Inhalt

Inhaltsverzeichnis

Übersicht

NFS (*Network File System*) ist die im UNIX-Bereich übliche Art, Verzeichnisse zu ex- bzw. zu importieren und dadurch auf mehreren Rechnern zu benutzen.

Es wird dabei transparent auf die freigegebenen Verzeichnisse zugegriffen. So kann also am Server das komplette `/home` oder `/usr`-Verzeichnis exportiert werden, welches dann vom Client übernommen wird - die Benutzer merken davon nichts (außer bei langsamen Netzen).

Das Angenehme ist, dass man bestimmte Verzeichnisse nur für bestimmte Rechner oder Domänen freigeben kann.

Wer einen festplattenlosen ThinClient aufziehen will, kann auch das gesamte DateiSystem des Servers per NFS importieren.

Siehe auch das Linux NFS-HowTo und die deutsche Übersetzung DE-Linux-NFS-HOWTO.

Um Datenverzeichnisse von anderen Rechnern einzubinden, kann auch der AutoMounter verwendet werden - dadurch reduziert sich bei sehr vielen Clients die Zahl der Mounts auf die wirklich gebrauchten.

NFS ist ein relativ unsicheres Protokoll und kann deshalb nur in geschützten Umgebungen (lokales Netz ohne feindliche Rechner) eingesetzt werden. NFS vertraut auf TCP/IP bzw. DNS und darauf, dass die Clientrechner zuverlässige Authentifizierung ihrer Benutzer machen und lässt sich dementsprechend über diese Mechanismen angreifen - zudem ist NFS unverschlüsselt. **Die Benutzerkennungen auf Server und Client müssen für die korrekte Funktion übereinstimmen.**

HowTo

Vorbereitung Server

Sofern das nicht automatisch bei der Installation des NFS Servers geschieht (apt-get install nfs-kernel-server).

Man installiere `portmap` - dies ist für NFS unbedingt notwendig, sonst gibt es merkwürdige Fehlermeldungen.

Außerdem braucht man natürlich entweder den Userspace-nfsd oder den Kernel-nfsd (und für letzteren natürlich auch den entsprechenden Support im Kernel).

Dann sorgt man dafür, dass portmap und nfsd automatisch beim Booten gestartet werden (Initskript entsprechend verlinken) und startet diese Dienste auch einmalig manuell per Initskript (erspart nen Reboot).

Bei RedHat müssen z. B. folgende Dienste laufen:

- network (versteht sich eigentlich von selbst)
- portmap
- nfslock
- nfs

Benutzerkennungen auf Server und Client müssen übereinstimmen also die Gruppen- und Passwortdatei:

/etc/group users:x:100:mein-Loginname (Beispiel users:x:100:detlef,tobias,sandra)
/etc/passwd mein-Loginname:x:1000:1000:mein-Loginname,,,:/home/mein-Loginname:/bin/bash

muss auf Server und Client identisch sein.

Bei vielen Clients kann man User- und Group-IDs (und anderes) auch über NIS synchron halten.

Konfiguration des Servers

Auf dem Server schreibt 'root' für jedes Verzeichnis, das freigegeben werden soll, eine Zeile in die Datei */etc/exports*:

```
/home           *.home.lan(rw,sync)
/save           *.home.lan(rw,sync)
```

Mit der Option async anstatt sync werden Schreibzugriffe vom Server gepuffert und damit schneller.

Zu beachten ist, dass die Freigaben automatisch vor "Rootzugriffen" geschützt sind. Der Admin `root` vom Client hat auf den freigegebenen Verzeichnissen also minimale Rechte (wie "nobody"). Um das *Ummappen* von root zu verhindern, kann die Option `no_root_squash` hinzu gefügt werden.

Näheres siehe `man 5 exports`.

Den * als Wildcard kann man nur bei Domainnamen verwenden, nicht bei IP-Adressen (diese kann man mit der Netzmaske angeben)

falsch	192.168.1.*
richtig	192.168.1.0/24
auch richtig	192.168.1.0/255.255.255.0

Damit die Änderung wirksam wird, auf dem Server als root ausführen:

```
exportfs -a -r
```

Anmerkung: bei meinem Debian-System *3.1r2 sarge* ist nach Änderungen in */etc/exports* erforderlich:

```
cd /etc/init.d
./nfs-common restart
```

oder den Server neu starten.

Wichtig: Es dürfen keine Unterverzeichnisse von exportierten Verzeichnissen freigegeben werden:

Bsp:

```
/raid           tiger.linux.lan(ro)
/raid/doc       tiger.linux.lan(rw)
```

Hier ist der 2. Eintrag falsch, da /raid/doc ein Unterverzeichnis von /raid ist. Dies führt zu der Fehlersituation, dass der Export funktioniert, beim mounten auf dem Client jedoch die Meldung "permission denied" erscheint.

Überprüfen der Serverfreigabe

Um zu überprüfen, ob die Verzeichnisse auch exportiert werden, könnt Ihr folgenden Befehl auf der lokalen Maschine absetzen:

```
showmount -e [Nodename]
```

Vorbereitung Clients

Auf den Clients muss ebenfalls der Portmapper laufen. Hierzu ist ein korrekt konfiguriertes Loopbackdevice notwendig.

Bei RedHat müssen folgende Dienste aktiv sein:

- portmap
- nfslock

NFS ohne portmapper

Man kann den NFS Client auch ohne Portmapper betreiben. Allerdings muss man dann das Locking abstellen (wodurch dann auch der nfslock überflüssig wird), da ansonsten beim Mounten versucht wird, Kontakt zum portmapper aufzunehmen. Dies macht man mit der zusätzlichen Option `nolock` in der `/etc/fstab` (siehe nächsten Abschnitt).

Konfiguration Clients

Auf den Clients hängt man an die `/etc/fstab` folgendes an -

```
<servername>.<netzname>.<lan>:/home  /home  nfs  rsize=8192,wsize=8192,hard,intr  0 0
<servername>.<netzname>.<lan>:/save  /s  nfs  rw,user,noauto  0 0
```

Dann benennt man auf dem Client das existierende `/home` erstmal um, (z.B. `mv /home /home.local`) oder unmountet das bisherige /home (`umount /home`) - als root natürlich.

Das zweite Beispiel zeigt, dass ein beliebiges Verzeichnis der Servers (hier: /save, könnte auch /home/otto sein) in den Arbeitsrechner (client) eingebunden werden kann. Bevor ein beliebiger Benutzer das Verzeichnis /s durch den nfs-Zugriff ersetzen kann muss root am Arbeitsrechner (client) folgenden eingeben:

```
mkdir /s
chmod ugo+rw /s
```

Danach kann ohne reboot des Clients das /home oder /s vom Netz gemountet werden:

```
mount /home
mount /s
```

und nun sollte `/home` den Inhalt von `/home` auf dem Server anzeigen.

Die angegebenen Optionen, `rsize=8192,wsize=8192,hard,intr` bedeuten folgendes:

- `rsize` und `wsize` sind Tuning-Optionen, die NFS beschleunigen. Man sollte sie nicht auf den Defaultwerten belassen, da dies die NFS-Verbindung bis zu einem Faktor 10 ausbremsen kann. Dies kann dann auch gerne zu weiteren Problemen wegen der Timeouts führen.
- `hard,intr` sind Kompatibilitätsoptionen, die bei manchen Programmen sonst Probleme machen könnten. `hard` lässt Programme bei Unterbrechungen ohne Timeout warten bis der Server wieder normal erreichbar ist. `intr` erlaubt ein wartendes Programm bei Bedarf dennoch zu unterbrechen/killen.
  - Und wie kann man das Dateisystem nun unmounten wenn der Server weg ist, und dadurch alles mögliche blockiert ist?
    - Da empfielt sich `umount -l <directory>`. Oder auch `umount -f <directory>` - wobei das gerade bei NFS nicht immer gut funktioniert. Beschreibung zu den Parametern siehe manpage von umount.
- Eine weitere mögliche Option hier ist `nolock`. Diese sollte benutzt werden, wenn kein portmap/lockd läuft und das locking sowieso nicht funktioniert oder benötigt wird. (z.B. bei AutomatisierteInstallation)

Weitere interessante mount-Optionen:

- `nodev`
- `nosuid`

Siehe auch `man mount`.

Der neue Kernel NFS Service ("neu" = ab v2.2)

Wenn man den Kernel-NFS-Daemon benutzen will (z.B. weil man File Locking benötigt), muss man allerdings jeden Rechner, an den die jeweiligen Dateisysteme exportiert werden sollen, explizit in der */etc/exports* aufführen - sonst kann es u.U. Probleme bei Wiederverbindung nach Netzwerkunterbrechungen geben (der Tipp ist aus der linux-kernel

Mailingliste). Der *knfsd* ist auch sonst zu empfehlen, er ist nämlich etwas schneller als der normale *nfsd*. Allerdings hat der *knfsd* (Stand Kernel 2.2 - ist das auch später noch der Fall?) das Problem, dass er a) noch nicht ganz fertig ist und b) mandatory locks nicht vernünftig implementiert, falls er also nicht funktioniert, benutzt man einfach den alten user-space *nfsd* - der ist zwar langsamer, aber dafür auch bombenstabil.

Der Standard Red Hat 7.3 Kernel (=2.4.18-3) hat mehrere massive Bugs, die ihn als NFS Client unbrauchbar machen. Lösungsmöglichkeit: Upgrade auf 2.4.18-5 oder aktueller.

Im LinuxKernel 2.4.19 scheint ein Fehler im Kernel NFS Service enthalten zu sein, welcher bei gleichzeitigem Lesen und Schreiben von einem NFS-Verzeichnis zu Fehlern beim Schreiben führt (cp meldet: Input/Output error und bricht ab). Dies scheint im LinuxKernel 2.4.20 behoben zu sein.

Fragen und Antworten

Frage : Was kann/muss ich machen wenn der Server nicht erreichbar ist und Programme blockiert werden ich jetzt aber lieber einen Abbruch bzw. eine Fehlermeldung erzwingen will? Ich kann das NFS nicht unmounten, um es wieder richtig zu mounten. Geht das denn nicht ohne kompletten Neustart? Hätte ich mal besser vorher den mount gelöst.

- Wenn das NFS Filesystem "hard" gemountet ist (default), dann ist das unmounten nicht möglich. Details siehe untenstehend.
- Um zu vermeiden, dass I/O Operationen bei der Nichtverfügbarkeit des NFS Servers hängen, muss man beim Mounten des Filesystems die option "intr" (mount -o intr) angeben. Dies bewerkstelligt, dass File I/O Operationen unterbrochen werden können. Zusätzlich kann auch die Option "soft" beim mount angegeben werden, damit Filesystemoperationen automatisch abgebrochen werden. Siehe nächster Punkt unten.
- Normalerweise ist es nicht möglich, über NFS gemountete Freigaben zu unmounten, wenn der NFS Server nicht mehr verfügbar ist, da das mounten per default als "Hard" erfolgt (mount -o hard). Dabei wird immer gewartet bis der NFS Server wieder verfügbar ist. Wird das Filesystem allerdings "Soft" gemountet (mount -o soft) so werden Filesystem Operationen abgebrochen sobald der Server nicht mehr verfügbar ist, das Unmounten ist damit möglich.

Frage : Wie kann man am besten die Homeverzeichnisse in Clients einmounten die gleichzeitig auch lokale User beherbergen sollen? falls der Server (Erstrechner) gerade aus ist. (in BSD kann man wohl die lokalen Verzeichnisse unter /home sichtbar lassen, auch wenn ein remote /home "drübergemounted" wird.) Das Problem was ich für umgelegte lokale Homeverzeichnisse sehe ist, dass manche Programme einfach von /home/benutzername ausgehen und nicht z. B. /home.local/benutzername. Hat da jemand Erfahrungen?

- InterMezzo sollte AFAIK genau da helfen.

Frage : Da NFS ob der fehlenden Verschlüsselung unsicher ist, wie realisiert man dann alternativ das Im- und Exportieren von Verzeichnissen übers Netz besser? Selbst M$ wirbt inzwischen damit, dass SMB/CIFS verschlüsselt und Linuxfreigaben unverschlüsselt seien .

- Man könnte auf dem Client die unverschlüsselte Freigabe via CFS Freigeben und das über loopback mounten. Nicht schön, aber sicher.
- Dazu taugen auch alle Formen von sicheren Tunneln. Etwa ein SSH-Tunnel auf den Server oder ein VPN.

OK, man kann ssh und unter KDE mit dem fish:/-Protokoll ganz komfortabel arbeiten, aber somit ist beispielsweise ein transparentes Mounten eines kompletten /home nicht so schön möglich. Ideen?

- Doch das geht schau mal nach shfsmount -- ReimarBauer 2004-03-27 08:17:12
- Alternativ dazu auch mit lufs bzw. sshfs von lufs.

Frage: Ist es moeglich ACL unter NFS (Linux) zu benutzen?

- Ja das geht. Natürlich muss das Dateisystem auf dem Server ACLs unterstützen.

Offene Frage: Wie kann man das schreiben auf optische Medien von einer Quelle auf einen NFS-Dateisystem beschleunigen? Ich schreibe hier gerade mit 1-2 facher Geschwindigkeit (300KB/s) an einem 100Mbit Netzwerk und komme viel zu spät zu meiner Verabredung!

- Ich würde die Daten erst einmal lokal kopieren, und danach erst brennen. Geht meistens problemloser und schneller. Ich hoffe nur, deine Verabredung war nicht zu sauer

Tipps und Tricks

Vor jeder Änderung alle Clients unmounten!

Werden mehrere Verzeichnisse eines NFS-Servers importiert, so empfiehlt es sich, bei Änderungen alle mounts zu lösen, da sonst beim Export ("exportfs -a") die Fehlermeldung "invalid parameter" erscheint und ein mount dann zur Fehlermeldung "keine Berechtigung" führt. Der "schwierige Zusammenhang" zwischen Fehlerbehebungsort und Fehlermeldung hat mich viel Zeit gekostet.

Alternativen

- Aufgrund von Sicherheitsaspekten ist wohl SHFS die Lösung: Es verschlüsselt und kann "ganz normal" gemountet werden, wie auch NFS:
  - Je nach Berechtigung durch User
  - In der fstab

    Weitere Infos befinden sich auf SHFS.
- SSHFS http://fuse.sourceforge.net/sshfs.html
  - mehr oder minder das gleiche wie SHFS, aber als File System in User Space (FUSE) implementiert (man benötigt keine root-Rechte, um es zu verwenden)
- AFS
- Coda
- arla
- NFS Version 4
  - ist bereits im 2.6 Kernel enthalten, userspace tools sind in den meisten Distributionen wohl noch nicht enthalten
- SecureNFS
  - NFS (und NIS) durch ssh getunnelt

KategorieProtokoll