, falls doch ein Bug auftritt. Da dies nicht immer gelingt, gilt es, den potenziellen Schaden zu begrenzen. Das ist die Aufgabe von Zugriffskontrollsystemen wie RSBAC [1]: Nutzt ein Angreifer Lücken in Servern oder S-Bit-Tools, dann soll er nur minimalen Zugang zum System erhalten. Selbst ein erfolgreicher Einbruch hat so nur begrenzte Auswirkungen. Der Schutz wird direkt im Betriebssystemkern implementiert.

Der unveränderte Linux-Kernel verhindert bereits viele Zugriffe auf diverse Ressourcen wie Dateien, Verzeichnisse oder Systemeinstellungen. Leider hat das Standardverfahren Schwächen:

- Geringe Granularität
- Discretionary Access Control
- Allmächtiger Root

Die Zugriffskontrolle von Linux kennt nur die Rechte Lesen, Schreiben und Ausführen, sie lassen sich zudem nur für den Besitzer einer Datei, die Mitglieder der Dateigruppe und alle anderen Benutzer getrennt festlegen. Für Root gelten keine Beschränkungen. Die Granularität dieser Rechte ist somit zu gering für viele Aufgaben.

Linux-Rechte: Ungenügend

Der Besitzer einer Datei hat die volle Verfügungsgewalt über sein Objekt (DAC, Discretionary Access Control). Kontrolliert ein Angreifer einen Prozess, dann gelten für seine Aktionen die Rechte des Benutzers, unter dessen Kennung der Prozess läuft. Der Angreifer kann so alle Dateien dieses Users beliebig manipulieren.

Der allmächtige Systemverwalter Root ist das gefährlichste der hier genannten Probleme. Viele Aktionen sind nur ihm vorbehalten, von Verwaltungsaufgaben bis zu einfachen Handlungen, etwa bestimmte Netzwerk-Ports öffnen. Folglich starten die meisten Serverdienste mit Root-Rechten und erhalten unbegrenzten Zugriff auf das gesamte System, ohne diese umfassenden Rechte zu benötigen. Es kommt noch schlimmer: Um später in beliebige Benutzerkennungen zu wechseln, müssen viele Dienste ständig als Root laufen oder diese Rechte jederzeit wiedererlangen können.

Die bereits seit längerem im Linux-Kernel eingeführten Posix-Capabilities erlauben es zwar einem von Root gestarteten Programm, auf einige Sonderrechte zu verzichten, das erfolgt aber freiwillig durch das Programm selbst.

Anforderungen an die Architektur

Das Hauptziel bei der Entwicklung von RSBAC war, ein flexibles und wirksames Zugriffskontrollsystem zu entwerfen, das die Linux-Mechanismen ergänzt. Um dies zu erreichen, muss das System einige Anforderungen erfüllen: Es soll als Rahmenwerk aufgebaut sein, das es dem Entwickler erlaubt, verschiedene Zugriffs-Entscheidungsmodelle einfach und schnell zu entwerfen und zu implementieren. Daraus ergibt sich eine klare Trennung zwischen entscheidenden Komponenten und solchen, die die Entscheidung durchsetzen. Die durchsetzenden Bestandteile sind unabhängig von denen, die Entscheidungen fällen. Neue Entscheidungsmodelle können die vorhandene Infrastruktur nutzen.

Es gibt inzwischen eine große Auswahl bewährter Sicherheitsmodelle für ver-

schiedenste Aufgaben, je nach Situation sind auch Kombinationen dieser Modelle sinnvoll. Das Rahmenwerk soll deshalb mehrere Sicherheitsmodelle gleichzeitig und unabhängig voneinander unterstützen, sodass der Administrator für den jeweiligen Einsatzbereich die am besten geeigneten Modelle auswählen kann. Unabhängig vom Modell ist es nötig, jede Aktion und jede getroffene Entscheidung zu protokollieren. Dieses Logging muss vor Manipulationen geschützt sein.

Bereits das erste Design des RSBAC-Systems erfüllte fast alle Anforderungen. Im Laufe der letzten fünf Jahre ist die Menge an Funktionen und überwachten Objekten stark gewachsen, auch Netzwerkzugriffe unterliegen inzwischen der RSBAC-Kontrolle. Die Grundzüge des Designs haben sich dabei bewährt und blieben unverändert.

Innere Werte

Um die interne Architektur zu beschreiben, sind einige Begriffe nötig. Aus der Sicht der Zugriffskontrolle versucht ein Subjekt, in einer bestimmten Zugriffsart auf ein Objekt zuzugreifen. In einem Linux-System liest sich das so: Ein Prozess (Subjekt) versucht eine Datei (Objekt) zu lesen (Zugriffsart).

Die verschiedenen Arten von Objekten klassifiziert RSBAC mit so genannten Zieltypen (Target Types), Tabelle 1 bietet dazu einen Überblick. RSBAC unterscheidet auch viele verschiedene Zugriffsarten (Request Types), die es jeweils auf verschiedene Objekttypen anwendet. Tabelle 2 enthält eine Auswahl von Zugriffsarten, ein Teil davon kommt später in einem Ablaufbeispiel zum Einsatz. Die gesamte Liste ist in der Dokumentation [1] zu finden.

Die Grundkomponenten des RSBAC-Systems sind in Abbildung 1 zu sehen. Die Durchsetzungskomponente (Access Control Enforcement Facility, AEF) besteht hauptsächlich aus Erweiterungen bestehender Systemfunktionen. Die Erweiterungen fordern vor jedem Zugriff eine Entscheidung von der Entscheidungskomponente (Access Control Decision Facility, ADF). Lehnt die ADF den Zugriff ab, gibt die AEF an das Subjekt den Fehler »Zugriff verweigert« zurück. Die Entscheidungskomponente und die benutzten Datenstrukturen sind von der Kernelversion weitgehend unabhängig. Nur für die AEF sind Änderungen an vorhandenen Kernelfunktionen nötig, diese Komponente ist durch Erweiterung der vorhandenen Syscalls realisiert.

Zusammenspiel der Komponenten

Die Zugriffskontrolle läuft in mehreren Schritten ab. Das Subjekt (der Prozess) ruft eine Systemfunktion auf, um auf ein Objekt zuzugreifen (1). Die Erweiterung dieser Funktion (die AEF) holt sich Informationen aus den Systemwerten, etwa die Prozess-ID sowie den Typ und die Identifikation des Zielobjekts (2).

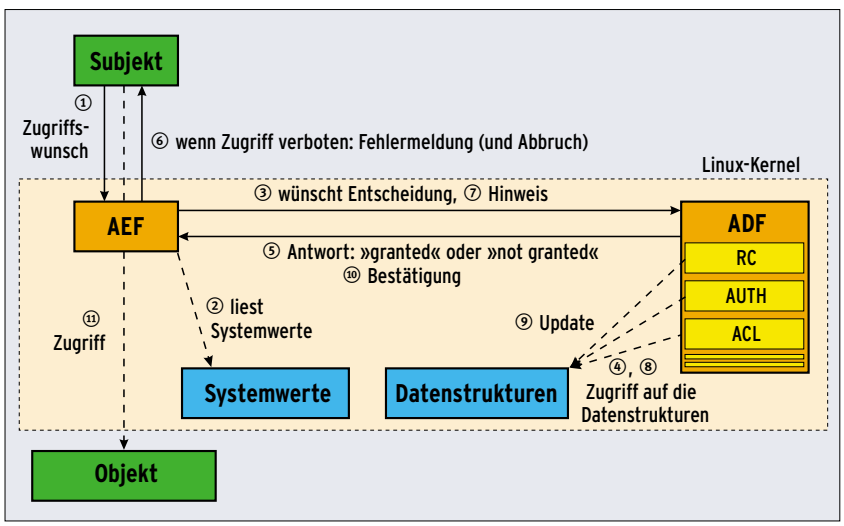

Abbildung 1: Der Zugriff eines Subjekts auf ein Objekt wird von der Access Enforcement Facility (AEF) kontrolliert. Ob der Zugriff erlaubt ist oder nicht, entscheidet die Access Decision Facility (ADF).

Anschließend ruft sie die Entscheidungskomponente ADF auf und übergibt ihr die gesammelten Systeminformationen sowie die Art des Zugriffs (3).
Die Anfrage ist zunächst an die zentrale Entscheidungsfunktion der ADF gerichtet. Diese Funktion fordert von allen aktiven Entscheidungsmodulen eine Einzelentscheidung an. Die Module holen sich die jeweils benötigten Attribute aus den Datenstrukturen (4) und fällen eine Entscheidung: erlaubt, egal oder verboten. Die Zentralfunktion fasst die einzelnen Urteile zusammen und liefert eine Gesamtentscheidung zurück (5). Die ADF verhält sich dabei restriktiv: Es genügt, wenn ein einzelnes Modul eine negative Antwort gibt, die ADF verbietet dann den Zugriff. Nur wenn alle Module zustimmen, erlaubt sie die Aktion.
Ist die Entscheidung negativ, bricht der Systemaufruf ab und liefert einen Zugriffsfehler an den Prozess zurück (6). Bei einer positiven Entscheidung verzweigt die AEF zum eigentlichen Systemaufruf. Ist er erfolgreich, sendet die AEF an die ADF einen Hinweis (7). In der ADF ist die zentrale Benachrichtigungsfunktion zuständig, sie ruft alle entsprechenden Modulfunktionen auf. Diese holen sich die aktuellen Attribute aus den Datenstrukturen (8), aktualisieren sie (9) und bestätigen den korrekten Ablauf (10).
Hat der Systemaufruf ein neues Objekt erzeugt, enthält die Nachricht der AEF an die ADF auch den Typ und die ID des neuen Objekts. Die Entscheidungsmodule erzeugen dann die Attribute dieses Objekts. Nach der Bestätigung gibt die Systemfunktion die angeforderten Daten (11) und die Kontrolle an den aufrufenden Prozess zurück.

Tabelle 1: Einige Objekttypen (Target Types)

Name	Beschreibung
FILE	Datei, hierzu zählen auch Geräte- und Unix-Netz-Spezialdateien, wenn sie als Dateien behandelt werden
DIR	Verzeichnis
FIFO	Pipe mit Namenseintrag im Dateisystem
SYMLINK	Symbolischer Link
IPC	Inter Process Communication, Prozesskommunikationsobjekt nach System V
SCD	System Control Data, globale Systemeinstellungen und -Objekte wie Rechnername oder Uhrzeit
USER	Benutzer als Objekt, hauptsächlich um die Attributvergabe zu kontrollieren
PROCESS	Prozess als Objekt, zum Beispiel beim Empfangen von Signalen oder beim Auslesen des Prozesszustands
NETDEV	Netzwerkgerät
NETTEMP	Netzwerkschablone
NETOBJ	Netzwerkobjekt, in der Regel Sockets

Ein Ablaufbeispiel

Der theoretische Ablauf lässt sich gut an einem Beispiel nachvollziehen. Will ein Prozess eine Datei zum Lesen und Schreiben öffnen, benutzt er den Systemaufruf »sys_open()« mit einigen Parametern. Beispielsweise legen die Parameter fest, dass »sys_open()« die Datei erzeugt, wenn sie noch nicht existiert, oder die Länge auf null setzt, wenn das File bereits vorhanden ist.
Sollte die ADF eine der folgenden Entscheidungsanfragen ablehnen, bricht der Systemaufruf mit der Fehlermeldung »Zugriff verweigert« ab. Als Erstes muss »sys_open()« den Dateinamen auflösen, um den Inode zu ermitteln. Diese Aufgabe übernimmt eine Hilfsfunktion, deren RSBAC-Erweiterung für jedes berührte Verzeichnis eine »SEARCH«-Anfrage an die ADF stellt. Ist die Datei noch nicht vorhanden, stellt die Erweiterung der Open-Funktion eine »CREATE«-Anfrage für das Zielverzeichnis, erzeugt die Datei und informiert die Entscheidungskomponente über das neue Objekt. Andernfalls stellt sie eine »TRUNCATE«-Anfrage für die Datei, die Open-Funktion setzt die Länge auf null und übermittelt den Erfolg der Operation.
Nach diesen Vorarbeiten generiert der Syscall eine »READ_WRITE_OPEN«-Anfrage und öffnet die Datei. Die ADF erfährt vom Öffnen der Datei und aktualisiert erneut ihre Attribute. Danach erhält der Prozess seinen Dateideskriptor und kann mit der Arbeit fortfahren.

Aufbau der Datenhaltung

Wie beschrieben dienen als Grundlage aller Zugriffsentscheidungen so genannte Attribute, die allen Benutzern, Prozessen und Objekten zugeordnet sind. Das Verwalten der Attribute ist die Aufgabe der allgemeinen Datenhaltung. Zusätzlich gibt es modellspezifische Daten, etwa Gruppen oder Zugriffsmatrizen, die nicht in generische Strukturen passen. Für diese Daten ist eine modellspezifische Datenhaltung zuständig.
Die Datenhaltungskomponente entlastet die einzelnen Module von der undankbaren Listenverwaltung und kümmert sich um das Speichern auf Festplatte, um SMP-Locking (bei Mehrprozessorsystemen) und Ähnliches. Sie legt fast alle Daten in einem generischen Listensys-

Tabelle 2: Einige Zugriffsarten (Request Types)

Name	Objekttypen	Beschreibung
BIND	NETDEV, NETOBJ	Binden einer Netzwerkadresse
CLOSE	FILE, DIR, FIFO, DEV, IPC, NETOBJ	Schließen eines Datei-Deskriptors
CONNECT	NETOBJ	Verbindung zu fernem Netzwerkendpunkt aufbauen
CREATE	DIR (wo), IPC, NETTEMP, NETOBJ	Objekt erzeugen
DELETE	FILE, DIR, FIFO, IPC, NETTEMP	Objekt löschen
EXECUTE	FILE	Datei ausführen
NET_SHUTDOWN	NETOBJ	Verbindungskanal schließen
READ	DIR, SYMLINK, IPC, NETTEMP (optional: FILE, FIFO, DEV, NETOBJ)	Lesen vom Objekt
READ_WRITE_OPEN	FILE, FIFO, DEV, IPC	Öffnen zum Lesen und zum Schreiben
RECEIVE	NETOBJ	Daten vom fernen Netzwerkendpunkt empfangen
SEARCH	DIR, SYMLINK	Namensauflösung
SEND	NETOBJ	Daten an fernen Netzwerkendpunkt senden
TRUNCATE	FILE	Länge einer Datei ändern

tem ab, das es erlaubt, beliebig viele ein- und zweistufige Listen (Listen von Unterlisten) mit beliebiger Größe der Indizes und der Datenfelder zu registrieren. Die Entscheidungsmodule registrieren ihre Listen in der Regel bei der RSBAC-Initialisierung oder beim Einbinden von Dateisystemen. Nur einige wenige Listen sind wegen besonderer Anforderungen anders implementiert.

Dauerhafte Daten

Auf Wunsch halten die generischen Listen die Daten auch persistent, das heißt, sie gehen durch einen Neustart des Systems oder ein Deregistrieren nicht verloren. Der Kernel-Daemon »rsbacd« speichert dazu regelmäßig alle als geändert markierten Listen in speziell geschützten Verzeichnissen auf der Festplatte, von wo sie die Datenhaltung beim nächsten Registrieren der Liste auch wieder abholt. Ein Parameter legt beim Registrieren fest, auf welcher Partition diese Dateien liegen sollen, um eine gezielte Zuordnung zu Dateisystemobjekten zu ermöglichen.

Optional können die Module beim Registrieren einer Liste auch einen Standardwert angeben. Fehlt später ein Listenelement, liefert die Datenhaltung diesen Wert zurück. Zur Optimierung löscht sie alle Elemente, die den Standardwert enthalten und unbegrenzt gültig sind. Ändert sich der Wert, legt die Datenhaltung das Element wieder neu an. Bei zweistufigen Listen legt sie bei Bedarf ganze Unterlisten an oder löscht sie. Dies Verfahren hält die Länge der Listen stets auf dem Minimum und verkürzt damit die Zugriffszeiten.

Jedes Listenelement erhält beim Erzeugen und beim Aktualisieren eine Lebensdauer zugeteilt, nach der es automatisch entfernt wird. Diese Eigenschaft verwen-

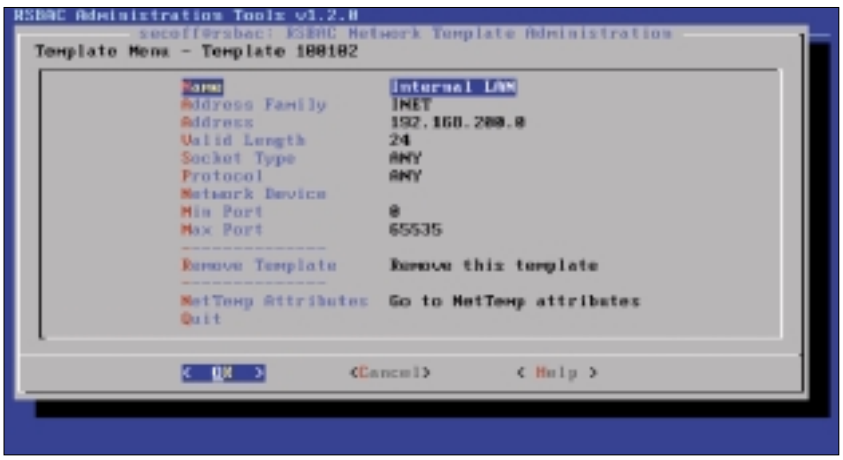

Abbildung 2: Über Netzwerk-Templates vergibt der Admin Zugriffsrechte auf ganze Netzblöcke und Port-Bereiche. Im Bild sind alle Ports auf allen Rechnern im IP-Netz 192.168.200.0/24 ausgewählt.

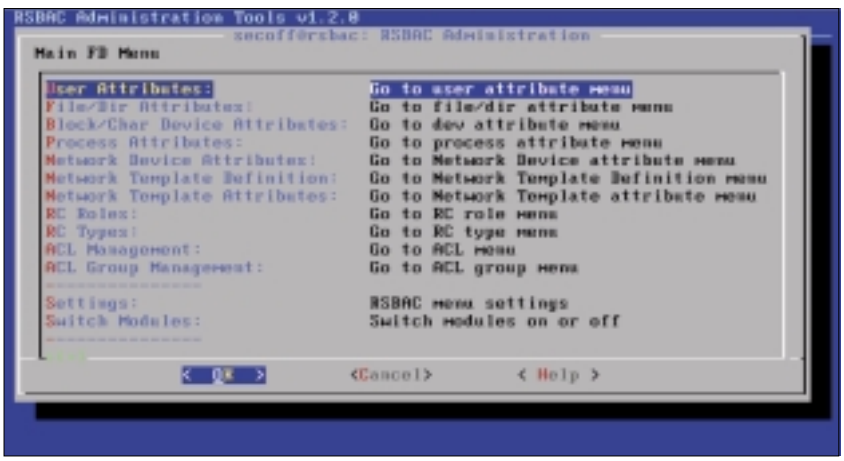

Abbildung 3: RSBAC lässt sich in allen seinen Aspekten bequem konfigurieren, im Bild ist das Hauptmenü für die Administration zu sehen.

den einige Entscheidungsmodule für befristete Einstellungen oder Rechte. Der Spezialwert »0« sorgt für eine unbegrenzte Lebenszeit.

Implementiert sind die generischen Listen als doppelt verkettete, geordnete Listen mit registrierbarer Ordnungs- und Vergleichsfunktion für optimierten Zugriff. Enthält die Registrierung keine solche Funktion, kommt der einfache Speichervergleich mit »memcmp()« zum Zuge. In [3] findet sich eine genauere Beschreibung der Schnittstellen und Parameter der Listenverwaltung.

Schablonenhafte Regeln

Netzwerkverbindungen sind in der Regel recht kurzlebig, viele Datenpakete werden sogar einzeln und unabhängig voneinander verschickt. Das macht es besonders schwierig, ihnen Attribute zuzuordnen. Der Verwaltungsaufwand wäre unangemessen hoch. In RSBAC gibt es daher die so genannten Netzwerkschablonen (Templates). Sie beschreiben viele Netzwerkendpunkte anhand verschiedener Kriterien wie Protokollfamilie, Verbindungstyp, Netzprotokoll oder Portnummern. Abbildung 2 zeigt exemplarisch ihre Definition.

RSBAC speichert die Attribute in diesem Fall nicht für jeden einzelnen Endpunkt getrennt oder für jede Verbindung, sondern pauschal in den Schablonen. Die Endpunkte des Netzwerks (also Quelle oder Ziel einer Datenübertragung) erben die Werte von der am besten passenden Schablone (die mit der niedrigsten Ordnungsnummer). Somit kann die ADF über einen Zugriff wie »CONNECT« leicht anhand der Schablonen-Attribute für die Quell- oder Zieladresse entscheiden, sie muss nur die richtige Schablone suchen.

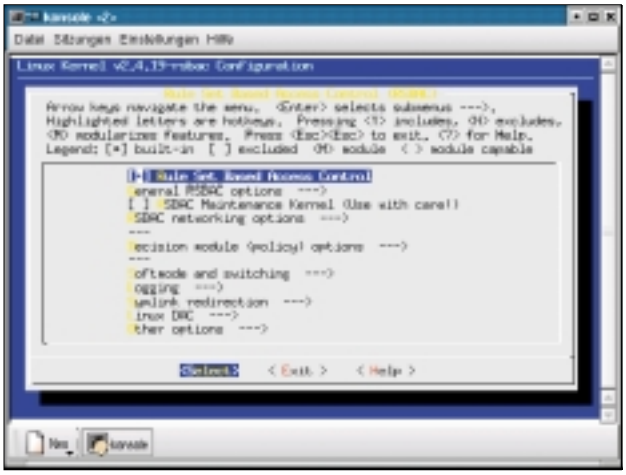

Abbildung 4: **RSBAC muss im Linux-Kernel aktiviert werden. Ein eigenes Konfigurationsmenü ist für die Grundeinstellung zuständig.**

Administration

Im Schablonenverfahren lässt sich leicht festlegen, dass etwa ein bestimmter Benutzer nur mit dem Internet-Protokoll TCP auf das lokale Netzwerk zugreifen darf oder ein Browser nur auf den Port des HTTP-Proxies auf der Firewall. Es ist nicht nötig, jede denkbare Verbindung einzeln zu konfigurieren.

Da RSBAC sämtliche Einstellungen ausschließlich im Kernel oder in geschützten Dateien speichert, kann die Administration nur über Systemaufrufe oder das »/proc«-Dateisystem erfolgen. Nur so behält der Kernel die volle Kontrolle darüber, welcher Benutzer welche Detaileinstellung ändert. Damit löst RSBAC das Hauptproblem des allmächtigen Sys-

Entscheidungsmodule in RSBAC

Die aktuelle, stabile RSBAC-Version 1.2.1 enthält die folgenden Entscheidungsmodule und Regelsätze, die zum Teil komplexe Sicherheitsmodelle implementieren.

MAC, Mandatory Access Control: Mandatorische Zugriffskontrolle nach Bell-La Padula.

FC, Functional Control: Dieses einfache Rollenmodell erlaubt Zugriffe auf Sicherheitsinformationen nur für Sicherheitsbeauftragte und gewährt nur Administratoren den Zugriff auf Systeminformationen.

SIM, Security Information Modification: Nur Sicherheitsbeauftragte dürfen Daten ändern, die als Sicherheitsinformationen gekennzeichnet sind.

PM, Privacy Model: Das Datenschutzmodell von Simone Fischer-Hübner setzt die deutschen und europäischen Datenschutzrichtlinien technisch um.

MS, Malware Scan: Überprüft alle Dateien beim Lesen und Ausführen auf bösartige Software. Version 1.2.1 enthält nur einen Scanner-Prototyp, Vorversion 1.2.2-pre1 benutzt die professionelle Antivirus-Software von F-Prot. Die Unterstützung weiterer Scanner ist geplant.

FF, File Flags: Globale Attribute gelten für Dateien und ganze Verzeichnisse, etwa »execute_only«, »no_execute«, »read_only«, »append_only«.

RC, Role Compatibility: Dieses mächtige Rollenmodell wurde extra für den Einsatz auf Linux-Servern entworfen. Es definiert Rollen für Benutzer und Programme sowie Typen für alle Arten von Objekten. Für jede Rolle können dann die Zugriffsrechte auf alle Typen genau festgelegt werden. Das Modell enthält auch ein Schema zur klaren Aufteilung von Administrationsaufgaben auf mehrere Rollen sowie Zeitlimits für alle Zugriffs- und Administrationsrechte.

AUTH, Authentication Enforcement: Dieses Modul kontrolliert alle »CHANGE_OWNER«-Anfragen für Prozesse und damit alle »setuid()«-Aufrufe. Nur die für ein Programm oder einen Prozess freigeschalteten Benutzer-IDs sind erreichbar.

ACL, Access Control Lists: Jedes Objekt erhält eine Zugriffskontrollliste, die die zulässigen Zugriffsarten für verschiedene Subjekte festlegt. Subjekte sind Benutzer-IDs, RC-Rollen und ACL-Gruppen. Enthält ein Objekt für ein Subjekt keinen passenden Eintrag, erbt es die Rechte vom übergeordneten Objekt, zum Beispiel von einem Verzeichnis. Eine Vererbungsmaske am Objekt filtert die Vererbung, damit lassen sich die gewährten Rechte für alle Subjekte einschränken. Das ACL-Modell bietet außerdem übergeordnete Default-ACLs, individuelle Gruppenverwaltung für jeden Benutzer sowie Zeitlimits für alle vergebenen Rechte und Gruppenmitgliedschaften.

CAP, Linux Capabilities: Erlaubt es, minimale und maximale Mengen von Linux-Capabilities (aufgeteilte Root-Rechte) für alle Benutzer und Programme zu definieren. Damit können zum Beispiel Serverprogramme als normaler Benutzer oder als Root-Programm mit weniger Rechten laufen.

JAIL, Process Jails: Dieses Modul bietet den neuen Systemaufruf »rsbac_jail()«, der im Prinzip eine Erweiterung der FreeBSD-Jails darstellt. Hiermit gestartete Programme befinden sich eingekapselt in einer Chroot-Umgebung mit stark beschränkten Administrations- und Netzwerkrechten.

temverwalters: Denn wären die Konfigurationsdaten in normalen Files abgelegt, hätte jeder User mit Schreibzugriff auf diese Dateien auch alle Administrationsrechte. In RSBAC können verschiedene Administratoren unterschiedliche Rechte haben.

Freiwillige Selbstkontrolle

Bis auf wenige Ausnahmen kontrolliert jedes Entscheidungsmodul eigenverantwortlich seine eigenen Attribute. Insbesondere die wissenschaftlich fundierten Modelle RC und ACL (siehe Kasten „Entscheidungsmodule in RSBAC“) unterstützen gezielt das Verteilen von Administrationsaufgaben auf mehrere User. Root hat zwar in der Default-Konfiguration der meisten Module besondere Rechte, ist sonst aber ein normaler Benutzer wie alle anderen auch.

Da die Kontrolle über die verwendeten User-IDs für alle Entscheidungsmodule sehr wichtig ist, gibt es das Unterstützungsmodul »AUTH«. Es gibt User-IDs für bestimmte Programme und Prozesse gezielt frei. Ein Prozess kann nur IDs annehmen, die »AUTH« für ihn freigibt, andere sind nicht erreichbar.

Um RSBAC zu verwalten, gibt es eine Reihe von Hilfsprogrammen. Sie erleichtern viele Aufgaben und dienen als Benutzerschnittstelle zu den RSBAC-Systemaufrufen. Menü-Oberflächen vereinfachen die Bedienung noch zusätzlich, Abbildung 3 zeigt als Beispiel das Hauptmenü »rsbac_menu«.

RSBAC ist wohl das älteste und – schon beim Code – umfangreichste freie Zugriffskontrollsystem für den Linux-Kernel. Nur durch die klare und modulare Struktur war es möglich, den Überblick zu behalten. Inzwischen ist RSBAC vor allem in Europa bekannt und verbreitet. Nach einer vorsichtigen Schätzung ist es auf mehreren hundert Produktivsystemen im Einsatz. *(fjl)* ■

Infos

[1] RSBAC-Homepage: [http://www.rsbac.org]

[2] Amon Ott, „Regelsatzbasierte Zugriffskontrolle nach dem GFAC-Ansatz am Beispiel Linux“, Diplomarbeit: [http://www.rsbac.org/diplarb.pdf]

[3] Schnittstellenbeschreibung des generischen Listensystems: [http://www.rsbac.org/lists.htm]

[4] Amon Ott, „Root ist nicht alles – Mehr Sicherheit für Linux-Server mit RSBAC“, iX 8/02, S. 99.

Der Autor

Amon Ott ist selbstständiger Diplom-Informatiker und Autor des RSBAC-Systems. Seine Brötchen verdient er überwiegend mit Auftragsentwicklungen und Linux-Firewalls, bevorzugt mit RSBAC. Nebenher schreibt er an seiner Dissertation zum Thema, die hoffentlich bald fertig wird.

Installation

Um RSBAC zu installieren, sind die Quellen von der Homepage [1] herunterzuladen. Sie bestehen aus drei Teilen: Ein Tar-Archiv enthält die Module, die von der Kernelversion unabhängig sind. Dazu kommen ein Kernel-Patch (versionsabhängig) und ein Tar-Archiv mit den Administrationsprogrammen. Das RSBAC-Patch enthält hauptsächlich den Initialisierungsaufruf und ergänzt die Syscalls um die AEF-Aufgaben. Alternativ gibt es auch fertig zusammengestellte Kernelquellen als Bzip-2-gepacktes Tar-Archiv.

Die von den meisten Distributoren gelieferten Kernel sind leider heftig modifiziert und führen öfter zu Problemen, hier hilft meist nur das Original von [ftp://ftp.de.kernel.org/pub/linux/kernels] oder einem Spiegel.

Nach dem Auspacken des Tar-Archivs im Hauptverzeichnis der Kernel-Sourcen und dem Einspielen des Patches wird der Kernel wie üblich konfiguriert, kompiliert und installiert. Das in Abbildung 4 dargestellte zusätzliche Menü »Rule Set Based Access Control« enthält mehrere Untermenüs mit einer Fülle von Optionen, jeweils mit ausführlichen Hilfetexten. In den meisten Fällen sind die Vorgaben akzeptabel.

Beim ersten Neustart erlaubt es der Kernelparameter »rsbac_auth_enable_login« dem Login-Programm, in beliebige Benutzerkennungen zu wechseln – damit kann dieses Programm User anmelden. Für erste Tests ist der Parameter »rsbac_softmode« hilfreich, der die Entscheidungen nur protokolliert, aber nicht durchsetzt.

Ist das System erfolgreich gestartet, sind noch die Hilfsprogramme auszupacken und wie üblich mit »./configure && make && make install« zu kompilieren und zu installieren. Sollten die RSBAC-Kernelquellen sich nicht unter »/usr/src/linux« befinden, hilft der Configure-Parameter »--with-kerneldir«.