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

Wink mit dem Zaunpfahl

RSBAC schützt auch vor unerwarteten Gefahren: Für alle Anforderungen bietet das Linux-Sicherheitssystem den passenden Zugriffsschutz. Der Admin kann die verschiedenen Sicherheitsmodelle sogar kombinieren. Amon Ott

Die Zugriffsregeln eines Betriebssystems bestimmen, welcher Prozess wie auf welche Objekte zugreifen darf. Der erste Artikel [2] über die RSBAC-Architektur (Rule Set Based Access Control) erläuterte, wie und warum sie mehrere Sicherheitsmodelle gleichzeitig in den Standardkernel integriert. Die Fortsetzung beschreibt im Folgenden die wichtigsten Modelle.

RSBAC besteht im Wesentlichen aus zwei Teilen: Die Access Enforcement Facility (AEF) überwacht die Zugriffe im System und setzt die Entscheidungen der Access Decision Facility (ADF) um. Wie die ADF zu ihrer Entscheidung kommt, hängt vom jeweiligen Sicherheitsmodell ab. Je nach Art der Anforderungen ist mal das eine, mal das andere besser geeignet – oft ist eine Kombination mehrerer Modelle die beste Wahl.

Diskret oder mandatorisch

Bei der klassischen Zugriffskontrolle in Linux hat der Besitzer einer Datei die volle Verfügungsgewalt über sein Objekt. Damit hängt die Sicherheit der enthaltenen Daten nur von der Diskretion (dem Ermessen) des Besitzers ab, man spricht deshalb von diskreter Zugriffskontrolle. Im Gegensatz dazu verlangt eine mandatorische Zugriffskontrolle, dass die Erzeuger und Bearbeiter eines Objekts keinen direkten Einfluss auf dessen Zugriffsrechte haben. Stattdessen entscheiden das verwendete Modell und der Sicherheitsadministrator über alle Sicherheitseinstellungen.

Die in RSBAC enthaltenen Modelle sind, bis auf einzelne Details, mandatorisch und erfordern jeweils einen unabhängigen Sicherheitsadministrator (Security Officer). In der Grundkonfiguration ist das für alle Modelle die Benutzer-ID 400. Einzelne Modelle erlauben es sogar, die Administration auf mehrere Personen zu verteilen, sodass sie nur gemeinsam ernsthaften Schaden anrichten können. Dieses Prinzip nennt sich Aufgabentrennung (Separation of Duty).

Die einzelnen Modelle sind in Entscheidungsmodulen implementiert, die je nach Bedarf in das RSBAC-Rahmenwerk eingeklinkt werden. Tabelle 1 zeigt einen kurzen Überblick aller Entscheidungsmodule der RSBAC-Version 1.2.2. Im Folgenden werden die Module AUTH, FF, RC, ACL, CAP und RES vorgestellt.

Benutzerkontrolle mit AUTH

Die meisten Zugriffskontrollmodelle vergeben Rechte direkt oder indirekt an Benutzer. Für die Sicherheit des Systems ist es deshalb nötig, Benutzer eindeutig zu identifizieren und streng zu kontrollieren, unter welchen User-Kennungen ein Prozess arbeiten darf. In einem Standard-Linux findet diese Kontrolle kaum statt: Jeder als Root laufende Prozess darf in jede beliebige Benutzerkennung wechseln. Root-Rechte sind aber für viele Aufgaben erforderlich. Dieses Problem lässt sich ohne Änderungen am Kernel nicht lösen.

Als Dienstleistung für alle Entscheidungsmodule kontrolliert das AUTH-Modul (Authentication Enforcement), welches Programm in welche Benutzerkennung wechseln darf. Ein AUTH-Sicherheitsadministrator gibt dazu der Programmdatei so genannte AUTH-Capabilities mit, das sind Mengen erreichbarer Benutzerkennungen. Sobald ein Prozess die Programmdatei startet, übernimmt er deren Capabilities und ersetzt damit seine bisherigen.

Während der Ausführung dürfen speziell gekennzeichnete Programme die Capabilities eines Prozesses ändern. Damit kann ein zentraler Authentisierungs-

Daemon einzelnen Prozessen gezielt erlauben, nach erfolgreicher Benutzer-Authentisierung in diese User-Kennung zu wechseln – aber in keine andere. Ab Version 1.2.2 sind sogar zeitlich begrenzte AUTH-Capabilities möglich.

Ein gesichertes System ist meist so konfiguriert, dass das AUTH-Modul nur einen Weg zulässt, um sich als System- oder Sicherheitsadministrator einzuloggen: das Anmeldeprogramm auf der lokalen Konsole. In manchen Fällen ist auch ein direkter Zugang per SSH sinnvoll, dann vergibt der Sicherheitsadmin die entsprechenden Capabilities auch an den SSH-Daemon (wenn SSH als »UseLogin« konfiguriert ist, muss »/bin/login« diese Capabilities erhalten). Auf diese Weise lassen sich auch andere Serverdienste auf bestimmte Kennungen mit angepassten Rechten fixieren.

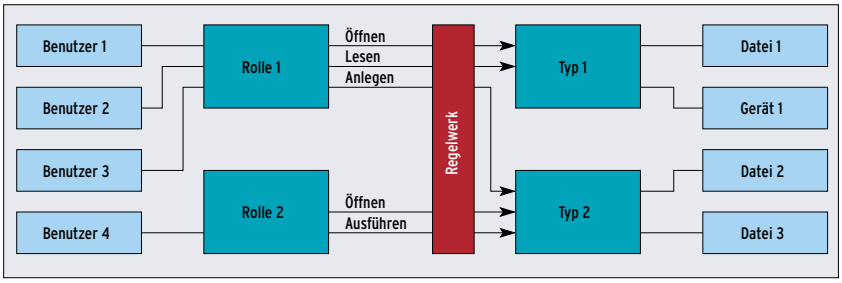

Abbildung 1: Die Regeln im RC-Modell legen fest, welche Rollen auf welche Typen wie zugreifen dürfen. Rollen fassen Benutzer zusammen, Typen sind Gruppen von Objekten.

Root darf nicht mehr alles

Das AUTH-Modul kontrolliert natürlich auch die Programme »su«, »sudo« und alle Zeit-gesteuerten Tools wie »cron« oder »atd«. Somit kann Root nicht mehr beliebige Kennungen erreichen, sondern nur noch die explizit vom Sicherheitsadministrator freigeschalteten. Aus nahe liegenden Gründen sollte Letzterer dafür sorgen, dass seine eigene Kennung möglichst geschützt ist und es nur gut kontrollierte Wege gibt, um in die User-ID des Security Officers zu wechseln.

Da alle Module von den Einstellungen des AUTH-Moduls abhängen, dürfen auch alle Module über sämtliche Änderungen an den AUTH-Parametern entscheiden. Dazu generiert das Modul jeweils eine Entscheidungsanfrage an die restlichen Komponenten.

Globale Attribute mit FF

Das Modul File Flags (FF) verwaltet Datei- und Verzeichnisattribute, die für alle Benutzer gleichermaßen gelten. Diese Attribute können zum Beispiel einen ganzen Verzeichnisbaum vor dem Beschreiben schützen oder in sämtlichen Homeverzeichnissen das Ausführen von Programmen verhindern.

Ein spezielles, per Default gesetztes Attribut erlaubt das Erben der Werte vom übergeordneten Verzeichnis. Allerdings gelten einige Attribute nur für spezielle Objekttypen (etwa Dateien oder Verzeichnisse). Tabelle 2 zeigt die in RSBAC bekannten Objekttypen, Tabelle 3 gibt einen Überblick über sämtliche Attribute und zeigt, für welche Objekttypen sie gelten.

Das File-Flags-Modul eignet sich besonders für Systeme, an denen der Admin wenig verändert. Ein »read_only«-Flag auf »/bin«, »/sbin«, »/usr/bin«, »/usr/sbin«, »/boot« und »/etc/init.d« schützt bereits die meisten Programme vor Manipulationen. Das »no_execute«-Flag auf »/home«, »/tmp« und »/var/tmp« verhindert, dass normale Benutzer eigene Programme ausführen. Im RSBAC-Menü für Dateisystemobjekte (»rsbac_fd_menu«) kann der Security Officer unter dem Menüpunkt »FF Flags« die gewünschten Attribute bequem auswählen und setzen.

Rollen und Typen mit RC

Das zur Zeit mächtigste und flexibelste RSBAC-Sicherheitsmodell ist Role Compatibility (RC). Trotzdem ist es für einfache Fälle leicht zu verstehen und zu benutzen – komplex wird es erst bei besonderen Aufgaben.

Das RC-Modell abstrahiert Benutzer zu Rollen und Objekte zu Typen. Eine Rolle beschreibt, welche Aufgabe ein Benutzer im System erfüllt. Üblicherweise haben mehrere User sehr ähnliche Aufgaben

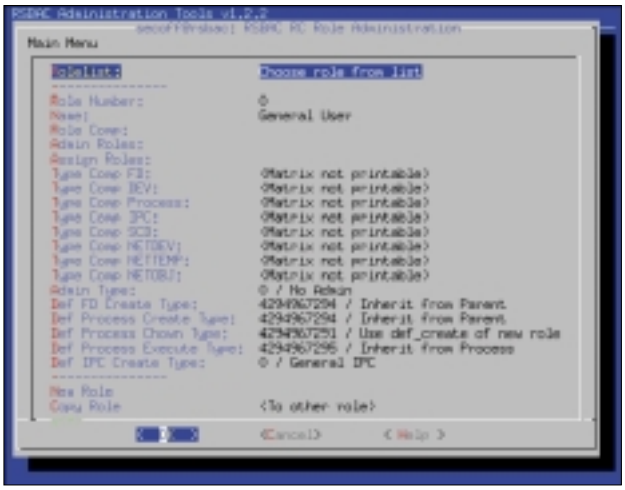

Abbildung 2: Das RC-Rollenmenü erleichtert die Rollendefinition. Es zeigt die Rolle »0«, General User genannt, und ihre Rechte.

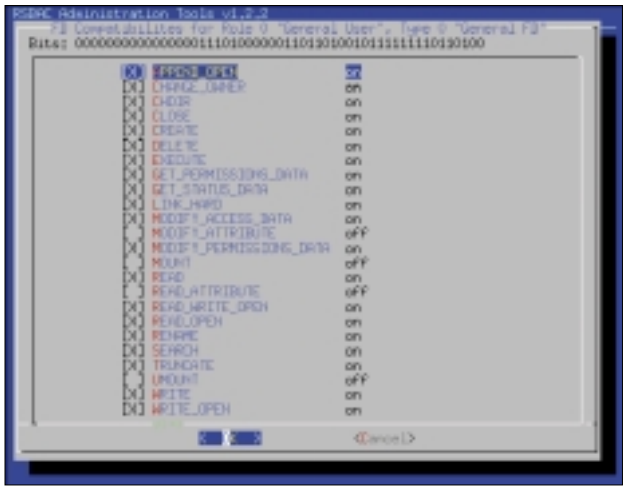

Abbildung 3: Im RC-Rollenmenü wählt der Admin aus, welche Zugriffsrechte eine Rolle (hier »0«, General User) auf einen Typ (hier »0«, General FD) besitzt.

und lassen sich daher gut in einer Rolle zusammenfassen. Mögliche Rollen sind Buchhalter, Systemadministrator oder Programmierer. Wechselt ein Prozess seine User-ID, erhält er automatisch die Rolle des neuen Benutzers. Ein Prozess hat immer genau eine Rolle, die er auch an Kindprozesse weitergibt.

Auf die gleiche Weise fasst dieses Modell die Objekte zu Typen zusammen. Dort gibt es zum Beispiel Benutzer- und Systemprogramme, Konfigurationsdateien oder Benutzerverzeichnisse. RC verwendet getrennte Mengen von Typen für Dateisystemobjekte (FD), Geräte (DEV), Prozesse (PROCESS), für die Prozesskommunikation (IPC), Systemeinstellungen (SCD) sowie Netzwerkgeräte (NETDEV), Netzwerkobjekte (NETOBJ) und Netzwerkschablonen (NETTEMP). Die einzelnen Objekttypen wurden in [2] bereits näher erläutert.

Der Sicherheitsadministrator mit der Security-Officer-Rolle definiert zunächst die benötigten Rollen und Typen. Er legt für jede Rolle fest, auf welche Typen ein Prozess auf welche Weise zugreifen darf, wenn er sich in dieser Rolle befindet (Abbildung 1). Die möglichen Zugriffsarten sind die vom RSBAC-Rahmenwerk vorgegebenen Request Types, sie werden hier Standardrechte genannt. Das in den Abbildungen 2 und 3 gezeigte Menü erleichtert diese Arbeit.

Angenehm ist die Möglichkeit, vorhandene Rollen mit allen ihren Rechten zu kopieren und dann anzupassen. Außerdem kann man die Rechte, die alle Rollen auf einen Typ haben, zu einem anderen Typ kopieren. Diese Tricks erleichtern später das Aufspalten von Rollen und Typen, wenn der Admin seine Rechtevergabe verfeinert.

Schließlich weist der Sicherheitsadministrator die Rollen den vorhandenen Benutzern und die Typen den Objekten zu. Auch hier sind wieder die RSBAC-Menüs hilfreich, in diesem Fall das Be-

Tabelle 1: Entscheidungsmodule

Name	Beschreibung
MAC	Mandatory Access Control: Mandatorische Zugriffskontrolle nach Bell-La Padula
FC	Functional Control: Einfaches Rollenmodell
SIM	Security Information Modification: Nur Sicherheitsbeauftragte dürfen Daten ändern, die als Sicherheitsinformation 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
FF	File Flags: Globale Attribute gelten für Dateien und ganze Verzeichnisse
RC	Role Compatibility: Mächtiges Rollenmodell
AUTH	Authentication Enforcement: Kontrolliert die Benutzer-IDs
ACL	Access Control Lists: Zugriffskontrolllisten
CAP	Linux-Capabilities: Vergibt Linux-Capabilities an Benutzer und Programme und versteckt Prozesse
JAIL	Process Jails: Vorkonfektionierte Programmkäfige
RES	Resource Control: Vergibt Ressourcen-Limits an Benutzer und Programme

Tabelle 2: 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 (named pipe)
SYMLINK	Symbolischer Link
IPC	Inter Process Communication: Prozesskommunikationsobjekt nach System V
SCD	System Control Data: Globale Systemeinstellungen und -Objekte wie Rechnername und 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

nutzermenü »rsbac_user_menu« und das Menü für die Dateisystemobjekte »rsbac_fd_menu«.

Neue Objekte erhalten ihren Typ automatisch entsprechend den Einstellungen der aktuellen Rolle. Zuständig dafür sind die Rollenattribute

- »default_fd_create_type« für neue Datei-Objekte,
- »default_process_create_type« für die neuen Prozesse und
- »default_ipc_create_type« für die Prozesskommunikation.

Die Prozesstypen ändern sich zusätzlich auch beim Ausführen neuer Programme (»default_process_execute_type«) und beim Wechseln der Benutzer-ID (»default_process_chown_type«). Der Typ eines neuen Netzwerkobjekts steht im RC-Typ-Attribut des passenden Netzwerktemplates.

In der Grundeinstellung vererbt sich der Dateisystemtyp eines Verzeichnisses an seine Unterverzeichnisse und die enthaltenen Dateien. Neue Benutzer haben immer die Standardrolle »0«, da der Kernel keine Benutzer-Accounts kennt, sondern nur 32-Bit-IDs. Hier muss der Sicherheitsadministrator eventuell noch eine sinnvolle Rolle zuweisen.

Tabelle 3: Attribute für Dateisystem-Zieltypen

Name	Targets	Beschreibung
execute_only	FILE, FIFO, SYMLINK	Nur ausführen (also kein Zugriff bei FIFO und SYMLINK)
search_only	DIR	Nur Namensauflösung
read_only	alle	Nur lesen und ausführen
write_only	FILE, FIFO, SYMLINK	Nur schreiben
secure_delete	FILE	Beim Löschen und Kürzen den alten Inhalt erst überschreiben
no_execute	FILE	Nicht ausführen
no_delete_or_rename	alle	Weder Löschen noch Umbenennen erlaubt (wird nicht vererbt)
append_only	FILE, FIFO, SYMLINK	Nur Anhängen als einziger Schreibzugriff, Lesen bleibt erlaubt
add_inherited	alle	Füge die im Vaterverzeichnis gesetzten Attribute hinzu (wird nicht vererbt)

Rollen für Programme

Die bisher beschriebenen Einstellungen genügen zwar für die meisten Fälle, das RC-Modell kann aber noch eine Menge mehr. Die wichtigste Spezialität ist das Zuweisen von Rollen an Programme. RC kennt initiale und erzwungene Programmrollen, die der Sicherheitsadmin vergibt. Startet ein Prozess ein Programm, dem der Admin eine initiale Rolle zugeteilt hat, so läuft der Prozess mit dieser Rolle weiter, unabhängig von seiner bisherigen. Anschließend verläuft alles wie gehabt: Ändert sich die Benutzerkennung, übernimmt der Prozess die Rolle des Users.

Initiale Rollen eignen sich besonders für Login-Programme, die für die Authentisierung eine besondere Rolle mit Zugriffsrechten auf die Authentisierungsdaten benötigen, aber danach mit der Rolle des gewählten Benutzers weiterarbeiten sollen. Ab Version 1.2.2 funktioniert das sogar dann, wenn die Benutzer-ID gleich bleibt – ein Setuid auf dieselbe ID reicht dafür völlig aus. Im Gegensatz dazu behält ein Prozess, der ein Programm mit erzwungener Rolle startet, diese Rolle mindestens bis zum Start des nächsten Programms bei, auch wenn er die Benutzerkennung wechselt. Diese Eigenschaft hilft beim strengen Einkapseln eines Serverprogramms, das seine eingeschränkte Rolle niemals verlassen darf.

Wenn nötig, gibt es trotzdem einen Weg aus der erzwungenen Rolle, der auch in allen anderen Fällen funktioniert: Der Sicherheitsadministrator kann für jede Rolle so genannte kompatible Rollen definieren, in die der Prozess aktiv durch einen Systemaufruf wechseln darf. Daraus ergeben sich ganze Ketten von erreichbaren Rollen, die dem Prozess – ähnlich wie ein »su«-Kommando – vorübergehend erweiterte oder verminderte Zugriffsrechte verschaffen. Ein wichtiger Unterschied ist, dass RSBAC die Benutzerkennung weiterhin korrekt protokolliert und damit eine zuverlässige Beweissicherung gewährleistet.

Aufgabentrennung

Die administrativen Aufgaben können im RC-Modell auf viele verschiedene Rollen verteilt sein; diese Aufteilung lässt sich sogar dauerhaft fixieren. Hierfür gibt es eine Reihe zusätzlicher Rolleneinstellungen und Zugriffsrechte. Trotzdem muss eine Rolle durch das Attribut »admin_role« vorerst als zentraler Sicherheitsadministrator dienen, denn nur dieses Attribut erlaubt die nachfolgend beschriebenen Einstellungen. Ist alles fertig konfiguriert, darf das Attribut zurückgesetzt oder die Rolle gelöscht werden – damit ist die Aufgabentrennung fest versiegelt.

Zunächst enthält die Rolle eines Teil-Admin eine Liste »admin_roles«. Sie bestimmt, welche anderen Rollen dieser Admin verwalten darf. In den meisten Fällen sind dazu zusätzlich weitere Privilegien erforderlich. Eine weitere Liste »assign_roles« enthält jene Rollen, die der Prozess zuweisen darf, und zwar als kompatible Rolle, als Default-Rolle für Benutzer, als initiale oder erzwungene Programmrolle oder als neue Rolle für einen laufenden Prozess. Es gelten folgende Einschränkungen:

- Kompatible Rollen: Die Ausgangsrolle muss in »admin_roles« enthalten sein, die kompatible (Ziel-)Rolle in »assign_roles«.
- Benutzerrolle: Die bisherige Rolle des Benutzers und die neue Rolle müssen in »assign_roles« enthalten sein. Das verhindert eine Rollenvergabe außerhalb des vorgesehenen Administrationsbereichs.
- Für Programm- und Prozessrollen: Die administrierende Rolle muss auf den Typ der Programmdatei oder des Prozesses das RSBAC-Standardrecht »MODIFY_ATTRIBUTE« haben.

Zusätzlich zu den RSBAC-Standardrechten definiert RC einige Spezialrechte auf Typen (siehe Tabelle 4). Die zusätzlichen Bedingungen beschränken den Wirkungskreis einer administrativen Rolle und fördern so die geschlossene Verwaltung von Benutzer- und Objektmengen als Arbeitsgruppe.

Alle Einstellungen zur Aufgabentrennung unterliegen – wie alle Rechtevergaben – einer optionalen Zeitbeschränkung. Dies ist stark von der korrekten Systemzeit abhängig, die deshalb über das SCD-Objekt Clock (siehe Tabelle 2) gut geschützt sein sollte.

Das RC-Modell empfiehlt sich immer dann, wenn Benutzer und Objekte sich gut zu Rollen und Typen zusammenfassen lassen und programmabhängige Rechte oder eine gute Aufgabentrennung benötigt werden. Wegen der guten Abstraktion und seiner Mächtigkeit spart es im laufenden Betrieb viel Arbeit und ist deshalb ganz klar das Lieblingsmodell des Autors. Unter [3] befindet sich eine umfangreiche formale Modellbeschreibung, allerdings auf Englisch.

ACL: Zugriffskontrolllisten

Zugriffskontrolllisten (ACL, Access Control Lists) legen an jedem Objekt individuell fest, wer dort welche Zugriffsrechte hat. Da dafür viel Arbeit und Speicher nötig sind, verwenden die meisten ACL-Modelle umfangreiche Vererbungsmechanismen. Das in RSBAC enthaltene ACL-Modell ist dem älterer Netware-Versionen recht ähnlich, alte Netware-Hasen werden sich also schnell wohlfühlen. Es gibt allerdings einige wichtige Unterschiede, zum Beispiel in der Gruppenverwaltung und der Unterstützung von RC-Rollen.

Das ACL-Modul kennt drei Arten von Subjekten: Benutzer, Gruppen und RC-Rollen. Gruppen sind hier nicht die Linux-Gruppen, die in einer einzelnen Datei konfiguriert werden, auf die nur Root Zugriff hat. Vielmehr gehört bei RSBAC jede ACL-Gruppe einem Benutzer, der sie selbst verwaltet. Nur dieser Besitzer kann die Einträge ändern, die Gruppe als global oder persönlich nutzbar einstellen oder sie an einen anderen Besitzer übertragen. In einer späteren Version sollen explizite ACLs dieses diskrete Besitzer-Verfahren ablösen.

Jeder ACL-Eintrag an einem Objekt enthält einen Subjekttyp, eine Subjekt-ID und eine Menge von Rechten. Verwendbare Rechte sind die üblichen RSBAC-Standardrechte und ein paar Spezialrechte für die Administration, die unten näher erklärt werden. Jedes Objekt besitzt auch eine Maske, die das Vererben von Rechten beschränkt. Welche Rechte ein Subjekt auf ein Objekt hat, ergibt sich nach folgendem Verfahren: Ist am Objekt ein ACL-Eintrag für dieses Subjekt vorhanden, dann gelten die dort eingetragenen Rechte. Andernfalls gelten die Rechte auf das übergeordnete Objekt, zum Beispiel das nächsthöhere Verzeichnis, abzüglich jener Rechte, die in der Vererbungsmaske des erbenden Objekts fehlen.

Als oberstes Objekt der Vererbungshierarchie gibt es für Dateisystemobjekte wie für alle anderen RSBAC-Objekttypen jeweils eine Default-ACL. Prozess-, Benutzer- und Interprozesskommunikations-Objekte (IPC) werden wegen ihrer kurzen Lebensdauer ausschließlich über ihre Default-ACLs kontrolliert.

Tabelle 4: Spezialrechte in RC

Recht	Beschreibung
ADMIN	Die Rolle darf diesen Typ administrieren, derzeit gibt es nur Umbenennen oder Löschen
ASSIGN	Die Rolle darf einem Objekt diesen Typ zuweisen, zusätzlich benötigt sie das Standardrecht »MODIFY_ATTRIBUTE« auf den bisherigen Objekttyp
ACCESS_CONTROL	Die Rolle darf anderen Rollen Standardrechte (aber keine Spezialrechte) auf diesen Typ zuweisen, zusätzlich muss die Zielrolle in der Menge der »admin_roles« enthalten sein
SUPERVISOR	Die Rolle darf anderen Rollen Spezialrechte auf diesen Typ geben, zusätzlich muss die Zielrolle in der Menge der »admin_roles« enthalten sein

Rechte erben und addieren

Die aktuellen Rechte eines Prozesses auf ein Objekt addieren sich nun aus den Rechten seines Benutzers, den Rechten aller Gruppen, in denen der Benutzer Mitglied ist, und den Rechten der aktuellen RC-Prozessrolle. Die Einzelrechte sind möglicherweise von übergeordneten Objekten geerbt.

Um zum Beispiel einen Verzeichnisbaum vor Schreibzugriffen zu schützen, reicht es aus, in der Vererbungsmaske dieses Verzeichnisses ausschließlich Leserechte freizugeben. Die Default-ACL der Dateisystemobjekte vergibt per Default volle Zugriffsrechte. Die Maske legt fest, was davon nach der Vererbung übrig bleibt. Anschließend kann ein Benutzer mit Administrationsrechten in einem Unterverzeichnis über gezielte

ACL-Einträge weitere Rechte an bestimmte Subjekte vergeben. Diese ACL-Einträge überdecken dort die gefiltert vererbten Rechte.

Das ACL-Modell ist nicht streng mandatorisch, es enthält über Spezialrechte diskrete Elemente. Andererseits werden auch diese explizit vergeben und von neu erzeugten Objekten lediglich geerbt. Alle ACL-Einträge haben optional eine begrenzte Lebensdauer, nach der RSBAC sie automatisch entfernt. Wie beim RC-Modell erwähnt ist dafür eine genaue Kontrolle der Systemzeit über das SCD-Objekt Clock nötig.

ACL administrieren

Die ACL-Administrationsrechte werden ebenfalls in den ACL-Einträgen verwaltet. Das Modell kennt drei Sonderrechte:

- Access Control: Mit diesem Recht darf das zugehörige Subjekt ACL-Einträge und Vererbungsmasken hinzufügen und entfernen sowie Standardrechte (aber keine Spezialrechte) in allen Einträgen setzen oder löschen.
- Forward: Dieses Recht erlaubt es dem Subjekt, seine eigenen Rechte an andere weiterzugeben. Es kann die Rechte anschließend nicht mehr widerrufen, ihnen aber eine begrenzte Lebensdauer geben.
- Supervisor: Das Supervisor-Recht enthält alle anderen Rechte, erlaubt das Verwalten der Spezialrechte und lässt sich nur unter bestimmten Bedingungen aus einer Vererbungsmaske löschen. In der Grundeinstellung hat Benutzer »400« dieses Recht in allen Default-ACLs und kann daher das Modell administrieren.

Die ACLs lassen sich mit einer Reihe von Kommandozeilentools administrieren, namentlich »acl_grant«, »acl_tlist«, »acl_mask« und »acl_rights«. Die Menüs »rsbac_acl_menu« (ACL-Einträge verwalten) und »rsbac_acl_group_menu« (Gruppen verwalten) sorgen für mehr Überblick und Bequemlichkeit.

Immer dann, wenn viele individuelle Rechte zu vergeben sind oder Benutzer ihre Dateien zumindest teilweise selbst schützen sollen, gibt es keine Alternative zum ACL-Modell. Sogar programmabhängige Rechte lassen sich mit Hilfe der RC-Programmrollen verwirklichen. Allerdings ist die Administration wegen der vielen verteilten Rechte wesentlich aufwändiger, der Überblick geht sehr viel leichter verloren als beim RC-Modell. Das recht häufige Problem mit vergessenen, nur auf Zeit gewährten Rechten ist aber mit der Lebenszeit-Option in den Griff zu kriegen.

Linux-Capabilities

Schon seit einiger Zeit kennt der Linux-Kernel Capabilities. Das sind im Prinzip aufgeteilte Sonderrechte, die in älteren Versionen nur dem Benutzer Root mit der User-ID »0« zustanden. Per Default geben auch die neuesten Kernel diesem Benutzer immer alle Linux-Capabilities, während alle anderen User leer ausgehen. Leider hat das Ganze zwei dicke Haken: Prozesse können zwar Capabilities freiwillig abgeben, es gibt aber keine Möglichkeit, dies zu erzwingen. Auch ist es unmöglich, andere Benutzer mit Capabilities auszustatten. Als kleine Hilfe zeigt Tabelle 5 eine Liste der derzeit verwendeten Capabilities.

Das RSBAC-Modul CAP schließt diese Lücke, indem es für jeden Benutzer und jedes Programm eine minimale und eine maximale Menge an Capabilities definiert. Jeder Benutzerwechsel und jeder Start eines neuen Programms passt die Capabilities eines Prozesses so an, dass

Tabelle 5: Posix-Capabilities

Name	Beschreibung
CAP_CHOWN	Besitzer eines Objekts ändern
CAP_DAC_OVERRIDE	Voller Zugriff im Dateisystem, unabhängig von vorhandenen Rechten
CAP_DAC_READ_SEARCH	Volle Leserechte im Dateisystem, unabhängig von vorhandenen Rechten
CAP_FOWNER	Erlaubt alle Objektoperationen, die sonst nur ihrem Besitzer vorbehalten sind, zum Beispiel die Rechteverwaltung
CAP_FSETID	Überstimmt die Beschränkungen für das Setzen der Set-UID- und Set-GID-Bits
CAP_KILL	Signale an beliebige Prozesse senden
CAP_SETGID	Der Prozess darf in beliebige Linux-Gruppen wechseln
CAP_SETUID	Der Prozess darf alle Benutzer-IDs wechseln, also reale, effektive und Dateisystem-IDs
CAP_SETPCAP	Eigene Capabilities auch in anderen Prozesse setzen oder beliebige Capabilities entfernen
CAP_LINUX_IMMUTABLE	Ändern der Immutable- und Append-Dateiattribute
CAP_NET_BIND_SERVICE	Binden an TCP- und UDP-Ports unter 1024
CAP_NET_BROADCAST	Broadcasts verschicken, Multicasts empfangen
CAP_NET_ADMIN	Netzwerkadministration
CAP_NET_RAW	Raw-Sockets benutzen, also beliebige Pakete im Netzwerk verschicken
CAP_IPC_LOCK	Shared Memory im Hauptspeicher halten, also das Auslagern (Swapping) verbieten
CAP_IPC_OWNER	Voller Zugriff auf Interprozesskommunikations-Objekte (IPC), unabhängig von deren Besitzer
CAP_SYS_MODULE	Kernelmodule laden und entfernen
CAP_SYS_RAWIO	Direkter Zugriff auf Hardware-Ports
CAP_SYS_CHROOT	Benutzen von »chroot()«
CAP_SYS_PTRACE	Beliebige Prozesse mit »ptrace()« untersuchen
CAP_SYS_PACCT	Prozess-Accounting verwalten
CAP_SYS_ADMIN	Administrationsaufgaben durchführen, die von den anderen Capabilities nicht erfasst werden
CAP_SYS_BOOT	System herunterfahren oder neu starten
CAP_SYS_NICE	Priorität von Prozessen erhöhen
CAP_SYS_RESOURCE	Ressourcen-Schranken überwinden, etwa Quota-Grenzen oder reservierten Plattenplatz
CAP_SYS_TIME	System- und Hardwarezeit setzen
CAP_SYS_TTY_CONFIG	TTY-Geräte (Konsolen) konfigurieren
CAP_MKNOD	Geräte-Spezialdateien (Device-Files) erzeugen
CAP_LEASE	Lease auf eine Datei setzen, sie also exklusiv nutzen

sie in den vordefinierten Schranken bleiben. Auch in diesem Modul gibt es eine Benutzerrolle, die auf normaler Benutzer, Systemadministrator oder Sicherheitsadministrator gesetzt werden kann. Nur Sicherheitsadmins dürfen CAP-Einstellungen ändern, Sysadmins dürfen sie immerhin noch lesen.

Als Bonbon fügt die neue RSBAC-Version 1.2.2 diesem Modul eine weitere Funktion hinzu: Prozesseigenschaften vor anderen Benutzern verstecken. Unter Prozesseigenschaften sind hier alle Daten aus dem zugehörigen Proc-Verzeichnis zu verstehen, zum Beispiel Parameter, Umgebungsvariablen und Status.

Dazu gibt es das zusätzliche Prozessattribut »cap_process_hiding« mit den Werten »none« (voll sichtbar), »hide_from_other_users« (nur für Prozesse dieses Benutzers oder für System- und Sicherheitsadministratoren sichtbar) und »full« (nur für den Prozess selbst und für Sicherheitsadmins sichtbar). Ein neuer Kernelparameter »rsbac_cap_process_hiding« setzt den Standardwert für alle Prozesse von »none« auf »hide_from_other_users« um, sodass normale Benutzer nur noch die Eigenschaften ihrer eigenen Prozesse sehen.

Das CAP-Modul ist immer dann praktisch, wenn man auf einfache Weise und im Rahmen des Standard-Linux-Modells die Administrationsrechte von Benutzern und Programmen ändern will. So erlaubt es insbesondere, Serverprogramme als normaler Benutzer zu starten oder – falls sie nur von Root gestartet werden dürfen – in ihren Rechten zu beschränken.

Wohl am häufigsten zum Einsatz kommt »CAP_DAC_READ_SEARCH« für die Sicherheitsadmins aller Modelle. Sie dürfen so alle Verzeichnisse lesen, trotz fehlender Linux-Rechte – eine wichtige Voraussetzung für die Administration.

Tabelle 6: Ressourcen-Schranken

Name	Beschreibung
CPU	CPU-Zeit pro Prozess in Sekunden
FSIZE	Maximale Dateigröße
DATA	Maximalgröße des Prozess-Datensegments
STACK	Maximale Stack-Größe
CORE	Maximalgröße eines Core-Dumps beim Programmabsturz
RSS	Maximale Größe des residenten Prozessspeichers
NPROC	Höchstzahl an Prozessen für diesen Benutzer
NOFILE	Maximale Anzahl geöffneter Dateien pro Prozess
MEMLOCK	Maximale Speichermenge, die im Hauptspeicher festgehalten, also nicht ausgelagert wird (Swapping)
AS	Obergrenze des Adressbereichs, also des virtuellen Speichers
LOCKS	Höchstzahl an blockierten Dateien

Linux-Ressourcen setzen

Ähnlich wie die Posix-Capabilities lassen sich auch die seit längerem bekannten Ressourcen-Kontrollen mit einem Standardkernel nur begrenzt administrieren. Das RES-Modul verwaltet für alle Benutzer und Programme untere und obere Schranken, die analog zu den Capabilities umgesetzt sind. Auch hier gibt es Benutzerrollen. Tabelle 6 zeigt die in der 2.4er Kernelserie unterstützten Ressourcen-Parameter.

Zu guter Letzt

Obwohl die vorgestellten Entscheidungsmodule viele Bedürfnisse abdecken, gibt es häufig neue Ideen und Wünsche. Ich bitte daher alle Leser um Anregungen und Rückmeldungen – die einzelnen Modelle sollen noch reifen. *(fjl)*

Infos

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

[2] Amon Ott, „Sicherheits-Architektur – Die Architektur des Linux-Sicherheitssystems Rule Set Based Access Control (RSBAC)“: Linux Magazin 01/03, S. 48

[3] Amon Ott, „The Role Compatibility Security Model“: [http://www.rsbac.org/rc-nordsec2002/]

[4] Amon Ott, „Root ist nicht alles – Mehr Sicherheit für Linux-Server mit RSBAC“: iX 08/02, S. 99

Der Autor

Amon Ott ist selbstständiger Diplom-Informatiker und der Autor des RSBAC-Systems. Seine Brötchen verdient er überwiegend mit Auftragsentwicklungen und Linux-Firewalls, bevorzugt natürlich mit RSBAC. Nebenher entwickelt er ein neues RSBAC-Serverprodukt und schreibt an seiner Dissertation zu diesem Thema, die hoffentlich bald fertig wird.