Einen NFS-Server konfigurieren

Bewertung: 3

Die Kandidaten sollten in der Lage sein, eine export-Datei zu erstellen und darin die freizugebenden Dateisysteme zu spezifizieren. Dieses Prüfungsziel beinhaltet das Editieren der Einträge in /etc/exports, um den Zugriff auf bestimmte Rechner, Subnetze oder Netzgruppen zu beschränken. Außerdem sind enthalten, die Angabe von mount-Optionen in der export-Datei, die Konfiguration von UserID-Mapping, das mounten eines NFS-Dateisystems auf einem Client, Mount-Optionen, um zwischen Soft-, Hard- oder Hintergrund-Wiederversuchen zu unterscheiden,sowie Signalverarbeitung, Locking und Block-Größen zu spezifizieren. Die Kandidaten sollten außerdem in der Lage sein, tcpwrappers zu konfigurieren, um NFS sicherer zu machen.

Schlüsseldateien, Begriffe und Hilfsmittel beinhalten:

1. /etc/exports
2. exportfs
3. showmount
4. nfsstat

NFS, das Network-Filesystem, ist eine Entwicklung der Firma Sun, die später ihren Weg in alle Unix-Systeme gefunden hat und dazu dient, Dateifreigaben in Unix-Netzen zu realisieren. Das wesentliche Funktionsprinzip läuft über die sogenannten Remote-Procedure-Calls (RPC), also Prozeduraufrufe auf anderen Rechnern. Diese Procedure-Calls werden - wie andere TCP/IP-Dienste auch - über Portnummern gesteuert, die festlegen, welcher Dienst gerade angefordert wird. Die zentrale Verwaltung der Portnummern des RPC wird von einem Programm namens portmap gesteuert. Dieses Programm muß zwingend laufen, wenn ein Rechner NFS-Dienste anbieten will.

Linux bietet zwei Arten von NFS an. Die ältere Methode läuft alleine über sogenannte Userspace-Programme, die neuere übergibt die Hauptfunktionalität des Servers an den Kernel weiter. Diese Darstellung bezieht sich auf die kernelbasierte NFS-Methode, auch wenn sich - zumindestens was die exports-Datei angeht - die meisten Dinge auch auf die alte Methode anwenden lassen.

In beiden Fällen benutzt der NFS-Server zwei Programme, die die eigentlichen Dienste zur Verfügung stellen. Zum einen muß der NFS-daemon rpc.nfsd laufen, der die eigentliche NFS-Funktionalität bietet. Bei kernelbasiertem NFS hat dieser Server sehr wenig zu tun, da der Kernel (bzw. das Modul nfsd.o) die Hauptlast trägt. Zum zweiten muß der Mount-Daemon rpc.mountd laufen, der für die Mount-Nachfragen der Clients zuständig ist und entscheidet, ob Clients berechtigt sind, Zugriff auf ein gewünschtes Dateisystem zu bekommen.

Die Datei /etc/exports

Freizugebende Dateisysteme können zwar mit dem Befehl exportfs temporär freigegeben werden, wenn aber nach jedem Systemstart grundsätzlich ein oder mehrere Dateisysteme freigegeben (exportiert) werden sollen, dann werden diese Dateisysteme in die Datei /etc/exports eingetragen.

Das Format dieser Datei ist sehr einfach. Grundsätzlich beschreibt jede Zeile (die nicht leer ist oder mit einem # beginnt) eine Freigabe. Sie besteht aus

```
Freizugebendes_Verzeichnis        Client(Optionen) Client2(Optionen2) ...
```

Am Anfang der Zeile steht also grundsätzlich das freizugebende Verzeichnis. Dem folgt eine Beschreibung der Rechner, die dieses Verzeichnis über NFS mounten dürfen (hier Clients genannt). Jedem Client-Eintrag folgt in Klammern ohne Leerzeichen zwischen Clientname und Klammer eine Liste von Optionen, die für diese Freigabe und diesen Client gültig sein sollen. Es können dann noch weitere Clientangaben für andere Rechner folgen, die durch ein oder mehrere Leerzeichen vom vorherigen Clienteintrag getrennt sind.

Die vollständige Erklärung der verschiedenen Formen, wie Client- und Optionsangaben gemacht werden, finden Sie in der entsprechenden Handbuchseite.

Beschränkung der Freigabe auf bestimmte Rechner, Subnetze oder Netzgruppen

Wenn ein Verzeichnis in /etc/exports freigegeben werden soll, dann wird normalerweise ein Rechnernamen angegeben, gefolgt von einer geklammerten Liste von Optionen, die für diesen Rechner gültig sind. So bedeutet die Zeile

```
/usr        einstein.bar.com(rw)
```

dass das Verzeichnis /usr dem Rechner einstein.bar.com freigegeben wird. Als Option wird hier die Schreibberechtigung (rw = read-write) gegeben. Soll ein Verzeichnis an alle Welt freigegeben werden, wird der Rechnernamen einfach weggelassen. Die Zeile

```
/public     (ro)
```

gibt also den Zugriff auf das Verzeichnis /public allen Rechnern frei, allerdings nur ReadOnly.

Da es aber häufig nötig ist, ganzen Mengen von Rechnern den Zugriff zu erlauben, können Jokerzeichen (Wildcards - * und ?) eingesetzt werden, um bestimmte Teile von Rechnernamen zu verallgemeinern. Allerdings gelten diese Muster nur innerhalb eines einzelnen Domainnamensabschnitts, sie repräsentieren also keinen Punkt. Die Zeile

```
/usr        *.bar.com(rw)
```

gibt das Verzeichnis /usr also allen Rechnern frei, deren Namen auf .bar.com endet, jedoch nicht Rechnern, deren Namen noch eine Subdomain enthalten wie etwa foo.sub.bar.com.

Es ist auch möglich, Subnetze anhand ihrer Netzmaske anzugeben, statt Rechnernamen zu benützen. Dazu wird eine IP-Netzwerkadresse zusammen mit ihrer Netzmaske eingegeben. Die Netzmaske wird von der Adresse mit einem Querstrich (Slash - /) getrennt und kann entweder in der klassischen Form (z.B. /255.255.255.0) oder in der Anzahl der gesetzten Maskenbits (z.B. /24) angegeben werden. Die folgenden Zeilen geben also beide den Zugriff allen Rechnern frei, die innerhalb des Netzes 192.168.100.0 mit der Maske 255.255.255.0 liegen:

```
/usr        192.168.100.0/24(rw)
/usr        192.168.100.0/255.255.255.0(rw)
```

Statt der Angabe von Rechnernamen oder IP-Netzadressen können auch noch NIS-Netzgruppen angegeben werden. Um einen Netzgruppennamen von einem Rechnernamen zu unterscheiden, wird ihm ein @ vorangestellt. Die Zeile

```
/usr        @gruppe1(rw)
```

meint also nicht den Rechner gruppe1, sondern die NIS-Netzgruppe.

Angabe von Freigabeoptionen

Wir haben oben schon in den Beispielen die Anwendung von Freigabeoptionen gesehen. Dort hatten wir jeweils die Option rw oder ro gesetzt. Grundsätzlich werden solche Optionen in Klammern angegeben, die **ohne Leerzeichen** der Clientangabe folgen. Mehrere Optionen werden mit Kommata voneinander getrennt.

Die Handbuchseite beschreibt alle gültigen Optionen, die wichtigsten im Alltagsbetrieb sind rw und ro.

Optionen für das UserID Mapping

Das größte Problem bei NFS-Freigaben liegt in der Sicherstellung des Zugriffsschutzes. NFS arbeitet hier genau wie das Unix-Dateisystem selbst mit den User- und GruppenIDs. Wenn eine Datei auf dem Server dem User otto gehört und dieser User die UID 501 besitzt, dann wird die Datei zusammen mit dieser Eigenschaft exportiert. Wenn jetzt aber auf dem NFS-Client, der das Verzeichnis mit unserer Beispieldatei mountet, der User hans die UID 501 besitzt, so hat er alle Rechte auf diese Datei.

Diese Problematik lässt sich in einem zentral verwalteten Netz dadurch umgehen, dass alle User auf allen Rechner die gleiche UID zugewiesen bekommen. Das ist mit NIS nicht schwer zu realisieren. Schwieriger ist es mit den Rechten des Systemverwalters root. Er hat ja immer die UID 0. Wenn ein NFS-Server ein Verzeichnis freigibt und ein Client dieses Verzeichnis mountet, dann wird automatisch der root-Zugang für den Systemverwalter des Clients auf die sogenannte anonyme UserID abgebildet. Die anonyme UserID ist standardmäßig die -2 oder - vorzeichenlos ausgedrückt - 65534. Das bedeutet, dass der Systemverwalter des Clients auf dem über NFS gemounteten Dateisystem nicht die Rechte des Users mit der UID 0 besitzt, sondern nur die des Users mit der UID 65534. Gleiches gilt für die GruppenID des Systemverwalters.

Wenn diese Einschränkung nicht gewünscht ist, der Systemverwalter des NFS-Clients also auch root-Rechte auf dem freigegebenen Verzeichnis haben soll, dann muß die Option **no_root_squash** angegeben werden. Das spielt immer dort eine Rolle, wo NFS-Server und -Client von dem selben Administrator verwaltet werden. Der folgende Beispieleintrag in /etc/exports gibt das Wurzelverzeichnis des Servers für den Rechner einstein.foo.com frei, ohne die

Einschränkung für root:

```
/           einstein.foo.com(rw,no_root_squash)
```

In manchen Fällen kann es auch notwendig sein, dass alle Userzugriffe auf die anonyme User- und GruppenID abgebildet werden. Für diesen Fall gibt es die Option **all_squash**. Zusätzlich ist es noch möglich, selbst zu bestimmen, welche UID und GID als anonyme IDs verwendet werden sollen. Dazu werden die Optionen **anonuid=** und **anongid=** verwendet. Das folgende Beispiel gibt also das Verzeichnis /ftp/pub an alle Rechner der Welt frei, wobei alle User auf die UID 123 und die GruppenID 555 gemappt werden:

```
/ftp/pub   (all_squash,anonuid=123,anongid=555)
```

Mounten eines NFS-freigegebenen Dateisystems

Auf der Clientseite werden NFS-Verzeichnisse gemountet, als wären sie lokale blockorientierte Geräte. Es wird also wiederum der mount-Befehl verwendet, nur statt der Angabe des blockorientierten Gerätes wird hier der Server und das zu mountende Verzeichnis (getrennt durch einen Doppelpunkt) angegeben. Die folgende Befehlszeile mountet also das freigegebene Verzeichnis /usr des NFS-Servers gauss.bar.com ins lokale Verzeichnis /mnt:

```
mount gaus.bar.com:/usr /mnt
```

Auf die selbe Art und Weise kann auch ein Eintrag in /etc/fstab vorgenommen werden, der das automatische Mounten beim Systemstart vornimmt. Auch hier wird statt der Blockdatei der Name des Servers und das Verzeichnis durch Doppelpunkt getrennt angegeben.

In diesem Fall kann es aber dazu kommen, dass der Startvorgang unseres Clients hängenbleibt, wenn der NFS-Server gerade nicht reagiert, etwa weil er heruntergefahren ist oder das Netz nicht funktioniert. Aus diesem Grund gibt es verschiedene Optionen für das mount-Programm, die für diesen Fall gedacht sind.

NFS-relevante Optionen für das mount-Programm

Die komplette Darstellung der Mount-Optionen für NFS finden Sie in der Handbuchseite nfs(5), die die NFS-relevanten Einstellungen in /etc/fstab beschreibt. All diese Optionen können natürlich auch via -o Parameter des mount-Programms manuell eingegeben werden, sollte ein NFS-Dateisystem von Hand gemountet werden sollen.

Wichtig ist - wie oben schon erwähnt - die Einstellung, wie das mount-Programm mit Timeouts umgeht, die dadurch entstehen, dass der Server gerade nicht erreichbar ist. Dazu stehen uns ein paar Mountoptionen zur Verfügung.

Grundsätzlich kann angegeben werden, nach wie vielen Zehntelsekunden ein erneutes Anforderungspaket geschickt werden soll, wenn eine Operation keine Antwort erhält. Der voreingestellte Wert ist hier 7 Zehntelsekunden und läßt sich mit der Option **timeo=*Wert*** modifizieren. Diese Timeouts werden intern als Minor-Timeouts bezeichnet. Wenn ein Minor-Timeout registriert wird, dann werden solange Wiederholversuche stattfinden, bis entweder 60 Sekunden verstrichen sind oder die mit der Option **retrans=*Wert*** festgelegte Anzahl (Standard=3) erreicht ist. Jetzt kommt es zu einem sogenannten Major-Timeout.

Wie mit einem Major-Timeout umgegangen wird, kann mit den Optionen **hard** oder **soft** festgelegt werden. Voreingestellt ist die Option **hard**. Wenn ein Dateisystem mit **hard** gemountet wurde, dann wird nach einem Major-Timeout erneut versucht, die fehlgeschlagene Operation zu wiederholen. Das passiert unendlich oft, bzw. so lange, bis es keinen Fehler mehr gab. Wurde dagegen das Dateisystem mit **soft** gemountet, so wird nach einem Major-Timeout aufgegeben und eine Fehlermeldung an das aufrufende Programm zurückgegeben.

Die beiden Optionen **hard** und **soft** beziehen sich hauptsächlich auf Timeouts, die im laufenden Betrieb vorkommen, also wenn ein NFS-Dateisystem längst gemountet ist, und bei einer Übertragung dann der Timeout vorkommt. Wenn es beim Mounten selbst zu einem Fehler kommt, dann kann auch hier noch durch weitere Optionen festgelegt werden, wie sich das mount-Programm verhalten soll. Das ist insbesondere wichtig, wenn der mount-Vorgang automatisiert beim Systemstart vorgenommen werden soll (Eintrag in /etc/fstab).

Normalerweise wird ein fehlgeschlagener Mount-Versuch einfach wiederholt. In diesem Fall spricht man von sogenannter Vordergrundabarbeitung, weil solange die erneuten Versuche laufen, keine anderen Mountvoränge abgearbeitet werden können. Mit Hilfe der Option **bg** kann aber auch festgelegt werden, dass ein fehlgeschlagener Mountversuch (im **hard** Modus) nicht einfach immer wieder im Vordergrund weiterversucht wird, sondern er im Hintergrund (bg=background) weiterläuft. Auch alle weiteren Mountanweisungen an diesen Server werden dann automatisch im Hintergrund stattfinden, so dass der Startvorgang des Rechners nicht unendlich lange dauert.

Eine laufende Dateisystemoperation kann normalerweise nicht durch Signale unterbrochen werden. Das kann - insbesondere bei hard-gemounteten Dateisystemen ärgerlich sein, weil eine hängende Operation dann nicht abgebrochen werden kann. Um festzulegen, dass Unterbrechungen durch Signale erlaubt werden sollen, gibt es die Option **intr**. Diese Option legt also fest, dass eine Dateisystemoperation durch Signale wie SIGTERM oder SIGKILL

(oder einfach durch Strg-C) unterbrochen werden kann.

Schließlich gibt es noch zwei wichtige Optionen, die den prinzipiellen Datentransfer betreffen. Mit **rsize** und **wsize** kann festgelegt werden, wie groß die verwendeten Blöcke sein sollen, jeweils für Lese- und Schreiboperationen. Mit **nolock** wird festgelegt, dass auf Dateisystem-Locks durch den Lockdaemon lockd verzichtet wird.

Weitere Hilfsprogramme

Es existieren noch zwei weitere Hilfsprogramme, die wir benötigen, um mit NFS umzugehen. Zum einen ist das das Programm showmount und zum anderen nfsstat. Beide sollen noch kurz beschrieben werden.

Das Programm showmount

Mit Hilfe des Programmes **showmount** können Informationen über NFS-Server abgefragt werden. Entweder werden Informationen auf dem lokalen Rechner abgefragt (es wird kein Hostname angegeben) oder es wird ein entfernter Rechner nach seinen Informationen gebeten. In diesem Fall wird einfach der entsprechende Hostname angegeben.

Die Art der Information ist unterschiedlich, je nachdem, welche Parameter angegeben wurden. Die wichtigsten Nachfragen sind

1. Welcher Client ist gerade aktiv
2. Welcher Client hat welches Verzeichnis gemountet
3. Welche Verzeichnisse werden vom Server angeboten

Die erste Abfrage benötigt keinerlei Parameter, außer dem Hostnamen des gewünschten NFS-Servers. Wenn der lokale Rechner abgefragt werden soll, kann selbst der Hostname noch weggelassen werden.

Die zweite Abfrage arbeitet mit der Option -a oder --all.

Die dritte (und meist interessanteste) Abfrage benötigt die Option -e oder --exports. Die Ausgabe enthält alle Einträge der /etc/exports-Datei des angegebenen Rechners. So kann also herausgefunden werden, welche Freigaben von einem bestimmten Server an welche Clients angeboten werden.

Eine vollständige Auflistung aller Parameter entnehmen Sie der Handbuchseite.

Das Programm nfsstat

Während showmount geeignet ist, auch auf fremde Rechner zuzugreifen, gibt nfsstat nur statistische Informationen über den lokalen NFS-Server, kann also nur auf dem Serverrechner selbst aufgerufen werden. Das Programm arbeitet mit dem /proc-Dateisystem zusammen und ermöglicht die Ausgabe von statistischen Informationen über die client- und serverseitigen Aktivitäten.

Dazu zählen etwa die Angaben, wieviele (und wieviel Prozent) bestimmte Aktionen wie read, write, create, rename, ... ausgeführt wurden.

Die vollständige Dokumentation entnehmen Sie der Handbuchseite.

Die Zusammenarbeit von NFS mit TCP-Wrappers

TCP-Wrappers beschreibt die Fähigkeit, einen Dienst nur für bestimmte Hosts zuzulassen oder zu verbieten. Normalerweise werden diese Angaben in /etc/hosts.allow und /etc/hosts.deny gemacht. Es existiert ein eigener Daemon, der diese Dateien auswertet und entsprechend Dienste startet oder eben nicht. Dieser Server heißt tcpd und wird normalerweise dazu benutzt, inetd-basierte Dienste zu schützen.

NFS wird nicht über inetd gestartet und wäre eigentlich also nicht geeignet, von TCP-Wrapper geschützt zu werden. Die moderneren Formen von NFS lassen aber diesen Schutz zu, da die Daemonen selbst auf /etc/hosts.allow zugreifen.

Interessanterweise kann nicht nur der Mount-Daemon rpc.mountd, sondern auch der Portmapper portmap durch den TCP-Wrapper-Mechanismus gestartet werden. Um das richtig anwenden zu können, muß jeweils der Daemon-Name richtig angegeben werden. Egal, wie die Binärdateien tatsächlich heißen, die Servernamen in /etc/hosts.allow müssen grundsätzlich **portmap** für den Portmapper und **mountd** für den Mount-Daemon lauten.

Um also etwa allen Rechnern der Domain `foo.com` den Zugriff auf den Mountdaemon zu erlauben, aber allen anderen Rechnern diesen Zugriff zu verbieten, könnten wir in /etc/hosts.allow folgende Zeilen eintragen:

```
mountd : .foo.com : ALLOW
```

```
mountd : ALL : DENY
```

Das bezieht sich schon auf die neuere Form von TCP-Wrappers, früher hätten wir das in die beiden Dateien hosts.allow und hosts.deny verteilen müssen.

[Zurück zur Startseite LPI 201] [Zurück zur LPI-Seite] [Zurück zur Hauptseite von Linux-Praxis] [Guestbook] [Kontakt]