- wird ein dennoch erfolgreicher Angriff auf die Unternehmensressourcen erst viel zu spät oder überhaupt nicht bemerkt und führt somit zu unkalkulierbaren Schäden (Datenvernichtung, Datenmanipulation),
- können die Sicherheitssysteme nicht kontinuierlich auf dem neuesten technologisch relevanten Stand gehalten werden (dies ist sehr wichtig, da sich die Vorgehensweisen von Crackern häufig ändern und demzufolge die Sicherheitssoftware der neuen Situation angepasst werden muss),
- gelingt es häufig nicht, die eigene Unternehmensstrategie für den Bereich Datensicherheit wunschgemäß umzusetzen, da es niemanden gibt, der sich in der Thematik wirklich auskennt und somit zur Gestaltung einer entsprechenden Unternehmenspolitik beitragen kann (ein »Outsourcing« dieser Aufgaben hilft hier meist auch nicht weiter, da das Verständnis für unternehmenseigene Anforderungen extern oft nur unzureichend ausgeprägt ist).

**HINWEIS**

Das Thema Sicherheit im Netzwerk hat mittlerweile den entsprechenden Stellenwert innerhalb der Unternehmen erreicht und wird auch weiterhin spürbar an Bedeutung gewinnen.

## 7.1 Interne Sicherheit

Spricht man von Datensicherheit in Netzwerken, so bringt man heutzutage oft nur einen einzigen Aspekt mit diesem Thema in Verbindung. Es handelt sich dabei um Probleme, die sich in folgenden Fragen äußern:

- Wie kann eine Anbindung an das weltweite verteilte Internet vorgenommen werden, ohne sich dabei den Gefahren eines Eindringens engagierter Hacker (bzw. krimineller Cracker) in das eigene Netzwerk bzw. den eigenen Rechner auszusetzen?
- Welche Sicherheitsmaßnahmen sind zu treffen und wie stark werden die anfallenden Kosten zu Buche schlagen?

Das Thema *Security* ist allerdings wesentlich komplexer und bezieht sich nicht nur auf die Gefahr externer Attacken auf das Netzwerk, sondern ebenso auf die interne Infrastruktur des Unternehmensnetzwerks. Es sind unter anderem Probleme wie der unerlaubte Zugriff auf sensible Daten aus den verschiedensten Unternehmensbereichen (Personal, Entwicklung, Finanzen usw.), die beabsichtigte oder unbeabsichtigte Zerstörung wichtiger Datenbestände oder die Infiltration durch Virenprogramme, die eine Inhouse-Security unbedingt erfordern.

Ein besonderes Problem in diesem Zusammenhang sind die zumeist in Klartext dem Netzwerk übergebenen Daten. Die Mehrheit der Netzwerkprotokolle,

gerade im TCP/IP-Umfeld, kennt nur in sehr eingeschränktem Maße eine Datenverschlüsselung. Mit entsprechenden Analyse-Tools können somit auch sensible Daten problemlos eingesehen und gegebenenfalls aufgezeichnet werden. In der bereits verfügbaren, aber noch sehr verhalten eingesetzten IP-Version 6 wird dem Problem der Datensicherheit auf Netzwerkebene in weit höherem Maße Rechnung getragen.

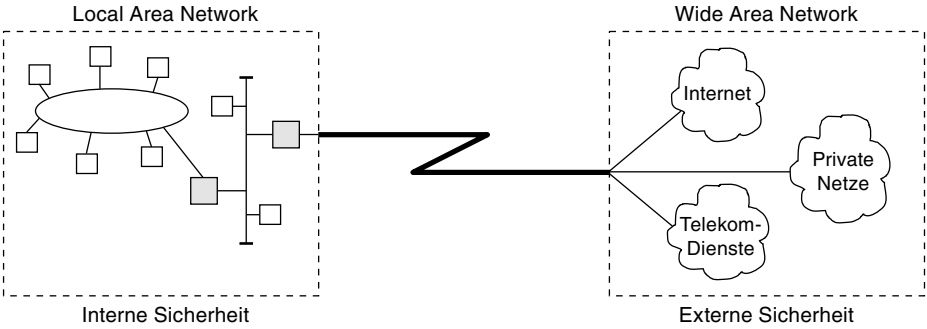


***Abb. 7–1*** *Einteilung zwischen interner und externer Sicherheit*

Bei allen geplanten Sicherheitsinstallationen sollte man jedoch zu keiner Zeit das richtige Augenmaß verlieren. Sicherheit hat natürlich auch ihren Preis (und damit ist nicht nur der monetäre Begriff gemeint). Sicherheit kann, sofern der Begriff äußerst restriktiv implementiert ist, eine Abteilung bzw. das Unternehmen erheblich lähmen. Mit einer Sicherheitsphilosophie, die nur Selbstzweck ist, wird niemandem gedient sein. Es hat sicher wenig Sinn, einen Netzwerkverbund aus dem Finanzbereich nahezu völlig zu isolieren, denn er ist natürlich auf Informationen aus zahlreichen Bereichen des Unternehmens angewiesen. Der Netzwerk- und Rechnerzugriff muss daher so gestaltet sein, dass dem Informationsbedarf einerseits und der Security andererseits in einem ausgewogenen Verhältnis Rechnung getragen wird.

Einige Möglichkeiten, Maßnahmen bzw. Technologien zur Realisierung von Sicherheit umzusetzen, werden nachfolgend dargestellt. Dabei wird zwischen der internen und der externen Sicherheit unterschieden (siehe Abb. 7–1). Letztere beginnt genau dort, wo Daten das Unternehmen, d.h. das eigene Netzwerk, verlassen und vom LAN (*Local Area Network*) an ein WAN (*Wide Area Network*) übergeben werden.

Die interne Sicherheit umfasst zunächst Sicherheitskonzepte des jeweiligen Rechners samt seiner definierten Benutzer. Sie lässt sich darüber hinaus auf das unmittelbare LAN bzw. den LAN-Verbund des Unternehmens erweitern. In diesem Umfeld können verschiedene Maßnahmen zur Realisierung einzelner Sicherungsmechanismen getroffen werden.

### 7.1.1 Hardwaresicherheit

Ein gewisses Maß an Sicherheit kann bereits durch einen vernünftig organisierten Sicherungsschutz an der jeweiligen Hardware erreicht werden. Es ist beispielsweise darauf zu achten, dass sensible Systeme, die etwa für die Steuerung von Produktionsabläufen eingesetzt werden, nicht unmittelbar zugänglich sind. Die Unterbringung in eigens für sie vorgesehenen, abschließbaren Schranksystemen sollte obligatorisch sein. Außerdem sollten regelmäßig durchgeführte Wartungen den störungsfreien Betriebszyklus garantieren.

### 7.1.2 UNIX-Zugriffsrechte

Das Betriebssystem UNIX (respektive auch die Linux-Derivate) stellt selbst einige wichtige Sicherungskonzepte zur Verfügung, die zumeist auf die Tatsache zurückzuführen sind, dass es sich um ein Mehrbenutzersystem handelt, bei dem der konkurrierende Zugriff verschiedener Benutzer auf gleiche und verschiedene Datenbestände Schutzmechanismen unbedingt erfordert. Anhand der folgenden Beispiele sollen diese erläutert werden: Standard-UNIX-Rechte, SVTX (*Sticky Bit)*, SGID (*Set Group Id*), SUID (*Set User Id*) und ACLs (*Access Control Lists*).

#### Standard-UNIX-Rechte

Nahezu sämtliche UNIX-basierenden Systeme (und dazu gehören auch die Linux-Derivate) ermöglichen für jedes Objekt, also jede Datei bzw. jedes Verzeichnis, die Zuordnung unterschiedlicher Zugriffsschutzmechanismen. Es existieren für jedes Objekt jeweils neun Zugriffsrechte, die folgendermaßen strukturiert sind:

| | | |
|---|---|---|
| Benutzerklasse: | Eigentümer | user |
| | Gruppe | group |
| | Andere | others |
| Rechte: | Lesen | read |
| | Schreiben | write |
| | Ausführen | execute |

Auf jede einzelne der drei Benutzerklassen lässt sich ein Lese-, Schreib- oder auch Ausführungsrecht übertragen, wobei das Schreibrecht das Leserecht impliziert. Die Rechte werden mit den Kennbuchstaben r (*read*), w (*write*) und x (*execute*) versehen. Außerdem erhalten die neun verschiedenen Zugriffsrechte die in der folgenden Tabelle dargestellte oktale Wertigkeit:

| Benutzerklasse | Leserecht (r) | Schreibrecht (w) | Ausführungsrecht (x) |
|---|---|---|---|
| user | 400 | 200 | 100 |
| group | 40 | 20 | 10 |
| others | 4 | 2 | 1 |

Für eine Kombination von Zugriffsrechten werden die dargestellten Werte einfach addiert. Das Kommando, mit dem eine Bestimmung bzw. eine Festlegung der Zugriffsrechte für Dateien oder Verzeichnisse vorgenommen werden kann, lautet CHMOD, wobei folgende Syntax zu beachten ist:

```
chmod [Oktalwert] [Objektname]
```

Die aktuelle Anzeige der Zugriffsrechte eines Objekts (Datei oder Verzeichnis) kann mit dem List-Befehl (ls) und der Option *-l* zur Anzeige gebracht werden, so wie nachfolgend beispielhaft dargestellt:

```
-rwxr-xr-x 1 root  bin     1316 May 5 1995 arch*
-rwxr-xr-x 1 root  bin   299012 Apr 30 1995 bash*
-rwxr-xr-x 1 root  bin    13316 Feb 14 1995 cat*
-rwxr-xr-x 1 root  bin    12292 Apr 29 1995 chgrp*
-rwxr-xr-x 1 root  bin    12292 Apr 29 1995 chmod*
-rwxr-xr-x 1 root  bin    12292 Apr 29 1995 chown*
drwxr-xr-x 1 root  bin    20484 Apr 29 1995 daten/
-rwxr-xr-x 1 root  bin    41988 Feb 14 1995 cpio*
-rwxr-xr-x 1 root  bin    16388 Apr 29 1995 dd*
-rwxr-xr-x 1 root  bin    12292 Apr 29 1995 df*
-r-xr-xr-x 1 root  bin    57348 May 11 1995 ftp*
-rwxr-xr-x 1 root  bin    12288 May 11 1995 ttysnoops*
-rwsr-xr-x 1 root  bin     8908 May 5 1995 umount*
-rwxr-xr-x 1 root  root   14760 Feb 17 1995 umssync*
```

Das in der Spalte für die Zugriffsrechte angegebene erste Bit zeigt den Dateityp an. Es sind folgende Werte definiert:

- `d` Directory
- `-` normale Datei
- `l` symbolischer Link
- `c` zeichenorientierte Spezialdatei
- `b` blockorientierte Spezialdatei
- `p` Dateibesonderheit: fifo

Beispiel:

```
chmod 420 testdatei
```

Mit dieser Anweisung wird dem Benutzer das Leserecht und der Gruppe das Schreibrecht zugewiesen; der Benutzerklasse *others* wird überhaupt kein Zugriffsrecht zugeordnet.

```
r-- -w- ---
```

Eine alternative Methode für die Zugriffsrechteverteilung ist die sogenannte symbolische Zuordnung. Hier erhalten die Benutzerklassen ebenso wie die einzelnen Zugriffsrechte Buchstabenkürzel: u (*user*), g (*group*) und o (*others*). Sollten gemeinsam für alle Benutzergruppen Zugriffsrechte vergeben werden, so geschieht dies mit dem Buchstabenkürzel a (*all*). Diese Methode wird im Allgemeinen dann eingesetzt, wenn zu den bereits bestehenden Zugriffsrechten weitere ergänzt bzw. teilweise entzogen werden sollen. Hier gilt folgende Syntax:

```
chmod [ugoa] [+-=] [rwx] [Objektname]
```

Beispiel:

```
chmod go+rw testdatei
```

Den bestehenden Zugriffsrechten werden für die Gruppe und für *others* die Lese- und Schreibrechte hinzugefügt.

Weitere UNIX-Anweisungen, die für eine Definition bzw. Manipulation von Zugriffsrechten verwendet werden, sind `unmask` oder `chown`.

**HINWEIS**

Bei der Vergabe von Zugriffsrechten muss darauf geachtet werden, dass die Schutzkonsistenz für Dateien und Verzeichnisse gewahrt bleibt, denn es ist durchaus möglich, Verzeichnissen andere Rechte zuzuordnen als den untergeordneten Dateiobjekten.

#### SVTX (Sticky Bit)

Normalerweise kann ein Benutzer in einem Verzeichnis, das ihm als »Eigentümer« zugewiesen ist, Dateien anlegen, lesen und auch wieder löschen. Haben diese von ihm generierten Dateien entsprechende Zugriffsrechte, so können diese Dateien auch von anderen Benutzern verändert oder gelöscht werden.

Ordnet man allerdings dem Verzeichnis, in dem sich die erstellte Datei befindet, durch die Anweisung *chmod +t [Verzeichnis]* ein sogenanntes *Sticky Bit* zu, so kann die Datei innerhalb des Verzeichnisses unabhängig von Zugriffsrechten lediglich vom Eigentümer (*owner*) gelöscht werden (der Oktalwert für das Sticky Bit ist 1000). Eine Datei mit den Zugriffsrechten - *rwx rwx rwx* ist demnach nur vom Eigentümer zu entfernen, obwohl die Zugriffsrechte normalerweise einen anderen Benutzer ermächtigen, den Löschvorgang erfolgreich durchzuführen. Das übergeordnete Verzeichnis trägt dann folgende Zugriffsrechte: *drwx rwx rwt*, wobei das letzte Bit das *Sticky Bit* ist.

**HINWEIS**

Das allgemein zugängliche Verzeichnis /tmp wird mit dem Sticky Bit versehen. Es kann durch die Zugriffsrechte *drwx rwx rwt* zwar von jedem Benutzer benutzt werden, allerdings kann ein Benutzer immer nur die eigenen Dateien wieder entfernen. Das versehentliche Löschen von Fremddateien wird somit verhindert.

SVTX (*Save Text Bit*) bewirkt – bei ausführbaren Dateien – eine bevorzugte Vorhaltung des jeweiligen Prozesses im Hauptspeicher (eine Art Cache). Bei einem erneuten Programmaufruf muss also nicht der gesamte Prozess wieder in den Speicher geladen werden, sondern es erfolgt ein direkter Zugriff auf den mit dem *Sticky Bit* versehenen Prozess. Durch ein zunehmend eingesetztes virtuelles Speichermanagement, das die Verweildauer von Prozessen im Hauptspeicher deutlich optimiert, hat diese Möglichkeit jedoch an Bedeutung verloren.

#### SUID (Set User Id) und SGID (Set Group Id)

Es gibt verschiedene Dateien (z.B. Systemdateien), die eines besonderen Zugriffsschutzes bedürfen. Sie werden daher automatisch dem Systemverwalter *root* als Eigentümer (*owner*) zugeordnet, mit entsprechenden Rechten versehen und sind somit für die anderen Benutzer nicht zu manipulieren. Gerade Systemdateien müssen jedoch regelmäßig auch von diesen Benutzern benutzt werden, um bestimmte Aktionen ausführen zu können, wie beispielsweise das Verändern des eigenen Kennworts durch das Kommando *passwd*. Da für die Manipulation von Kennwortdateien (*/etc/passwd*) Root-Berechtigung erforderlich ist, ein normaler Benutzer diese aber nicht besitzt, wird ihm durch das SUID-Bit lediglich für die Ausführungszeit des *passwd*-Kommandos die Root-Berechtigung erteilt. Allgemein formuliert bedeutet dies: Besitzt eine ausführbare Datei A das SUID-Bit, so erhält der jeweilige Benutzer für die Laufzeit des entsprechenden Prozesses die Berechtigung des Datei-Eigentümers. Dies führt dazu, dass auch Dateien manipuliert werden dürfen, die eigentlich einem anderen Benutzer (z.B. *root*) gehören – ein Umstand, der in einigen Situationen zwar erforderlich ist (Kennwortänderung), gleichzeitig aber auch ein nicht zu unterschätzendes Sicherheitsrisiko darstellt.

Während der Laufzeit des Prozesses besitzt der aufrufende Benutzer (bei Root-Berechtigung) volle Zugriffsrechte. Gelingt es ihm, ein Programm mit SUID-Bit, das eigentlich dem Systemverwalter (*root*) gehört, für seine Belange zu verändern, so kann er in einem UNIX-System durch den ungehinderten Zugriff auf gesicherte Root-Ressourcen erheblichen Schaden anrichten. Selbst die Tatsache, dass ein SUID auf Shell-Scripts nicht angewandt werden kann, ist durch die Integration in ein einfaches C-Programm zu umgehen.

Bei gesetztem SUID-Bit erhält der aufrufende Benutzer eine effektive *User-ID*, die lediglich für die Dauer des Programmlaufs gültig ist. Seine reale *User-ID* verändert sich in diesem Zeitraum nicht. So erhält die effektive *User-ID* während der

Ausführung des *passwd*-Kommandos den Wert *euid = 0* (0 ist der für den Benutzer *root* im System hinterlegte Wert), während die *ruid = nnn* unverändert bleibt und nicht überprüft wird. Allerdings gibt es Anweisungen, die eine Gegenüberstellung von *euid* und *ruid* vornehmen und nur bei völliger Identität entsprechende Manipulationen zulassen (beispielsweise das Kommando *mount*).

Mit der Eingabe *chmod u+s [ausführbare Datei]* erfolgt die Zuordnung des SUID-Bits. Entsprechend lautet die Eingabe *chmod g+s [ausführbare Datei]* für das SGID-Bit, das analog zum SUID-Bit die Group-ID weitergeben kann.

#### ACL (Access Control List)

Ein weiteres Instrumentarium zur Sicherstellung restriktiver Zugriffsrechte bietet die Erweiterung von UNIX-Standardrechten durch *Access Control Lists* (ACLs). Durch die Funktionen *permit* (ausdrückliche Erlaubnis), *deny* (Verweigerung) und *specify* (Spezifikation) können innerhalb editierbarer ACLs erweiterte Zugriffsrechte definiert werden. Diese haben gegenüber den UNIX-Standardrechten absoluten Vorrang, wobei folgende Kommandos eingesetzt werden:

- `acledit`
  Editieren der Zugriffsrechte und ihre syntaktische Überprüfung
- `aclput`
  Übertragung von ACLs auf andere Dateiobjekte
- `aclget`
  Lesen und Anzeige der ACLs für bestimmte Dateiobjekte

Sofern ACLs verwendet werden, erreichen sie normalerweise umfangreiche und damit äußerst unübersichtliche Ausmaße. Eine exakte Verwaltung ist dann meist nur noch sehr schwer möglich. Darüber hinaus werden ACL-Informationen bei Datensicherungen, die mit den Kommandos *tar* und *cpio* durchgeführt werden, nicht übertragen. Lediglich *backup* und *restore* übernimmt diese sicherheitsrelevanten Informationen.

ACLs werden allerdings auch häufig im Zusammenhang mit Netzwerkkomponenten wie Routern oder Switchen eingesetzt. Entsprechende Konfigurationen ersetzen zwar keine komplexen Regelwerke, wie sie beispielsweise bei Firewalls zum Einsatz kommen, können jedoch in bestimmten Fällen eine Basissicherheit liefern. Manchmal reichen solche ACLs zur Erfüllung der Sicherheitsanforderungen bereits aus. Viel häufiger werden ACLs bei Switchen und Routern jedoch zur funktionalen Konfiguration eingesetzt (beispielsweise bei einer VPN-Konfiguration zwischen zwei Routern, die eine ausschließliche Kommunikation über das konfigurierte VPN zulässt und jede andere Kommunikation unterbindet).

### 7.1.3 Windows- und macOS-Zugriffsrechte

Trotz oft diskutierter Sicherheitslücken bei Windows-Betriebssystemen hat sich auch hier mittlerweile ein sehr umfangreiches Sicherheitssystem entwickelt, das zur Realisierung eines hohen Sicherheitsstandards verwendet werden kann. In vielen Fällen sind Vergleiche mit UNIX-Systemen möglich. So können beispielsweise auch unter einem Windows-Serversystem einzelnen Dateiobjekten explizite Zugriffsberechtigungen zugeordnet werden. Abbildung 7–2 zeigt beispielhaft unter Windows 11 die Berechtigungen, die einer Datei zugeordnet werden können.

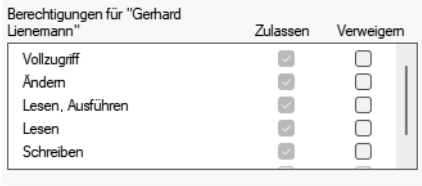


***Abb. 7–2*** *Sicherheitseinstellungen einer Datei unter Windows 11*

Die Zugriffsrechteverwaltung auf macOS 12.4 (Monterey) sieht im Unterschied dazu so aus, wie Abbildung 7–3 zeigt.

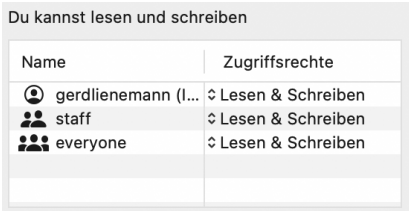


***Abb. 7–3*** *Sicherheitseinstellungen einer Datei unter macOS 12.4*

Hier können die Zugriffsrechte für die einzelnen Benutzer durch die in der Spalte »Zugriffsrechte« der Drop-down-Liste individuell vergeben werden.

**HINWEIS**

Aufgrund der hohen Komplexität kann im Rahmen dieses Buches auf eine weitere Darstellung der Windows- und macOS-Zugriffsrechte nicht eingegangen werden. Hier gilt die Empfehlung weiterführender Literatur.

### 7.1.4 Benutzerauthentifizierung

Hinter dem Begriff *Benutzerauthentifizierung* verbirgt sich im Wesentlichen die Benutzeranmeldung über einen Login-Prompt. Das Login (Anmeldung) ist ein Anwendungsprogramm, das nach einer im System definierten Authentifizierungsmethode vom Benutzer einen »einfachen Ausweis« verlangt. Dieser wird an Systemprozesse übergeben und mit gespeicherten Benutzerinformationen verglichen, bevor der Zugang zum System für den Benutzer geöffnet wird.

Normalerweise wird von einem Benutzer seine Kennung (*Username*) bzw. seine *User-ID* sowie ein Zugangskennwort verlangt. In einigen UNIX-Systemen kann eine Erweiterung dieser einfachen Authentifizierungsmethode vorgenommen werden, indem beispielsweise ein weiteres Kennwort angefordert wird. Leistungsfähige Systeme mit einigermaßen gut realisierten Sicherheitskonzepten lassen darüber hinaus auch besondere Kriterien für die Kennwortvergabe und deren Wartung zu (Kennwortmindestlänge, Kennwortwiederholungen, integrierbare Sonderzeichen, Kennwortablauf usw.).

Auf einem UNIX-System befinden sich in der Datei */etc/passwd* die relevanten Benutzerdaten, die zur Authentifizierung benötigt werden. Eine solche Datei könnte beispielhaft wie folgt strukturiert sein:

```
eva:FoMdWq20afN72:128:5:Benutzer Eva:/usr/eva/:/bin/ksh
gerd:Bpr2o2cy4HB4E:129:5:Benutzer Gerd:/usr/gerd:/bin/ksh
norbert:Hk.ABlAK4P53c:130:6:Benutzer Norbert:/usr/norbert:/bin/ksh
willi:Anr994nHc6E30:131:6:Benutzer Willi:/usr/willi:/bin/ksh
```

Jedem Benutzer, der am System definiert ist, wird eine eigene Zeile der Kennwortdatei zugeordnet. Welche Felder dabei verwendet werden, ist in der folgenden Tabelle dargestellt:

| Feldnummer | Bezeichnung | Beispielwert |
|---|---|---|
| 1 | Anmeldename (Kennung) | eva |
| 2 | Verschlüsseltes Kennwort | FoMdWq20afN72 |
| 3 | User-ID (numerisch) | 128 |
| 4 | Gruppen-ID (numerisch) | 5 |
| 5 | Kommentar | Benutzerin Eva |
| 6 | Login-Verzeichnis | `/usr/eva` |
| 7 | Startprogramm | `/bin/ksh` |

In den meisten UNIX-Implementierungen erfolgt eine Benutzerdefinition mit dem Kommando */usr/sbin/adduser*. Im Dialog werden die erforderlichen Benutzerinformationen abgefragt und in der Datei */etc/passwd* hinterlegt. Das Kennwort kann durch das Kommando */usr/bin/passwd* modifiziert werden. Die Datei */etc/group* definiert den Namen und die ID einer Gruppe und vermerkt außerdem alle Benutzer, die zu der entsprechenden Gruppe gehören.

### 7.1.5 Die R-Kommandos

Unter der Bezeichnung *R-Kommandos* oder *Berkley-Kommandos* leisten folgende drei Befehle einen wichtigen Beitrag zur Errichtung größtmöglicher Transparenz in einem UNIX-Netzwerk:

- *rlogin*remote login
- *rsh*remote shell
- *rcp*remote copy

Diese Anweisungen ermöglichen den Zugriff auf andere Systeme und damit auf deren Prozesse und Dateien, ohne dabei entscheidend restriktiven Schutzmechanismen zu begegnen. Die Erlaubnis, diese Kommandos zu benutzen, bedarf somit einer gründlichen Prüfung durch den System- bzw. Netzwerkverwalter.

Die Sicherheitsproblematik dieser Kommandos liegt darin begründet, dass für die Authentifizierung des aktiven Benutzers keine weiteren Kennwortprüfungen erforderlich sind. So werden lediglich Prüfungen auf Basis des eigenen Host-Namens bzw. der IP-Adresse sowie des Benutzernamens vorgenommen. Die entsprechenden sicherheitsrelevanten Dateien werden auf dem Rechner des Kommunikationspartners folgendermaßen definiert:

- Datei */etc/hosts.equiv*
  Diese Datei wird auf dem Ziel-Host generiert und enthält eine Liste »vertrauenswürdiger« Hosts und Benutzer, die eine Ausführung der *Berkley-Kommandos* vornehmen dürfen. Es lassen sich allerdings auch »negative« Einträge vornehmen, d.h., der Zugriff kann bestimmten Rechnern bzw. Benutzern explizit untersagt werden.
- Datei *[$HOME]/.rhosts*
  Mit dieser Datei erhält jeder einzelne Benutzer auf dem Ziel-Host die Möglichkeit, Zugriffsrechte zu vergeben oder zu verweigern, da die *.rhosts*-Datei in seinem Login-Verzeichnis angelegt und dort ausgewertet wird. Der Zugriffsschutz über die *.rhosts*-Datei sollte alternativ zur Datei */etc/hosts.equiv* verwendet werden, da Zugriffe, die in der Datei erlaubt werden, in der *.rhosts*-Datei nicht wieder abgelehnt werden können. Der Dateiaufbau entspricht dem der Datei */etc/hosts.equiv*.

### rlogin

Das Kommando *rlogin* (*remote login*) startet eine Terminal-Sitzung auf einem entfernten Ziel-Host. Die Syntax dieses Kommandos lautet:

```
rlogin [options] [-l username] hostname
```

Eine sofortige Anmeldung ohne explizite Authentifizierung ist somit (unter Verwendung der Option *-l*) ohne Weiteres möglich, beispielsweise wie folgt: *rlogin –l willi kiel*. Eine in verschiedenen UNIX-Systemen mehr oder weniger umfangreiche Liste von Optionen (siehe nachfolgende Tabelle) kann dem Sicherheitsaspekt wieder (ein wenig) Relevanz verschaffen.

| Option | Beschreibung |
|---|---|
| `-8` | Ermöglicht einen 8-Bit-Datenstrom. |
| `-E` | Escape-Character werden ignoriert; zusammen mit der Option -8 wird eine absolut transparente Verbindung realisiert. |
| `-K` | KERBEROS-Authentifizierung wird abgeschaltet. |
| `-d` | Schaltet das TCP-Socket-Debugging ein. |
| `-e` | Spezifizierung des Escape-Characters (als Zeichen oder als Oktalwert *\nnn*) |
| `-x` | Schaltet die DES-Verschlüsselung auf der *rlogin*-Session ein. |

### rsh

Die *Remote Shell* (*rsh*) führt auf einem entfernten Host ein Kommando aus. Dabei kopiert *rsh* seinen Standard-Input zum entfernten Kommando, das wiederum seinen Standard-Output an den Standard-Output des *rsh* weitergibt. Das *rsh*-Kommando wird normalerweise dann beendet, wenn auch der dadurch gestartete Prozess beendet ist. Die Syntax lautet:

```
rsh [options] [-l username] hostname [command]
```

Soll beispielsweise auf dem Ziel-Host *kiel* im Login-Verzeichnis des Benutzers *willi* eine Datei mit dem Namen *testdatei* angelegt werden, so muss folgende Eingabe erfolgen:

```
rsh -l willi kiel touch /home/willi/testdatei
```

### rcp

Die Anweisung *Remote Copy* (*rcp*) kopiert Dateien und Verzeichnisse zwischen zwei Systemen und verwendet dabei die Namenssyntax *filename@hostname.path*. Die vollständige Kommandosyntax lautet wie folgt:

```
rcp [options] source-file target-file
```

bzw. beim Kopieren von Verzeichnissen

```
rcp [options] [-r] source-file ... target-directory
```

Verwendbare Optionen sind in der nachfolgenden Tabelle dargestellt:

| Option | Beschreibung |
|---|---|
| `-r` | Handelt es sich bei dem *source-file* um ein Verzeichnis, das komplett (samt interner Hierarchiestruktur) auf den Zielrechner kopiert werden soll, so ist diese Option anzugeben. |
| `-p` | Ermöglicht die Beibehaltung von Time Stamp und Dateicharakteristika der Quelldatei während des Kopiervorganges. Die *umask*-Einstellung wird dabei ignoriert. |
| `-x` | Schaltet die DES-Verschlüsselung auf der *rlogin*-Session ein. |

### 7.1.6 Remote Execution (rexec)

Das Kommando *rexec* (*Remote Execution*) ist Bestandteil des TCP/IP und wird für die Jobausführung auf entfernten Rechnern verwendet. Es erfordert allerdings, anders als die Berkley-Kommandos, eine »normale« Benutzerauthentifizierung (wie bei *telnet* oder *ftp*). Die sicherheitskritische Komponente besteht hier in der Lesbarkeit des als Parameter übertragbaren Kennworts, sofern der sichere interaktive Dialog (Angabe von Benutzer und Kennwort) nicht verwendet werden soll. Für eine Jobautomation ist jedoch der mühsame Dialog oft ungeeignet, sodass die Sicherheitslücken zugunsten komfortabler Funktionalität bewusst in Kauf genommen werden.

Ein einigermaßen sicherer Automatismus lässt sich allerdings über die Datei *[$HOME]/.netrc* realisieren. In dieser Datei werden auf der lokalen Maschine Ziel-Host, Benutzername und Kennwort hinterlegt, sodass der *rexec*-Client bei Aufruf diese Datei lesen kann und die entsprechend hinterlegten Werte bei Kontaktaufnahme dem *rexec*-Server übergibt. Die so vorgenommene Authentifizierung ergibt jedoch nur dann Sinn, wenn die lokale Datei *./netrc* vor unberechtigtem Zugriff geschützt werden kann; mit der Eingabe *chmod 600 $HOME/.netrc* lässt sich dies realisieren. Damit besitzt lediglich der Dateieigentümer Schreib- und Leserechte; alle anderen Benutzer haben an dieser Datei keinerlei Rechte. Die Datei *[$HOME]/.netrc* ist folgendermaßen aufgebaut:

```
machine [hostname] login [username] password [password]
```

Beispieleinträge:

```
machine berlin login eva password fahrrad
machine kiel login willi password kugel
machine muenster login gerd password gurti
```

Bei Eingabe des Befehls *rexec kiel ls -lat* wird lediglich überprüft, ob der angesprochene Ziel-Host mit dem autorisierten Benutzer (hier: *eva*) in der Datei *.netrc* übereinstimmt, bevor der Befehl *ls -lat* auf dem Ziel-Host ausgeführt wird. Die allgemeine Syntax des Kommandos *rexec* lautet wie folgt:

```
rexec hostname [-l username] [-p password]
[option] command
```

## 7.2 Externe Sicherheit

Die bislang erörterten Sicherheitsaspekte bezogen sich auf unternehmensspezifische interne Komponenten wie die einzelnen Systeme oder den Rechnerverbund, also das lokale Netzwerk (LAN). Wenn auch einige der diskutierten Möglichkeiten (z.B. die *Berkley-Kommandos* oder die Standardrechte für UNIX- oder Windows-Systeme) in Bereichen netzwerkübergreifender Kommunikation eingesetzt werden können, so gehören sie doch zum klassischen Katalog interner Schutzmechanismen. Für die externe Sicherheit spielen andere Aspekte eine Rolle, die hauptsächlich den LAN-zu-LAN-Verbindungen über WANs (*Wide Area Networks*) Rechnung tragen. In der Unternehmenskommunikation erfolgt dabei die Auseinandersetzung mit einem besonderen Gefahrenmoment: dem Kontakt zu öffentlichen Netzwerken. Ob es lediglich um die Verbindung zweier verschiedener Standorte durch einfache Telefonleitungen geht, um den Aufbau eines vermaschten Router-Netzes oder die Anbindung an das Internet, es steht immer die Frage im Vordergrund: Wie können der sichere Datenfluss aus dem lokalen Standort hinaus und, vor allem, eine sichere dem Standort zufließende *Inbound*-Verbindung realisiert werden?

### 7.2.1 Öffnung isolierter Netzwerke

Über eines ist man sich bereits seit Längerem im Klaren: Die Zeiten der isolierten Kommunikation sind ein für alle Mal vorbei. Der Informationsbedarf eines Unternehmens mit verschiedenen Standorten ist derart gewaltig, dass er nur noch dann befriedigt werden kann, wenn die benötigten Informationen in einer äußerst kurzen Zeit zur Verfügung gestellt werden können. Entscheidungen in nahezu allen Unternehmensbereichen erfordern den unmittelbaren Zugriff auf Informationen. Wartezeiten von mehreren Stunden oder gar Tagen können nicht in Kauf genommen werden, um die Flexibilität des Entscheidungspotenzials nicht zu gefährden.

Darüber hinaus spielt natürlich auch die Quantität von Informationen eine große Rolle. Die Beurteilung komplexer ökonomischer Sachverhalte und die Berücksichtigung relevanter Nebenbedingungen erfordern die Zusammenführung