open source library

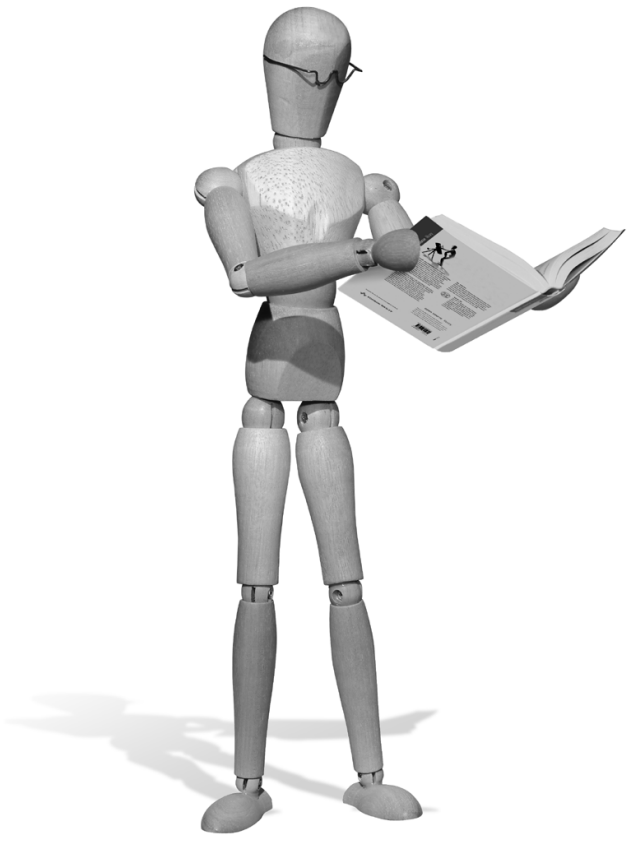
Evi Nemeth
Garth Snyder
Trent R. Hein

Linux-Administrations-handbuch

ADDISON-WESLEY

Teil A

Grundlegende Verwaltung



3 Die Macht von root

Jede Datei und jeder Prozess auf einem Linux-System gehören einem bestimmten Benutzerkonto. Andere Benutzer haben ohne die Erlaubnis des Besitzers keinen Zugriff auf diese Objekte, daher schützt diese Konvention den Benutzer sowohl vor absichtlichen als auch vor unbeabsichtigten Missetaten anderer.

Systemdateien und Prozesse gehören üblicherweise einem fiktiven Benutzer namens `root`, auch als der Superuser bekannt. Wie bei jedem Konto, ist das Eigentum von `root` vor der Einmischung anderer Benutzer geschützt. Um administrative Änderungen vorzunehmen, müssen Sie eine der in diesem Kapitel beschriebenen Verfahren für den Zugang zum `root`-Konto einsetzen.

Das `root`-Konto hat einige »magische« Eigenschaften. `root` kann als Besitzer jeder Datei und jedes Prozesses handeln. `root` kann außerdem mehrere besondere Tätigkeiten ausführen, die für andere Benutzer verboten sind. Dieses Konto ist sowohl mächtig als auch, in sorglosen oder böswilligen Händen, möglicherweise gefährlich.

Dieses Kapitel führt Sie in die Grundlagen des Superuser-Zugangs für Administratoren ein. Kapitel 20, »Sicherheit«, beschreibt, wie Sie den unerwünschten Superuser-Zugriff durch andere verhindern. Kapitel 30, »Management und Geschäftspolitik«, behandelt die einschlägigen politischen und administrativen Gesichtspunkte.

3.1 Besitz von Dateien und Prozessen

Jede Datei hat einen Besitzer und einen »Gruppenbesitzer«. Der Besitzer hat ein besonderes Privileg, das er mit niemandem im System teilt: die Fähigkeit, die Berechtigungen der Datei zu verändern. Insbesondere kann er die Berechtigungen einer Datei so weit einschränken, dass kein anderer Zugriff auf sie hat.[1] In Kapitel 5, »Das Dateisystem«, erfahren Sie mehr über Dateiberechtigungen.

1 *Tatsächlich können die Berechtigungen so restriktiv gesetzt werden, dass nicht einmal der Besitzer der Datei auf sie zugreifen kann.*

Obwohl der Besitzer einer Datei stets eine Einzelperson ist, können viele Personen Gruppenbesitzer der Datei sein, sofern sie alle Mitglieder in einer einzelnen Linux-Gruppe sind. Gruppen werden herkömmlicherweise in der Datei `/etc/group` definiert, doch heute ist es gebräuchlicher, die Gruppeninformationen auf einem NIS- oder LDAP-Server im Netzwerk zu speichern. Einzelheiten dazu finden Sie in Kapitel 17, »Systemdateien zentral verwalten«.

Tipp

Weitere Informationen über Gruppen erhalten Sie in Abschnitt 6.1.4.

Der Besitzer der Datei gibt an, was die Gruppenbesitzer mit ihr machen können. Dieses Verfahren ermöglicht die gemeinsame Nutzung von Dateien durch Mitarbeiter desselben Projekts. Wir verwenden z. B. eine Gruppe, um den Zugriff auf die Quelldateien für die Website *www.admin.com* zu steuern.

Beide Zugehörigkeiten einer Datei können mit dem Befehl `ls -l` *`dateiname`* bestimmt werden. Betrachten Sie das folgende Beispiel:

```
$ ls -l /staff/scott/todo
-rw-------    1    scott    staff    1258    Jun 4 18:15    /staff/scott/todo
```

Die Datei gehört dem Benutzer `scott` und der Gruppe `staff`.

Linux verwaltet Benutzer und Gruppen jedoch in Form von Zahlen und nicht als Namen. Im einfachsten Fall werden Identifizierungsnummern (kurz UIDs genannt) in der Datei `/etc/passwd` auf Benutzernamen und Gruppenidentifizierungsnummern (GIDs) in der Datei `/etc/group` auf Gruppennamen abgebildet. Die Namen, die zu den UIDs und GIDs gehören, dienen nur der Bequemlichkeit für die menschlichen Benutzer des Systems. Wenn Befehle wie `ls` die Angaben zu den Besitzern in einem für Menschen lesbaren Format ausgeben, müssen sie jeden Namen in der entsprechenden Datei oder Datenbank nachschlagen.

Der Besitzer eines Prozesses kann dem Prozess Signale schicken (siehe Abschnitt 4.3) und außerdem seine Priorität herabsetzen. Prozesse sind tatsächlich mit mindestens sieben Identitäten verknüpft: einer realen, effektiven und gespeicherten UID, einer realen, effektiven und gespeicherten GID sowie unter Linux einer »Dateisystem-UID«, die nur zur Bestimmung der Dateizugriffsberechtigungen verwendet wird. Grob gesprochen dienen die realen IDs der Buchführung und die effektiven der Bestimmung der Zugriffsberechtigungen. In der Regel sind sie identisch.

Gespeicherte IDs haben keine unmittelbare Auswirkung. Sie ermöglichen Programmen, eine inaktive ID für den späteren Gebrauch zu »parken«, und erleichtern damit die sparsame Verwendung erweiterter Privilegien. Die Dateisystem-ID wird im Allgemeinen als Implementierungsdetail von NFS beschrieben und ist in der Regel dieselbe wie die effektive UID.

Obwohl ein Prozess normalerweise seine Besitzerrechte nicht ändern kann, gibt es einen besonderen Fall, in dem die effektiven Benutzer- und Gruppen-IDs abgewandelt werden können. Wenn der Kernel eine Datei ausführt, deren `setuid`- oder `setgid`-Berechtigungsbit gesetzt ist, ändert er die effektive UID oder GID des sich ergebenden Prozesses auf die UID oder GID der Datei, die das Programm enthält, statt auf die UID oder GID des Benutzers, der das Programm ausführt. Die Berechtigungen des Benutzers werden also nur für das Ausführen dieses bestimmten Befehls heraufgestuft.

Tipp

In Abschnitt 5.1 finden Sie weitere Informationen über Berechtigungsbits.

Der Einsatz von `setuid` unter Linux erlaubt Programmen, die von normalen Benutzern ausgeführt werden, das `root`-Konto auf eingeschränkte und streng kontrollierte Weise zu nutzen. Zum Beispiel ist der Befehl `passwd`, den Benutzer zum Ändern ihres Anmeldepassworts ausführen, ein `setuid`-Programm. Er ändert die Datei `/etc/shadow` (oder `/etc/passwd`) auf wohldefinierte Art und Weise und beendet sich dann. Natürlich hat auch dieses begrenzte Verfahren die Möglichkeit eines Missbrauchs, sodass `passwd` vom Benutzer fordert, sein aktuelles Passwort einzugeben, bevor die geforderte Änderung durchgeführt wird.

3.2 Der Superuser

Das kennzeichnende Merkmal des `root`-Kontos ist seine UID von 0. Linux hindert Sie nicht daran, den Benutzernamen dieses Kontos zu ändern oder zusätzliche Konten mit der UID 0 zu erstellen, doch das sind beides keine guten Ideen. Solche Änderungen neigen dazu, unvorhergesehene Lücken in der Systemsicherheit zu öffnen. Sie sorgen auch für Verwirrung, wenn andere Personen sich mit der seltsamen Art herumschlagen müssen, in der Sie Ihr System konfiguriert haben.

Das traditionelle UNIX erlaubt dem Superuser (d.h., jedem Prozess, dessen effektive UID gleich 0 ist), jede gültige Operation an jeder Datei und jedem Prozess auszufüh-

ren.[2] Darüber hinaus können einige Systemaufrufe (Anfragen an den Kernel) nur vom Superuser ausgeführt werden. Im Folgenden sehen Sie einige Beispiele derart eingeschränkter Operationen:

- Das Wurzelverzeichnis eines Prozesses mit `chroot` ändern
- Gerätedateien anlegen
- Die Systemuhr stellen
- Nutzungsbeschränkungen für Ressourcen und Prozessprioritäten erhöhen[3]
- Den Hostnamen des Systems setzen
- Netzwerkkarten konfigurieren
- Privilegierte Netzwerkports (Ports unterhalb 1024) öffnen
- Das System herunterfahren

Ein Beispiel für die Macht des Superusers ist die Fähigkeit, die UID und GID eines Prozesses zu ändern, der `root` gehört. Das Programm `login` und seine grafischen Gegenstücke sind ein solcher Fall. Der Prozess, der Sie beim Anmelden am System nach Ihrem Passwort fragt, wird als `root` ausgeführt. Wenn das von Ihnen eingegebene Passwort und der Benutzername anerkannt wurden, ändert das Loginprogramm seine UID und GID in Ihre UID und GID und fährt Ihre Benutzerumgebung hoch. Sobald ein `root`-Prozess seinen Besitzer geändert hat und ein normaler Benutzerprozess geworden ist, kann er seinen vorherigen privilegierten Zustand nicht zurück erlangen.

Linux-Systeme sind theoretisch dazu in der Lage, die Privilegien des `root`-Kontos gemäß dem POSIX-Standard für »Fähigkeiten« (Capabilitys) zu unterteilen. Aus verschiedenen Gründen, darunter Problemen mit der aktuellen Implementierung, ist diese Eigenschaft für Systemadministratoren nicht so hilfreich oder von Bedeutung, wie es zunächst scheinen mag. Weitere Kommentare zu POSIX-Capabilitys finden Sie in Abschnitt 20.6.

3.3 Ein root-Passwort auswählen

Das `root`-Passwort sollte mindestens acht Zeichen lang sein, denn Passwörter mit sieben Zeichen sind wesentlich einfacher zu knacken. Auf Systemen, die DES-Passwörter verwenden, nützt es nichts, ein längeres Passwort zu verwenden, da nur die ersten acht Zeichen von Bedeutung sind. Im Abschnitt 6.1.2 finden Sie Informationen darüber, wie Sie MD5-Passwörter aktivieren, die länger als acht Zeichen sein können.

2 *Das Wort »gültig« ist hier wichtig. Manche Tätigkeiten (z. B. eine Datei auszuführen, deren Ausführungsbit nicht gesetzt ist) sind sogar dem Superuser verboten.*

3 *Ab der Kernelversion 2.16.12 ermöglicht eine neue Ressourcenbeschränkung Nicht-Superusern das Erhöhen von Prozessprioritäten, wenn der Systemadministrator das erlaubt.*

Es ist wichtig, das `root`-Passwort so zu wählen, dass es nicht einfach erraten oder durch Versuche herausgefunden werden kann. In der Theorie bestehen die sichersten Passwörter aus einer zufälligen Folge von Buchstaben, Satzzeichen und Ziffern. Doch da sich ein solches Passwort schlecht merken und in der Regel schwer eingeben lässt, ist es möglicherweise nicht optimal sicher, wenn Administratoren es aufschreiben oder langsam tippen.

Bis vor Kurzem war ein Passwort aus zwei zufällig gewählten, durch ein Satzzeichen getrennten Wörtern ein recht guter Kompromiss zwischen Sicherheit und Merkbarkeit, doch leider können solche Passwörter jetzt ziemlich schnell geknackt werden, sodass wir dieses Verfahren ausdrücklich ablehnen.

Tipp

In Abschnitt 20.10.3 gibt es weitere Informationen zum Knacken von Passwörtern.

Wir empfehlen, dass Sie ein `root`-Passwort bilden, indem Sie einen Satz aus »schockierendem Blödsinn« zusammenfassen, wie es von Grady Ward in einer früheren Version der FAQ zu PGP Passphrase definiert wurde:

> *»Schockierender Blödsinn« bedeutet, eine kurze Wortfolge oder einen Satz aufzustellen, der sowohl sinnlos als auch im kulturellen Umfeld des Benutzers schockierend ist. Das bedeutet, er enthält äußerst obszöne, rassistische, unmögliche oder anderweitig extreme Aneinanderreihungen von Ideen. Diese Methode ist erlaubt, weil die Passphrase naturgemäß niemals jemandem gezeigt wird, dessen Gefühle verletzt werden könnten.*
>
> *Es ist äußerst unwahrscheinlich, dass derselbe schockierende Blödsinn irgendwo anders nochmal auftaucht, da er keine Tatsachen beschreibt, die jemand anders zufällig ebenfalls beschreibt. Aufgrund der starken emotionalen Wirkung kann der Urheber seinen Satz auch nicht so leicht wieder vergessen. Ein eher harmloses Beispiel für solchen schockierenden Blödsinn könnte etwa lauten: »Mollusken fressen meine davongaloppierenden Geschlechtsteile.« Der Leser kann sich ohne Schwierigkeiten eigene weit schockierendere oder unterhaltsamere Beispiele ausdenken.«*

Sie können einen solchen Satz zu einem Passwort verkürzen, indem Sie nur den ersten Buchstaben eines jeden Worts verwenden oder eine ähnliche Transformation durchführen. Die Passwortsicherheit wird gewaltig erhöht, wenn Sie auch Ziffern, Satzzeichen oder Großbuchstaben verwenden.

Wann sollten Sie das `root`-Passwort ändern?

- Mindestens ca. alle drei Monate
- Jedes Mal, wenn jemand, der das Passwort kennt, Ihr Unternehmen verlässt

- Immer wenn Sie der Meinung sind, die Sicherheit könnte gefährdet sein
- An einem Tag, an dem Sie nicht vorhaben, abends so lange zu feiern, dass Sie das Passwort am nächsten Morgen vergessen haben
- Nicht unmittelbar vor Ihrem Urlaub

3.4 Als root arbeiten

Da `root` nur ein weiterer Benutzer ist, können Sie sich direkt mit dem `root`-Konto anmelden. Das erweist sich jedoch als schlechte Idee. Zunächst einmal wird bei dieser Vorgehensweise nicht aufgezeichnet, welche Operationen als `root` ausgeführt wurden. Das ist schlimm genug, wenn Sie feststellen, dass Sie in der letzten Nacht um 3:00 Uhr etwas kaputt gemacht haben und sich nicht mehr daran erinnern können, was Sie geändert haben; es ist noch schlimmer, wenn es einen unerlaubten Zugriff gegeben hat und Sie versuchen, herauszufinden, was der Eindringling mit Ihrem System angestellt hat. Ein weiterer Nachteil besteht darin, dass es bei einer direkten Anmeldung mit `root` keine Aufzeichnung darüber gibt, wer die Arbeit tatsächlich ausgeführt hat. Wenn mehrere Personen Zugang zum `root`-Konto haben, können Sie nicht nachvollziehen, wer es wann benutzt hat.

Aus diesen Gründen bieten die meisten Systeme die Möglichkeit, `root`-Logins auf Terminals und über das Netzwerk zu deaktivieren – überall, außer auf der Systemkonsole.[4] Wir empfehlen Ihnen, dass Sie diese Möglichkeiten nutzen. Im Abschnitt 20.9.2 können Sie nachlesen, welche Datei Sie auf Ihrem System dazu bearbeiten müssen.

3.4.1 su: Benutzeridentitäten ersetzen

Ein etwas besserer Zugriff auf das `root`-Konto ist die Verwendung des Befehls `su`. Ohne Argumente aufgerufen, erwartet `su` die Eingabe des `root`-Passworts und startet dann eine `root`-Shell. Die Privilegien dieser Shell bleiben bestehen, bis sie beendet wird (über Strg-D oder den Befehl `exit`). `su` protokolliert nicht die als `root` ausgeführten Befehle, erstellt aber einen Protokolleintrag, der angibt, wer `root` wurde und wann.

Der Befehl `su` kann auch andere Identitäten als `root` setzen. Manchmal besteht die einzige Möglichkeit, das Problem eines Benutzers zu reproduzieren oder zu debuggen, darin, `su` für sein Konto auszuführen, sodass Sie die Umgebung, in der das Problem auftritt, nachstellen können.

4 *Ubuntu geht sogar noch weiter. Standardmäßig hat das System kein gültiges* `root`*-Kennwort und fordert den Einsatz von* `sudo`*, was weiter hinten in diesem Abschnitt genauer erläutert wird.*

Wenn Sie das Passwort eines anderen Benutzers kennen, können Sie direkt auf dessen Konto zugreifen, indem Sie `su` *`benutzername`* aufrufen. Wie bei `su` für `root`, werden Sie nach dem Kennwort für *`benutzername`* gefragt. Sie können auch zunächst `su` für `root` ausführen und dann `su` für ein anderes Konto; `root` kann `su` für jedes Konto ausführen, ohne ein Passwort anzugeben.

Sie sollten sich angewöhnen, den vollständigen Pfadnamen des Befehls `su` einzugeben (d. h. `/bin/su`), anstatt darauf zu vertrauen, dass die Shell ihn für Sie findet. Das schützt Sie vor Programmen mit dem Namen `su`, die in Ihren Suchpfad geraten sein könnten, um Passwörter auszuspionieren.[5]

3.4.2 sudo: su mit Einschränkung

Da die Privilegien des Superuser-Kontos nicht aufgeteilt werden können (zumindest nicht willkürlich), ist es schwierig, jemandem die Erlaubnis für eine Aufgabe zu geben (z. B. Backups durchzuführen), ohne ihm einen Freifahrtschein für das System auszustellen. Wenn das `root`-Konto von mehreren Administratoren verwendet wird, haben Sie nur eine vage Vorstellung davon, wer es nutzt und was er getan hat.

Die am weitesten verbreitete Lösung für dieses Problem ist ein Programm namens `sudo`, das zurzeit von Todd Miller gewartet wird. Es ist standardmäßig in allen unseren Beispieldistributionen enthalten, aber auch unter *www.courtesan.com* als Quellcode erhältlich.

`sudo` übernimmt einen Befehl als Argument, der als `root` (oder als anderer eingeschränkter Benutzer) auszuführen ist. Es liest die Datei `/etc/sudoers` ein, die auflistet, welche Benutzer zur Ausführung von `sudo` berechtigt sind und welche Befehle sie auf den einzelnen Rechnern ausführen dürfen. Wenn der gewünschte Befehl zulässig ist, fragt `sudo` den Benutzer nach *seinem eigenen* Passwort und führt den Befehl aus.

Während eines fünfminütigen Zeitraums (konfigurierbar) ohne weitere `sudo`-Aktivitäten kann der »Sudoer« zusätzliche `sudo`-Befehle ausführen, ohne sein Passwort erneut eingeben zu müssen. Diese Zeitbeschränkung dient als vernünftiger Schutz vor Benutzern mit `sudo`-Berechtigung, die ihr Terminal unbeaufsichtigt lassen.

Der Befehl `sudo` protokolliert die ausgeführten Befehle, die Rechner, auf denen sie ausgeführt wurden, wer sie angefordert hat, das Verzeichnis, aus denen sie aufgerufen wurden, sowie die zugehörige Uhrzeit. Diese Angaben können von `syslog` oder in einer Datei Ihrer Wahl aufgezeichnet werden. Wir empfehlen die Verwendung von `syslog`, um die Einträge an einen sicheren zentralen Rechner weiterzuleiten.

5 *Aus demselben Grund empfehlen wir dringend, »`.`« (das aktuelle Verzeichnis) nicht in den Suchpfad der Shell aufzunehmen. Diese Konfiguration ist zwar bequem, erleichtert es aber, aus Versehen »besondere« Versionen von Systembefehlen auszuführen, die ein Benutzer oder Eindringling als Falle hinterlegt hat. Für* `root` *gilt dieser Rat natürlich doppelt.*

Ein Protokolleintrag für den Benutzer `randy`, der `sudo /bin/cat /etc/sudoers` ausführt, kann wie folgt aussehen:

```
Dec 7 10:57:19 tigger sudo: randy: TTY=ttyp0 ; PWD=/tigger/users/randy; USER=root ;
COMMAND=/bin/cat /etc/sudoers
```

Die Datei `sudoers` ist so aufgebaut, dass eine einzige Version gleichzeitig auf vielen verschiedenen Rechnern verwendet werden kann. Nachfolgend sehen Sie ein typisches Beispiel:

```
# Define aliases for machines in CS & Physics departments
Host_Alias    CS = tigger, anchor, piper, moet, sigi
Host_Alias    PHYSICS = eprince, pprince, icarus
# Define collections of commands
Cmnd_Alias    DUMP = /sbin/dump, /sbin/restore
Cmnd_Alias    PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias    SHELLS = /bin/sh, /bin/tcsh, /bin/bash, /bin/ash, /bin/bsh
# Permissions
mark, ed    PHYSICS = ALL
herb    CS = /usr/sbin/tcpdump : PHYSICS = (operator) DUMP
lynda    ALL = (ALL) ALL, !SHELLS
%wheel    ALL, !PHYSICS = NOPASSWD: PRINTING
```

Die ersten fünf Zeilen ohne Kommentarzeichen geben Gruppen von Rechnern und Befehlen an, auf die weiter unten in den Berechtigungsangaben verwiesen wird. Diese Listen könnten wortgetreu in den Spezifikationen enthalten sein, doch die Verwendung von Aliasen erleichtert die Lesbarkeit und Verständlichkeit der Datei `sudoers` sowie deren spätere Aktualisierung. Es ist auch möglich, Aliase für Gruppen von Benutzern zu erstellen, unter denen Befehle ausgeführt werden dürfen.

Jede Zeile, die Berechtigungen beschreibt, enthält die folgenden Angaben:

- Die Benutzer, auf die die Zeile zutrifft
- Die Rechner, die diese Zeile beachten sollten
- Die Befehle, die die angegebenen Benutzer ausführen können
- Die Benutzer, unter denen die Befehle ausgeführt werden können

Die erste Zeile im Abschnitt `Permissions` aus dem Beispiel betrifft die Benutzer `mark` und `ed` auf den Rechnern der Gruppe `PHYSICS` (`eprince`, `pprince` und `icarus`). Der eingebaute Befehlsalias `ALL` erlaubt ihnen, alle Befehle auszuführen. Da keine Benutzerliste in Klammern angegeben ist, führt `sudo` nur Befehle als `root` aus.

Die zweite Zeile dieses Abschnitts erlaubt dem User `herb`, den Befehl `tcpdump` auf den Rechnern der Gruppe `CS` und die Befehle der Gruppe `DUMP` auf Rechnern der Gruppe `PHYSICS` auszuführen. Die `DUMP`-Befehle können jedoch nur als `operator` ausgeführt werden, nicht als `root`. Die Befehlszeile, die `herb` eingeben muss, lautet also wie folgt:

```
$ sudo -u operator /sbin/dump 0u /dev/hda2
```

Der Benutzer `lynda` kann Befehle als jeder Benutzer auf jedem Rechner ausführen, mit der Ausnahme einiger üblicher Shells. Bedeutet das, dass `lynda` tatsächlich keine `root`-Shell erhalten kann? Natürlich nicht:

```
$ cp -p /bin/bash /tmp/bash
$ sudo /tmp/bash
```

Allgemein gesagt ist jeder Versuch, »alle Befehle außer ...« zu erlauben, zum Scheitern verurteilt, zumindest im technischen Sinne. Es kann sich jedoch trotzdem lohnen, die Datei auf diese Art einzurichten, als Erinnerung daran, dass Shells missbilligt werden. Somit können sie vor dem gelegentlichen Einsatz von Shells abschrecken.

Die letzte Zeile erlaubt Benutzern der Gruppe `wheel`, die Befehle `lpc` und `lprm` auf allen Rechnern mit Ausnahme von `eprince`, `pprince` und `icarus` als `root` auszuführen. Darüber hinaus ist zum Ausführen der Befehle kein Kennwort erforderlich.

Beachten Sie, dass Befehle in `/etc/sudoers` mit vollem Pfadnamen angegeben sind, um die Benutzer davon abzuhalten, ihre eigenen Programme und Skripte als `root` auszuführen. Obwohl hier keine Beispiele dafür vorgestellt worden sind, ist es auch möglich, die Argumente anzugeben, die für die einzelnen Befehle erlaubt sind. Diese einfache Konfigurationsdatei zeigt bei Weitem nicht alle Möglichkeiten an, die die Datei `sudoers` bietet.

Nutzen Sie zum Ändern der Datei `/etc/sudoers` den Befehl `visudo`, der sicherstellt, dass kein anderer die Datei gerade bearbeitet, einen Editor aufruft und dann vor der Installation die Syntax der geänderten Datei verifiziert. Besonders der letzte Schritt ist wichtig, da eine ungültige `sudoers`-Datei Sie daran hindern könnte, `sudo` erneut aufzurufen, um den Fehler zu beheben.

Die Verwendung von `sudo` bietet folgende Vorteile:

- Die Rechenschaftslegung wird aufgrund der Befehlsprotokollierung stark vereinfacht.
- Operatoren können Aufgaben ohne unbegrenzte `root`-Berechtigungen durchführen.
- Das Wissen um das tatsächliche `root`-Passwort kann einem oder zwei Mitarbeitern vorbehalten sein.
- Es geht schneller, mit `sudo` einen einzelnen Befehl auszuführen, als mit `su` den Benutzer zu wechseln oder sich als `root` anzumelden.
- Privilegien können widerrufen werden, ohne das `root`-Passwort ändern zu müssen.
- Es gibt eine verbindliche Liste aller Benutzer mit `root`-Privilegien.
- Die Wahrscheinlichkeit für unbeaufsichtigt gelassene `root`-Shells sinkt.
- Der Zugriff für ein ganzes Netzwerk kann über eine einzige Datei gesteuert werden.

Das Verfahren birgt auch einige Nachteile. Der größte liegt darin, dass jede Verletzung der Sicherheit des persönlichen Kontos einer für `sudo` berechtigten Person auch die Sicherheit des `root`-Kontos selbst betrifft. Gegen diese Bedrohung können Sie nicht viel mehr machen, als Ihre Mitarbeiter zu warnen, ihr persönliches Konto so zu schützen, als wäre es das `root`-Konto. Sie können auch regelmäßig John the Ripper mit den Passwörtern der `sudo`-Benutzer ausführen, um sicherzustellen, dass sie gut gewählt sind.

Tipp

In Abschnitt 20.10.3 finden Sie weitere Informationen über John the Ripper.

Die Befehlsprotokollierung von `sudo` kann durch Tricks wie Escape-Zeichen der Shell aus einem erlaubten Programm heraus oder durch `sudo sh` und `sudo su` untergraben werden, sofern Sie das zulassen.

3.5 Andere Pseudobenutzer

`root` ist der einzige Benutzer, der aus der Sicht des Kernels einen besonderen Status hat, doch es sind noch mehrere andere Pseudobenutzer im System definiert. Es ist üblich, das verschlüsselte Passwortfeld dieser besonderen Benutzer in `/etc/passwd` durch einen Stern zu ersetzen, sodass sich niemand mit diesen Konten anmelden kann.

3.5.1 bin: Veralteter Besitzer von Systemkommandos

Auf einigen älteren UNIX-Systemen gehörten dem Benutzer `bin` die Verzeichnisse, die die Systembefehle enthielten, sowie auch die meisten dieser Befehle. Heute wird dieses Konto oft als überflüssig angesehen (oder vielleicht sogar als ein wenig unsicher), deshalb verwenden moderne Systeme (einschließlich Linux) im Allgemeinen einfach das `root`-Konto. Andererseits kann das `bin`-Konto, da es nun zum »Standard« gehört, nicht einfach entfernt werden.

3.5.2 daemon: Besitzer nicht privilegierter Systemsoftware

Dateien und Prozesse, die Teil des Betriebssystems sind, aber nicht `root` gehören müssen, werden manchmal an `daemon` gegeben. Die Idee dahinter war, die mit `root` verbundenen Sicherheitsrisiken zu vermeiden. Aus ähnlichen Gründen gibt es eine Gruppe namens `daemon`. Die meisten Linux-Distributionen nutzen das `daemon`-Konto jedoch ebenso wenig wie das `bin`-Konto.

3.5.3 nobody: Der generische NFS-Benutzer

Das Network File System (NFS) nutzt das Konto `nobody`, um `root`-Benutzer auf anderen Systemen zwecks gemeinsamer Dateinutzung darzustellen. Um entfernte `root`-Benutzer ihrer Macht zu berauben, muss die entfernte UID 0 auf etwas anderes als die lokale UID 0 abgebildet werden. Das Konto `nobody` handelt als allgemeines Alter Ego für diese entfernten `root`s.

Tipp

In Abschnitt 16.1.8 finden Sie weitere Informationen über das Konto `nobody`.

Da das `nobody`-Konto einen allgemeinen Benutzer mit relativ wenig Berechtigungen darstellen soll, sollten ihm keine Dateien gehören. Andernfalls könnten entfernte `root`s die Kontrolle darüber übernehmen. `nobody` sollten keine Dateien gehören!

Traditionell wurde für `nobody` die UID -1 oder -2 verwendet, und der Linux-Kernel nutzt standardmäßig immer noch die UID 65534 (das 16-Bit-Zweierkomplement -2). Einige Distributionen geben `nobody` eine niedrige User-ID (z. B. verwenden Red Hat und Fedora die Zahl 99), was sinnvoller ist, da User-IDs jetzt 32 Bit groß sind. Der einzige Haken ist, dass `exportfs` die Datei `passwd` nicht zu beachten scheint, sodass Sie ihm mit der Option `anonuid` mitteilen müssen, eine andere UID für `nobody` zu verwenden.

3.6 Übungen

☆ 1. Verwenden Sie den Befehl `find` mit der Option `-perm`, um auf Ihrem System fünf `setuid`-Dateien zu finden. Erläutern Sie zu jedem Befehl, warum das `setuid`-Verfahren für die einwandfreie Funktion des Befehls erforderlich ist.

☆ 2. Erstellen Sie drei Passphrasen nach dem Prinzip »schockierender Blödsinn«, aber behalten Sie sie für sich. Führen Sie für alle drei Passphrasen den Befehl `md5sum` aus. Warum ist es sicher, anderen die MD5-Ergebnisse mitzuteilen?

☆ 3. Geben Sie eine Abfolge von Befehlen an, die den Passworteintrag eines Benutzers ändern, und zeigen Sie, wie Sie Ihre Spuren verwischen. Nehmen Sie an, dass Sie lediglich `sudo`-Berechtigung haben (alle Befehle außer Shells und `su` sind erlaubt).

☆ 4. Erstellen Sie zwei Einträge für die Konfigurationsdatei `sudoers`:

   a. Einen Eintrag, der den Benutzern `matt`, `adam` und `drew` die Bedienung und das Entsperren des Druckers sowie den Neustart der Druck-Daemons auf dem Rechner `printserver` ermöglicht.

   b. Einen Eintrag, der den Benutzern `drew`, `smithgr` und `jimlane` das Abbrechen von Jobs und den Neustart von Rechnern im Studentenlabor erlaubt.

☆ 5. Konfigurieren Sie `sudo` so, dass Sie im Falle der unrechtmäßigen Verwendung per E-Mail benachrichtigt werden. Verwenden Sie den Befehl, um die `sudo`-Einträge der vorangegangenen Frage mit lokalen Benutzer- und Rechnernamen zu testen. Überprüfen Sie, dass `sudo` korrekt in `syslog` protokolliert. Schauen Sie sich die von Ihren Tests erstellten Syslog-Einträge an. (Erfordert `root`-Zugriff; Sie müssen wahrscheinlich auch `/etc/syslog.conf` anpassen.)