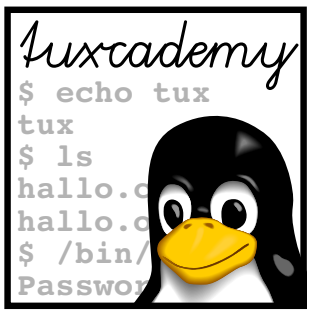
Linux-Infrastrukturdienste

DHCP, PAM, LDAP, Samba und NFS



tuxcademy – Linux- und Open-Source-Lernunterlagen für alle
www.tuxcademy.org · info@tuxcademy.org

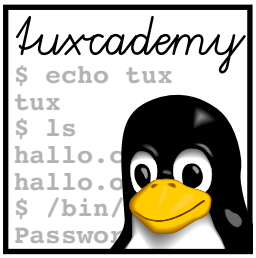
Das tuxcademy-Projekt bietet hochwertige frei verfügbare Schulungsunterlagen zu Linux- und Open-Source-Themen – zum Selbststudium, für Schule, Hochschule, Weiterbildung und Beruf.
Besuchen Sie https://www.tuxcademy.org/! Für Fragen und Anregungen stehen wir Ihnen gerne zur Verfügung.

Linux-Infrastrukturdienste DHCP, PAM, LDAP, Samba und NFS

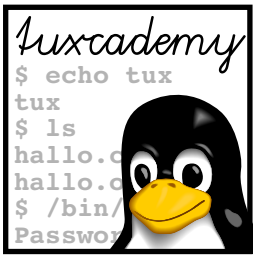
Revision: infs:4d61969997c6112b:2013-01-15
fsrv:52fdf9fd96574d15:2013-01-15 4–10
infs:3f4a9c41f082bf5f:2012-10-15 1–3
infs:CD32wPcI0SlMPjLY2xnBW3

http://www.tuxcademy.org · info@tuxcademy.org

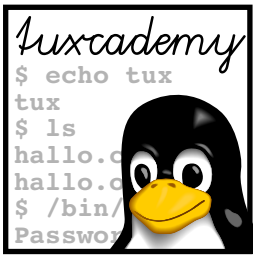
Autoren: Tobias Elsner, Anselm Lingnau
Technische Redaktion: Anselm Lingnau ⟨*anselm.lingnau@linupfront.de*⟩
Gesetzt in Palatino, Optima und DejaVu Sans Mono



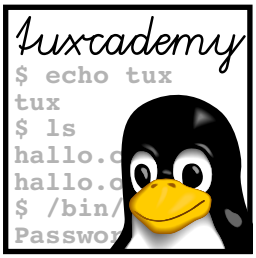
Inhalt



Tabellenverzeichnis



Abbildungsverzeichnis



Vorwort

Linux-Systeme spielen durch ihre Stabilität, Leistung, Flexibilität und günstige Kostenstruktur eine sehr wichtige Rolle als Server in lokalen Netzen und sind auch aus heterogenen Umgebungen nicht mehr wegzudenken. Dieser Kurs erweitert die in *Linux-Netzadministration* vermittelten Kompetenzen um Wissen über die Konfiguration und den Betrieb von Basisdiensten für Unternehmensnetze wie DHCP, PAM und LDAP und liefert damit nicht nur das Fundament für den professionellen Einsatz von Linux in der Praxis, sondern legt auch Grundlagen für fortgeschrittene Zertifizierungen wie LPIC-3. Ferner gibt er einen Einstieg in die Konfiguration von Linux als Server für andere Linux- und Unix-Systeme mit NFS und als Server in einer Windows-Umgebung mit Samba.

Voraussetzung für den erfolgreichen Abschluss ist das LPIC-1-Zertifikat oder äquivalente Kenntnisse, etwa aus den Kursen bis zu *Linux-Administration II* sowie die Teilnahme an *Linux-Netzadministration* oder äquivalente Kenntnisse. Für das Thema „Samba“ sind Windows-Kenntnisse von Vorteil.

Diese Schulungsunterlage soll den Kurs möglichst effektiv unterstützen, indem das Kursmaterial in geschlossener, ausführlicher Form zum Mitlesen, Nach- oder Vorarbeiten präsentiert wird. Das Material ist in Kapitel eingeteilt, die jeweils für sich genommen einen Teilaspekt umfassend beschreiben; am Anfang jedes Kapitels sind dessen Lernziele und Voraussetzungen kurz zusammengefasst, am Ende finden sich eine Zusammenfassung und (wo sinnvoll) Angaben zu weiterführender Literatur oder WWW-Seiten mit mehr Informationen.

Kapitel

Lernziele

Voraussetzungen

Zusätzliches Material oder weitere Hintergrundinformationen sind durch das »Glühbirnen«-Sinnbild am Absatzanfang gekennzeichnet. Zuweilen benutzen diese Absätze Aspekte, die eigentlich erst später in der Schulungsunterlage erklärt werden, und bringen das eigentlich gerade Vorgestellte so in einen breiteren Kontext; solche »Glühbirnen«-Absätze sind möglicherweise erst beim zweiten Durcharbeiten der Schulungsunterlage auf dem Wege der Kursnachbereitung voll verständlich.

Absätze mit dem »Warnschild« weisen auf mögliche Probleme oder »gefährliche Stellen« hin, bei denen besondere Vorsicht angebracht ist. Achten Sie auf die scharfen Kurven!

Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind numeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad angegeben. Aufgaben, die mit einem Ausrufungszeichen (»!«) gekennzeichnet sind, sind besonders empfehlenswert.

Übungsaufgaben

Auszüge aus Konfigurationsdateien, Kommandobeispiele und Beispiele für die Ausgabe des Rechners erscheinen in `Schreibmaschinenschrift`. Bei mehrzeiligen Dialogen zwischen Benutzer und Rechner werden die Benutzereingaben in **`fetter Schreibmaschinenschrift`** angegeben, um Missverständnisse zu vermeiden. Wenn Teile einer Kommandoausgabe ausgelassen wurden, wird das durch »◁◁◁◁◁« kenntlich gemacht. Manchmal sind aus typografischen Gründen Zeilenumbrüche

erforderlich, die in der Vorlage auf dem Rechner nicht stehen; diese werden als »▷ ◁« dargestellt. Bei Syntaxdarstellungen stehen Wörter in spitzen Klammern (»⟨*Wort*⟩«) für »Variable«, die von Fall zu Fall anders eingesetzt werden können; Material in eckigen Klammern (»[`-f` ⟨*Datei*⟩]«) kann entfallen und ein vertikaler Balken trennt Alternativen (»`-a`|`-b`«).

Wichtige Konzepte

Definitionen

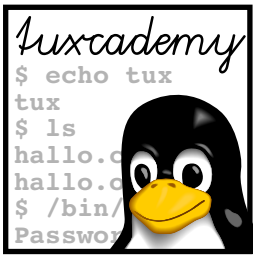
Wichtige Konzepte werden durch »Randnotizen« hervorgehoben; die **Definitionen** wesentlicher Begriffe sind im Text fett gedruckt und erscheinen ebenfalls am Rand.

Verweise auf Literatur und interessante Web-Seiten erscheinen im Text in der Form »[GPL91]« und werden am Ende jedes Kapitels ausführlich angegeben.

Wir sind bemüht, diese Schulungsunterlage möglichst aktuell, vollständig und fehlerfrei zu gestalten. Trotzdem kann es passieren, dass sich Probleme oder Ungenauigkeiten einschleichen. Wenn Sie etwas bemerken, was Sie für verbesserungsfähig halten, dann lassen Sie es uns wissen, etwa indem Sie eine elektronische Nachricht an

```
info@tuxcademy.org
```

schicken. (Zur Vereinfachung geben Sie am besten den Titel der Schulungsunterlage, die auf der Rückseite des Titelblatts enthaltene Revisionsnummer sowie die betreffende(n) Seitenzahl(en) an.) Vielen Dank!



1

Das Dynamic Host Configuration Protocol (DHCP)

Inhalt

Lernziele

- Zweck und Einsatzgebiete von DHCP kennen
- Den Protokollablauf von DHCP verstehen
- Den ISC-DHCP-Server konfigurieren und administrieren können

Vorkenntnisse

- TCP/IP-Kenntnisse
- Kenntnisse über Linux-Systemkonfiguration
- Kenntnisse über Linux-Netzkonfiguration

1.1 Warum DHCP?

IP-Adresse
Netzmaske
Standardgateway
DNS-Server

Damit ein Linux-Rechner an einem TCP/IP-Netz teilnehmen kann, braucht er gewisse Informationen – zumindest eine IP-Adresse sowie eine Netzmaske (Broadcast- und Netzadresse lassen sich daraus berechnen), dazu meist noch die IP-Adresse eines Standardgateways und eines oder mehrerer DNS-Server.

Nach oben sind natürlich keine Grenzen gesetzt: Adressen von NTP- und WINS-Servern, NetBIOS-Parameter, statische Routen, Informationen für das Booten von plattenlosen Clients über das Netz und vieles andere mehr können Bestandteil einer Netzkonfiguration sein.

Es ist möglich, alle diese Parameter auf jedem Rechner manuell zu setzen. Allerdings leuchtet es ein, dass diese Option gerade in umfangreichen Netzen mit Hunderten oder Tausenden von Stationen nicht mehr wirklich verlockend ist. Ein Ansatz mit einem zentralen Server, wo die Konfigurationsdaten für viele oder alle Clients gewartet werden können, ist viel sinnvoller.

Dynamic Host Configuration Protocol

Das **Dynamic Host Configuration Protocol** (DHCP) stellt einen solchen Ansatz dar. Es wurde 1993 von Ralph Droms in [RFC1531] beschrieben und in [RFC2131] (1997) noch einmal verbessert. DHCP gestattet es, diverse Konfigurationsparameter in einer Tabelle zu speichern und Clients entweder auf der Basis ihrer MAC-Adresse zuzuordnen oder völlig dynamisch zuzuteilen. Ein DHCP-Client fordert Konfigurationsdaten über einen Broadcast an (für den er noch keine spezifische Netzkonfiguration braucht) und kann die Angaben aus der Antwort dann nutzen, um seine Netzschnittstelle so zu konfigurieren, dass sie zum aktuellen Netz passt.

BOOTP

DHCP ist eine Erweiterung und Weiterentwicklung des »Bootstrap Protocol« (BOOTP), das 1985 von Bill Croft und John Gilmore der Öffentlichkeit präsentiert wurde [RFC0951].

Die wesentlichen Unterschiede zwischen DHCP und BOOTP bestehen darin, dass BOOTP sich auf die Konfiguration der Netzschnittstelle beschränkt (DHCP kann, wie gesagt, mehr Daten übermitteln) und DHCP im Gegensatz zu BOOTP die Vergabe von Parametern »auf Zeit« unterstützt. Damit können möglicherweise nur in begrenztem Umfang vorhandene Parameterwerte (Stichwort: IP-Adressen) nach einer gewissen Zeit an den Server »zurückfallen« und neu vergeben werden, auch wenn Stationen – etwa Notebook-PCs – still und heimlich aus dem Netz verschwinden, ohne sich offiziell beim DHCP-Server abzumelden. (Vor allem im Umgang mit WLANs ist das hilfreich.)

BOOTP und DHCP sind sich so ähnlich, dass DHCP-Server auch BOOTP-Clients bedienen können (falls Sie irgendwo noch welche finden). Umgekehrt geht das leider nicht.

Übungen

1.1 [!1] Verwendet Ihr Rechner eine fest voreingestellte Netzkonfiguration oder DHCP? Warum? Wo würden Sie die aktuelle Einstellung ändern?

1.2 DHCP-Protokollablauf

1.2.1 Client-Konfiguration ohne gültige Adresse

Der offensichtlichste Fall, in dem DHCP zum Einsatz kommen muss, besteht darin, dass ein Client frisch eingeschaltet wird oder aus anderen Gründen über keine gültige Netzkonfiguration verfügt.

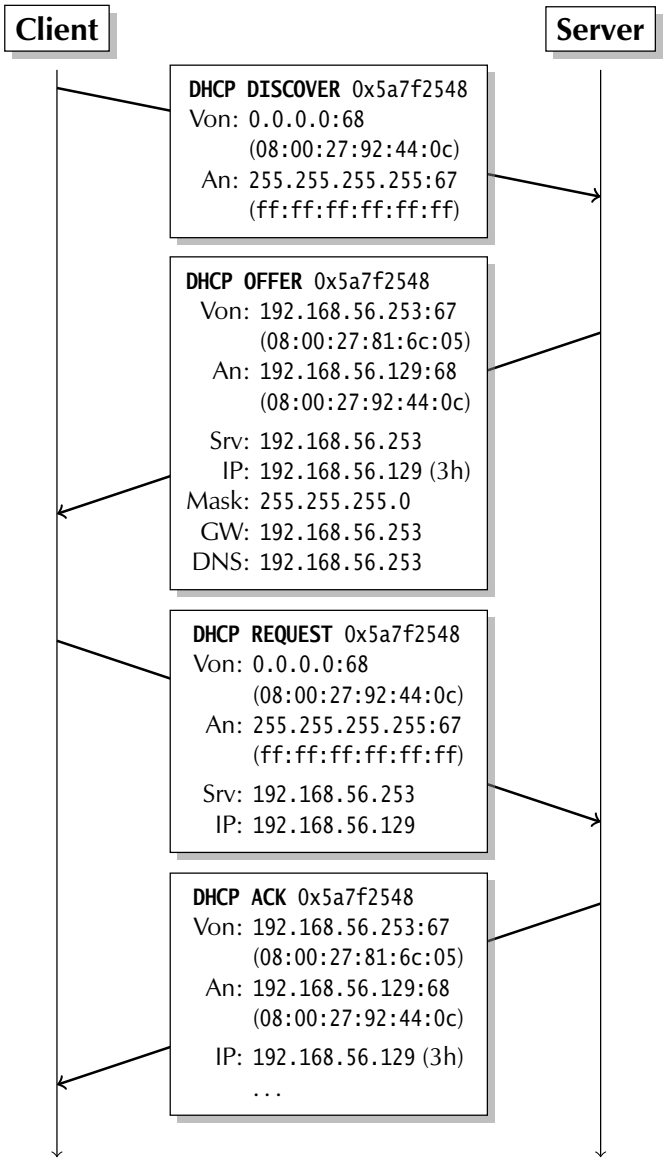


Bild 1.1: Ein DHCP-Client holt sich Netzparameter

Zu diesen »anderen Gründen« gehört beispielsweise, dass die Netzkonfiguration ihr »Verfalldatum« überschritten hat oder die Station zwar eine plausible Netzkonfiguration hat, diese aber nicht zum aktuellen Netz passt, in dem sie sich befindet.

Bild 1.1 zeigt einen typischen Dialog zwischen einem DHCP-Client (links) und einem DHCP-Server (rechts). Die folgenden Beobachtungen sind bemerkenswert:

- Der Client schickt ein `DHCPDISCOVER`-Datagramm, auf das der Server (im Idealfall) mit einem `DHCPOFFER` antwortet. Das heißt, er ist bereit, dem Client die darin enthaltenen Netzparameter zur Verfügung zu stellen. Der Client prüft, ob diese ihm genehm sind, und signalisiert seine Zustimmung mit einem `DHCPREQUEST`. Der Server bestätigt die Zuteilung dann mit einem `DHCPACK`.

  Da gleichzeitig mehrere DHCP-Transaktionen im lokalen Netz aktiv sein können (denken Sie daran, dass in der Firma alle Angestellten um 8 Uhr früh ihre Rechner starten), werden alle Datagramme durch eine eindeutige Transaktionsnummer gekennzeichnet (in unserem Beispiel `0x5a7f2548`). Damit kann jeder DHCP-Client sich die für ihn gül-

tigen Antworten aus dem Netzverkehr herausfischen, auch ohne dass er schon eine eigene Adresse hat.

Broadcast-Adresse

- Der Client verwendet für seine Datagramme die Absenderadresse `0.0.0.0` und die Zieladresse `255.255.255.255`. Letztere ist eine »begrenzte« Broadcast-Adresse, die nicht geroutet wird, sondern nur im lokalen Netz gilt (siehe hierzu auch [Ste94, Abschnitt 12.2]). Da der Client noch gar nichts über seine Netzumgebung weiß, werden seine Datagramme auch auf der Ethernet-Ebene als Broadcasts (mit der MAC-Adresse `ff:ff:ff:ff:ff:ff`) verschickt.
- Der Server verwendet für seine Datagramme seine eigene Absenderadresse (ist ja kein Problem) und die Zieladresse, die er dem Client zuzuteilen gedenkt. Bis dieser sie offiziell angenommen und bestätigt hat, bleibt er aber bei den vagen Adressen aus dem vorigen Punkt.
- DHCP-Server verwenden die UDP-Portnummer 67, DHCP-Clients die Portnummer 68.

  Eigentlich würde für den Client wie üblich eine zufällig gewählte (ansonsten unbenutzte) Portnummer reichen. Allerdings ist es laut Protokoll erlaubt, wenn auch nicht besonders gebräuchlich, dass der Server seine Antworten als IP-Broadcast verschickt – und Broadcasts an zufällig gewählte Portnummern sind nicht wirklich die feine englische Art.

- Im `DHCPACK` stehen in der Regel dieselben Parameter wie im `DHCPOFFER`. Die im `DHCPACK` zählen aber als amtlich.

Mehrere Angebote?

Grundsätzlich kann ein DHCP-Client auf sein `DHCPDISCOVER` von mehreren DHCP-Servern `DHCPOFFER`-Antworten erhalten. In diesem Fall kann er sich das Angebot aussuchen, das ihm am besten zusagt (zum Beispiel das erste eingetroffene) und auf dessen Basis mit `DHCPREQUEST` fortfahren. Auch die `DHCPREQUEST`-Nachrichten werden per Broadcast verbreitet, so dass die nicht ausgewählten Server mitbekommen können, dass ihre Dienste nicht von Interesse sind.

Die `DHCPOFFER`- und `DHCPREQUEST`-Nachrichten enthalten eine »Server ID«, die angibt, von welchem Server ein Angebot kommt bzw. wessen Angebot angenommen wird. Auch das soll Verwirrung vermeiden helfen.

Nach dem Empfang des `DHCPREQUEST` kann der DHCP-Server die Zuordnung der IP-Adresse an den Client vermerken und mit `DHCPACK` bestätigen. Das `DHCPACK` ist das Signal für den Client, die übergebenen Netzparameter tatsächlich in Kraft zu setzen.

1.2.2 Client-Konfiguration mit gültiger Adresse

Wenn ein DHCP-Client beim Neustart oder dem Betreten eines neuen Netzes noch über potentiell passende Konfigurationsparameter verfügt, deren Gültigkeitsdauer noch nicht überschritten ist, kann er versuchen, sich die Gültigkeit dieser Konfiguration von einem DHCP-Server bestätigen zu lassen. Dazu schickt er eine `DHCPREQUEST`-Nachricht mit der betreffenden Adresse an alle (!) DHCP-Server – per Broadcast, denn es könnte sein, dass die Netzkonfiguration *nicht* zum lokalen Netz passt und der ursprüngliche DHCP-Server so nicht direkt erreichbar ist.

Gültigkeit prüfen

Antwort des Servers

Der DHCP-Server prüft anhand seiner Datenbank, ob die angefragte IP-Adresse des Clients vernünftig aussieht, also zum Beispiel aus dem richtigen IP-Subnetz stammt. In diesem Fall antwortet er mit einer `DHCPACK`-Nachricht, die die IP-Adresse und andere aktuelle Konfigurationsparameter enthält.

Wenn Sie aufgepasst haben, dann ist Ihnen sicherlich aufgefallen, dass diese Vorgehensweise genau der aus dem vorigen Abschnitt entspricht, bis darauf, dass `DHCPDISCOVER` und `DHCPOFFER` wegfallen. Das ist auch ganz vernünftig

so, denn der Client hat seine Netzparameter ja irgendwann schon einmal akzeptiert; es geht nur um die Neubestätigung.

Ist die IP-Adresse dagegen ungültig (etwa weil sie aus einem anderen Netz eingeschleppt oder, bei dynamischer Vergabe, in der Zwischenzeit einem anderen Client zugeteilt wurde), schickt der DHCP-Server eine DHCPNAK-Nachricht. Für den DHCP-Client ist das das Signal, wie in Abschnitt 1.2.1 erklärt mit einem frischen DHCPDISCOVER anzufangen.

keine Antwort

Wenn der DHCP-Client auf seine Anfrage gar keine Antwort erhält, kann er die IP-Adresse selbstverständlich weiter verwenden, bis ihre Gültigkeitsdauer abgelaufen ist. (Wenn er nicht mehr im selben Netz ist – Stichwort: WLAN –, dann bringt ihm das natürlich präzise überhaupt nichts.)

1.2.3 Gültigkeitsdauer verlängern

Wir haben schon erwähnt, dass DHCP-Server ihre Konfigurationsparameter »auf Zeit« vergeben. Lang laufende Clients kommen also früher oder später an einen Punkt, wo ihre Netzkonfiguration sich dem Ende ihrer Gültigkeitsdauer nähert. Dann ist es Zeit für eine Verlängerung. Dazu schickt der DHCP-Client einen DHCPREQUEST an »seinen« Server – diesmal direkt und nicht per Broadcast. Dieser antwortet wie im vorigen Abschnitt diskutiert mit DHCPACK oder DHCPNAK. Ist der DHCP-Server nicht zu erreichen, dann versucht der Client es später per Broadcast.

Der Client darf sich in seiner Anfrage eine neue Gültigkeitsdauer wünschen, aber es ist dem Server überlassen, ob er darauf eingeht oder nicht. Als Administrator können Sie natürlich steuern, welche Fristen Ihr Netz verwendet (hierzu später mehr); die ausschlaggebenden Parameter im Protokoll sind:

IP Address Lease Time (Gültigkeitsdauer) Gibt an, wie lange die Netzparameter verwendet werden können.

Renewal Time Value Gibt an, ab welchem Zeitpunkt der Client sich mit einer direkten Anfrage um eine Verlängerung bemüht (Standardwert: Hälfte der Gültigkeitsdauer)

Rebinding Time Value Gibt an, ab welchem Zeitpunkt der Client zur Verlängerung per Broadcast Kontakt zu einem Server sucht (Standardwert: 7/8 der Gültigkeitsdauer)

Ablauf der Gültigkeit

Wenn der Client vor Ablauf der Gültigkeit keinen Server für eine Verlängerung erreichen kann, muss er alle aktiven Verbindungen schließen und von vorne beginnen wie in Abschnitt 1.2.1.

1.2.4 Abmelden

Ein DHCP-Client kann schon vor Ablauf der Gültigkeitsdauer mit einer DHCPRELEASE-Nachricht seine Konfiguration an den Server zurückgeben. Nach dem DHCPRELEASE stellt er die Verwendung der betreffenden IP-Adresse sofort ein.

Das DHCPRELEASE wird vom Server nicht bestätigt, da der Client eine Bestätigung mangels IP-Adresse gar nicht empfangen könnte.

1.2.5 Parameter-Wunschlisten

Außer den unmittelbar für die Netzkonfiguration nötigen Parametern wie der IP-Adresse, Netzmaske oder dem Standard-Gateway können per DHCP auch noch viele weitere Einstellungen übertragen werden. Im DHCPDISCOVER und DHCPREQUEST kann der Client eine Parameter-Wunschliste (engl. *parameter request list*) angeben, die dem Server signalisiert, für welche Informationen der Client sich interessiert. Im DHCPOFFER und DHCPACK liefert der Server dann Parameterlisten, die

```
Option: (t=55,l=12) Parameter Request List
    1 = Subnet Mask
    28 = Broadcast Address
    2 = Time Offset
    3 = Router
    15 = Domain Name
    6 = Domain Name Server
    119 = Domain Search
    12 = Host Name
    44 = NetBIOS over TCP/IP Name Server
    47 = NetBIOS over TCP/IP Scope
    26 = Interface MTU
    121 = Classless Static Route
```

Bild 1.2: Parameter-Wunschliste eines DHCP-Clients (dekodiert mit `tshark`)

```
Option: (t=51,l=4) IP Address Lease Time = 3 hours
Option: (t=58,l=4) Renewal Time Value = 1 hour, 30 minutes
Option: (t=59,l=4) Rebinding Time Value = 2 hours, 37 minutes,\LB
                                                \BL 30 seconds
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=28,l=4) Broadcast Address = 192.168.56.255
Option: (t=3,l=4) Router = 192.168.56.253
Option: (t=6,l=4) Domain Name Server = 192.168.56.253
```

Bild 1.3: Parameter-Liste eines DHCP-Servers (dekodiert mit `tshark`)

(hoffentlich) die Wünsche des Clients befriedigen – der Server ist genauso frei, Anforderungen des Clients zu ignorieren, wie der Client die vom Server geschickten Parameter überlesen oder ändern kann. (Er tut das natürlich auf eigene Gefahr.)

Bild 1.2 zeigt die Wunschliste eines typischen Linux-Clients und Bild 1.3 die Antwort des DHCP-Servers darauf. In diesem Fall ist die Liste des Servers eine Teilmenge der Liste des Clients, aber das muss nicht so sein; der Server könnte von sich aus noch andere Parameter hinzufügen, die der Client dann verwerten kann oder nicht. In der Konfiguration eines DHCP-Servers können Sie in der Regel angeben, welche Parameter der Server liefert, und meistens sogar für verschiedene Clients oder Gruppen von Clients unterschiedliche Parametersätze vorhalten.

Übungen

1.2 [!3] Beobachten Sie mit einem geeigneten Netzmonitor (`tcpdump` oder `wireshark`), wie ein DHCP-Client mit einem DHCP-Server kommuniziert. Identifizieren Sie die `DHCPDISCOVER`-, `DHCPOFFER`-, `DHCPREQUEST`- und `DHCPACK`-Datagramme. Welche Optionen werden angefordert, welche übergeben?

1.3 [2] Wenn ein DHCP-Client beim DHCP-Server Netzparameter anfordert, passieren dafür – zumindest bei dynamischer Adressvergabe – außer dem reinen DHCP-Dialog mitunter noch ein paar andere Dinge auf dem Netz. Welche und warum? (Wenn Ihr DHCP-Server IP-Adressen statisch vergibt, dann arbeiten Sie erst den Rest des Kapitels durch und beantworten die Frage dann anhand einer Testumgebung.)

1.3 DHCP-Clients

1.3.1 Überblick

Ein Linux-Rechner kann als DHCP-Client auftreten, aber diese Funktionalität ist nicht im Linux-Kernel enthalten. Statt dessen wird sie von einem externen Programm erbracht. Hierfür gibt es verschiedene, weitgehend gleichwertige Möglichkeiten; welche davon auf Ihrem System zum Einsatz kommt, hängt hauptsächlich von der Distribution ab.

Die verbreitetsten DHCP-Client-Programme sind `dhclient` aus der DHCP-Implementierung des *Internet Systems Consortium* (ISC), der »DHCP-Client-Daemon« `dhcpcd` und `pump` von Red Hat.


Die Debian-GNU/Linux- und Ubuntu-Distributionen verwenden standardmäßig `dhclient`; `dhcpcd` und `pump` (und noch einige andere weniger verbreitete Programme) können alternativ installiert werden.

Bei den Novell/SUSE-Distributionen kommt der `dhcpcd` zum Einsatz. Der ISC-`dhclient` und einige andere Alternativen (nicht `pump`) stehen zur Installation zur Verfügung.


Die Red-Hat-Distributionen verwenden ebenfalls `dhclient`. Auch hier sind Alternativen verfügbar.

1.4 Der ISC-DHCP-Server

1.4.1 Überblick

Das vielleicht gebräuchlichste DHCP-Softwarepaket unter Linux wird vom *Internet Systems Consortium* (ISC) als freie Software zur Verfügung gestellt. Verwirrenderweise heißt es auch »DHCP«. Das Paket besteht aus DHCP-Serversoftware (`dhcpd`), DHCP-Clientsoftware (`dhclient`) und einem DHCP-Relay-Agent (`dhcrelay`).


Einen DHCP-Relay-Agent brauchen Sie, wenn Sie DHCP-Clients in einem physikalischen Netz betreiben wollen, in dem kein kompletter DHCP-Server zur Verfügung steht (etwa weil Sie die ganze Server-Konfiguration auf einem Rechner bündeln möchten). DHCP wie in Abschnitt 1.2 funktioniert dann nicht, weil IP- und Ethernet-Broadcasts nicht außerhalb des physikalischen Netzes empfangen werden können. Der Relay-Agent lauscht auf DHCP-Anfragen und leitet sie gezielt an den DHCP-Server (in einem anderen Netz) weiter.

DHCP-Relay-Agent

Das ISC-DHCP-Paket ist bei den großen Linux-Distributionen nicht Bestandteil der Standardinstallation, aber kann leicht nachgerüstet werden:


Bei Debian GNU/Linux und Ubuntu müssen Sie dazu das Paket `isc-dhcp-server` installieren (das ebenfalls nötige `isc-dhcp-common` ist in der Regel schon vorhanden, weil der `isc-dhcp-client`, ein Paket mit der Priorität `important`, es braucht). Die Konfigurationsdatei `dhcpd.conf` für den Server befindet sich im Verzeichnis `/etc/dhcp3`. Nach der Paketinstallation versucht das System, den DHCP-Server zu starten, aber das geht mit großer Sicherheit schief, weil Sie erst die Konfiguration anpassen müssen. – Der Server heißt bei Debian GNU/Linux und Ubuntu übrigens `dhcpd3`, nicht `dhcpd` (und die Handbuchseite entsprechend `dhcpd3`(8)); dies, weil `dhcpd` sich auf die frühere Version 2 bezog und man eine Migration erleichtern wollte.


Das `dhcp`-Paket von Red Hat enthält sowohl den DHCP-Server als auch den DHCP-Relay-Agent (der bei Debian GNU/Linux und Ubuntu ein separates Paket darstellt). Nach der Installation des Pakets ist der DHCP-Server

```
ddns-update-style none;
ddns-updates off;
authoritative;

option domain-name "example.com";
option domain-name-servers 192.168.56.253, 192.168.56.254;

subnet 192.168.56.0 netmask 255.255.255.0 {
  range 192.168.56.1 192.168.56.127;
  option routers 192.168.56.254;
}
```

Bild 1.4: Eine einfache `dhcpd.conf`-Datei

wie bei Debian GNU/Linux und Ubuntu zunächst unkonfiguriert und läßt sich nicht starten, außerdem muss er explizit für den Systemstart aktiviert werden. Die Konfigurationsdatei für den Server ist `/etc/dhcp/dhcpd.conf`.

Bei openSUSE steht der ISC-DHCP-Server im Paket `dhcp-server`, das Sie bei Bedarf manuell nachinstallieren müssen. Vor der Verwendung müssen Sie in der Datei `/etc/sysconfig/dhcpd` die Schnittstelle(n) eintragen, um die der DHCP-Server sich kümmern soll, und `SuSEconfig` aufrufen; dafür gibt es auch ein YaST-Modul. Auch hier muss der DHCP-Server für den Systemstart explizit aktiviert werden. Konfigurieren können Sie ihn in der Datei `/etc/dhcpd.conf`.

Grundsätzlich können Sie natürlich auch die aktuelle Version von `http://www.isc.org/` herunterladen und installieren. Wir betrachten das aber nicht weiter.

1.4.2 Die Konfigurationsdatei `dhcpd.conf`

Die Konfigurationsdatei für den ISC-DHCP-Server heißt `dhcpd.conf` (auch wenn die Distributionen sich anscheinend nicht einigen können, wo sie abgelegt werden soll). Wie üblich handelt es sich dabei um eine lesbare Text-Datei, die diverse Deklarationen und Parameterdefinitionen enthält. Ihr Aufbau ist genauer in `dhcpd.conf`(5) dokumentiert. Bild 1.4 zeigt ein Beispiel für eine einfache `dhcpd.conf`-Datei.

Deklarationen

Deklarationen in der Konfigurationsdatei beschreiben die Topologie des zu verwaltenden Netzes, die Clients in diesem Netz oder die zu verwendenden IP-Adressen. In Bild 1.4 taucht zum Beispiel die `subnet`-Deklaration auf, die Definitionen für das Netz `192.168.56.0/24` enthält. Parameter dagegen regeln die Details, also ob und wie bestimmte Dinge getan werden sollen oder (zum Beispiel) welche Optionen an einen Client übergeben werden sollen – in Bild 1.4 etwa die Zeilen mit `ddns-updates` oder `option`. Es ist möglich, eine Gruppe von Parametern nur für eine bestimmte Gruppe von Deklarationen zu aktivieren.

Parameter

Fest oder dynamisch? Der DHCP-Server kann entweder versuchen, bestimmten Stationen bestimmte IP-Adressen zuzuordnen (etwa auf der Basis von deren MAC-Adresse), oder IP-Adressen dynamisch aus einem dafür bestimmten Vorrat verteilen. Beide Ansätze haben ihre Vor- und Nachteile: Die feste Vergabe erfordert einen höheren Konfigurationsaufwand (Sie müssen eine Tabelle von MAC-Adressen und dazugehörigen IP-Adressen warten), aber bedeutet einen gewissen Zugewinn an Sicherheit, da »unbekannte« Stationen keine Netzkonfiguration erhalten. Die dynamische Vergabe ist flexibler, aber weniger resistent gegen ungebetene »Zaungäste«.

Auch feste Adressenvergabe ist keine Garantie gegen böse Buben (und Mädchen), die fremde Rechner in Ihr Netz einschmuggeln wollen – die können notfalls ja auch Netzparameter raten (wobei sie sich von einem legitimen Rechner in der Nähe inspirieren lassen können). Um so etwas zu erschweren, müssen Sie die Latte etwas höher legen und zum Beispiel einen Switch verwenden, der Ports mit unbekannten MAC-Adressen am anderen Ende des Kabels sperrt (»Port Security«). Natürlich sollten Sie dann auch keine unbenutzten Ethernet-Dosen gepatcht lassen und lautstark beim Administrator Krach schlagen, wenn erfundene IP-Adressen in Ihrem Netz kursieren.

dynamische Vergabe

Ein Beispiel für die dynamische Vergabe sehen Sie in Bild 1.4: Der `range`-Parameter in der `subnet`-Deklaration gibt an, dass in dem betreffenden Netz die Adressen `192.168.56.1` bis `192.168.56.127` nach Bedarf an Clients verteilt werden können.

Auch bei dynamischer Vergabe versucht der ISC-DHCP-Server, Clients nach Möglichkeit immer dieselbe IP-Adresse zuzuordnen (wiederum auf der Basis der MAC-Adresse). Wenn mehr Clients im Umlauf sind, als der Adressenvorrat Adressen enthält, ist das natürlich nicht konsequent einhaltbar, aber dank [RFC1918] sollte das eigentlich nicht vorkommen (müssen).

feste Vergabe
`host`

Für die feste Vergabe müssen Sie eine Zuordnung von Stationen zu IP-Adressen schaffen. Dazu brauchen Sie `host`-Deklarationen, etwa wie folgt:

```
host blue {
  hardware ethernet 08:00:27:92:44:0c;
  fixed-address 192.168.56.129;
}
```

Diese Deklaration sorgt dafür, dass der Rechner mit der MAC-Adresse `08:00:27:92:44:0c` fest die IP-Adresse `192.168.56.129` zugeordnet bekommt.

Der Client bekommt damit eine feste *Adresse*, aber nicht notwendigerweise auch einen festen *Namen*, da DHCP sich nicht direkt um Rechnernamen kümmert. Ein naheliegender Ansatz ist, der Adresse im DNS einen festen Namen zuzuordnen, unter dem der Client dann im Netz erreichbar ist (ob er selber sich unter dem Namen angesprochen fühlt, ist eine andere Frage), was die Konfiguration für diesen Client natürlich über DHCP- und DNS-Server verstreut. Alternativ kann der DHCP-Server dem DNS-Server Bescheid sagen, wenn der Client eine IP-Adresse bekommt, und dessen Namen dann im DNS zugänglich machen (Stichwort »dynamische DNS-Aktualisierung«). Ob der Client sich seinen eigenen Namen wünschen darf (kritisch bei Sachen wie `www`) oder der DHCP-Server eine feste Vorgabe macht, ist dann wiederum Sache der Server-Konfiguration.

`host`-Deklarationen müssen nicht notwendigerweise eine feste IP-Adresse vorgeben, sondern können auch ganz allgemein dazu dienen, einem einzelnen Rechner bestimmte Parameter zuzuordnen.

Der Server sucht nach einer `host`-Deklaration für einen Client zunächst auf der Basis eines vom Client gelieferten *client identifier* (in der `host`-Deklaration über den Parameter `dhcp-client-identifier` anzugeben). Nur wenn es keinen solchen gibt oder keine dazu passende `host`-Deklaration gefunden werden kann, wird auf die MAC-Adresse zurückgegriffen.

`dhcp-client-identifier`

Topologie In der DHCP-Server-Konfiguration muss es eine `subnet`-Deklaration für jedes Netz geben, an das der Server angeschlossen ist – unabhängig davon, ob er tatsächlich Adressen für das betreffende Netz verteilt oder nicht. Dies ist notwendig, damit er erkennen kann, in welchem Subnetz sich eine Adresse befindet. Zu einer `subnet`-Deklaration gehören eine Netzadresse und eine Netzmaske:

`subnet`

```
subnet 192.168.56.0 netmask 255.255.255.0 {
  ⟨Parameter⟩
  ⟨Deklarationen⟩
}
```

Die innerhalb der `subnet`-Deklaration angegebenen Parameter und Deklarationen gelten nur für Stationen innerhalb des betreffenden Netzes – Parameter und Deklarationen außerhalb gelten für die komplette Konfiguration. Die naheliegende auf ein bestimmtes Netz bezogene Einstellung ist die einer `routers`-Option für die Clients (siehe Bild 1.4), aber auch andere Parameter und Optionen kommen in Frage.

`shared-network` Mit der `shared-network`-Deklaration lassen sich mehrere `subnet`-Deklarationen zusammenfassen, wenn deren Subnetze sich zum Beispiel auf demselben physikalischen Netz befinden.

Das Beispiel dafür in `dhcpd.conf`(5) zitiert den Fall, dass es in einem Unternehmen eine lokale Vorgabe dafür gibt, Netze mit Adressen der Form *x.y.z*`.0/24` zu verwenden. Wenn eine Abteilung mit einem einzigen physikalischen Ethernet dann so weit wächst, dass sie mehr als 254 Stationen benötigt, kann es nötig sein, auf demselben Ethernet zwei logische »Klasse-C«-Subnetze zu betreiben, bis ein weiteres physikalisches Ethernet hinzugefügt werden kann. In diesem Fall ist eine Deklaration der Form

```
shared-network bignet {
  subnet 192.168.56.0 netmask 255.255.255.0 { ... }
  subnet 192.168.57.0 netmask 255.255.255.0 { ... }
}
```

erforderlich.

Nicht wirklich mit der Netztopologie verbunden, aber dennoch erwähnens-
`group` wert ist die `group`-Deklaration, mit der Sie verschiedene Stationen zusammenfassen können, um ihnen Parameter zuzuordnen, auch wenn sie sich nicht notwendigerweise im selben Netz befinden.

`pool` Mit der `pool`-Deklaration können Sie unterschiedliche *Adressen* unterschiedlich behandeln, selbst wenn sie sich im selben physikalischen Netz oder `subnet` befinden. Sie können zum Beispiel »bekannten« Stationen – solchen, die in Ihrer Konfiguration einen `host`-Eintrag (ohne feste Adresse) haben – andere dynamische Adressen zuteilen als »unbekannten«:

```
host blue { hardware ethernet 08:00:27:92:44:0c; }
host green { hardware ethernet 08:00:27:a0:98:4c; }

subnet 192.168.56.0 netmask 255.255.255.0 {
  option routers 192.168.56.254;

  pool {
    range 192.168.56.1 192.168.56.63;
    allow unknown-clients;
  }
  pool {
    range 192.168.56.65 192.168.56.127;
    deny unknown-clients;
  }
}
```

Durch eine geeignete Firewall-Konfiguration können Sie dann zum Beispiel den Adressen für »bekannte« Stationen Zugriff auf das Internet gewähren und den anderen nicht.


Die `allow`- und `deny`-Parameter in `pool` erlauben auch noch feinkörnigere Unterscheidung, aber das würde hier zu weit führen. Lesen Sie den Abschnitt »Client Classing« in `dhcpd.conf`(5).

Autorität Eine wichtige Frage für einen DHCP-Server ist die nach seiner »Autorität«, salopp gesagt ob er die »Alleinherrschaft« über ein Netz hat oder ob Clients auch DHCP-Daten aus anderer Quelle erhalten können. Dies äußert sich vor allem in seiner Reaktion auf `DHCPREQUEST`-Nachrichten, die Clients zum Beispiel zur Rückbestätigung ihrer schon vorhandenen Netzparameter absetzen (Abschnitt 1.2.2). Meldet sich ein Client mit einer IP-Adresse, für die der DHCP-Server sich nicht unmittelbar zuständig fühlt, beantwortet er als autoritativer Server den `DHCPREQUEST` mit einem `DHCPNAK`, während er als nicht autoritativer Server gar nichts tut (es ist nicht seine Baustelle).


Unter normalen Umständen, also wenn Ihr DHCP-Server der einzige im Netz ist oder sein sollte, sollten Sie Ihren DHCP-Server zum autoritativen Server erklären, um lange Wartezeiten für Clients zu vermeiden, die sich sonst bis zum letztmöglichen Moment an ihre eigentlich ungültigen Netzparameter klammern. Die Standardeinstellung beim ISC-DHCP-Server ist übrigens »nicht autoritativ«!


Die ganze Miete ist das noch nicht: Für ein `DHCPNAK` eines autoritativen Servers ist nicht nur erforderlich, dass die angefragte Adresse einem Subnetz dieses Servers entstammt, sondern auch spezifisch in seiner Konfiguration erwähnt ist, etwa indem sie einem Rechner fest zugeordnet wurde oder zu einem für die dynamische Vergabe vorgesehenen Adressbereich gehört.

Zur Einstellung der Autorität dienen die Parameter »`authoritative`« oder »`not authoritative`«. Diese können entweder auf der globalen Ebene oder innerhalb von Subnetzen, Gruppen usw. auftreten.


Sie können die Autorität auch für verschiedene Subnetze unterschiedlich angeben. In Subnetzen, die von einem anderen DHCP-Server bedient werden, sollte Ihrer natürlich nicht autoritativ sein.

Interessante Parameter Der ISC-DHCP-Server unterstützt einen reichhaltigen Zoo von Konfigurationsparametern, von denen wir hier nur die wichtigsten und interessantesten vorstellen können. Die volle Liste finden Sie in `dhcpd.conf`(5) und `dhcp-options`(5).

Hier sind einige Parameter, die sich mit der Gültigkeit von Konfigurationsdaten befassen: Gültigkeit

`default-lease-time` ⟨*Dauer*⟩ bestimmt die Gültigkeitsdauer der zugewiesenen Konfigurationsdaten in Sekunden, sofern der Client keine eigenen Wünsche äußert.

`max-lease-time` ⟨*Dauer*⟩ bestimmt die maximale Zeit in Sekunden, die der Server als Gültigkeitsdauer zuläßt. Wenn ein Client sich eine längere Gültigkeitsdauer wünscht, wird der durch diesen Parameter gegebene Wert eingesetzt.

`min-lease-time` ⟨*Dauer*⟩ bestimmt analog die minimale Zeit in Sekunden, die der Server als Gültigkeitsdauer zuläßt.

DHCP-Server und -Clients können dynamische Änderungen des DNS veranlassen. In diesem Kapitel können wir das nicht im Detail erklären, aber einige wichtige Parameter in diesem Zusammenhang müssen wir trotzdem erwähnen:

`ddns-update-style none|ad-hoc|interim` bestimmt die Methode, die für die dynamische Aktualisierung verwendet wird. `none` bedeutet, dass keine Aktualisierung gemacht werden soll, und ist der sichere Wert für den Fall, dass Sie diese Funktion nicht verwenden wollen. `interim` ist für den Fall nötig, dass Sie sich daran versuchen möchten.

`ad-hoc` ist aus dem DHCP-Mesozoikum übriggeblieben, funktioniert überhaupt nicht, und Sie sollten darum einen weiten Bogen machen.

Sie bekommen genau eine Chance, den `ddns-update-style` zu setzen, nämlich auf der globalen Ebene der Konfigurationsdatei. Diese Einstellung gilt dann, solange der Server läuft, für alle Clients.

`ddns-updates on|off` Bestimmt, ob der Server versucht, bei einem `DHCPACK` das DNS zu aktualisieren. Der Standardwert ist (etwas lästigerweise) `on`.

Mit `ddns-updates` können Sie Aktualisierungen nicht nur auf der globalen Ebene, sondern auch in untergeordneten Deklarationen wie `subnet` oder `group` ein- oder ausschalten.

Wenn Sie überhaupt keine Aktualisierungen wollen, sollten Sie sie auf der globalen Ebene mit »`ddns-update-style none`« ausschalten.

`ddns-hostname` ⟨*Name*⟩ könnte in einer `host`-Deklaration auftreten und bestimmt den Namen, der für den betreffenden Client im DNS installiert wird. Ohne so einen Parameter überlegt sich der DHCP-Server selber einen Namen.

`allow client-updates` bedeutet, dass der Client sich selbst einen Namen wünschen kann. »`deny client-updates`« verbietet das. »`ignore client-updates`« signalisiert dem Client, dass er sich selbst um seine Aktualisierung kümmern soll (ob der DNS-Server Aktualisierungen von irgendwelchen Clients annimmt, ist eine andere Frage).

`use-host-decl-names` sorgt dafür, dass anstelle von `ddns-hostname`-Parametern der Rechnername aus der `host`-Deklaration (der, der zwischen dem Wort `host` und der öffnenden geschweiften Klammer steht) verwendet wird. (Der Parameter hat auch noch ein paar andere Konsequenzen.)

Hier sind noch ein paar andere interessante Parameter:

`log-facility` ⟨*Syslog-Kategorie*⟩ Hiermit können Sie angeben, welche Kategorie (siehe `syslog.conf`(5)) der DHCP-Server benutzt, wenn er Meldungen an den Protokolldienst schickt. Standard ist `daemon`.

Fehler vor dem oder beim Lesen von `dhcpd.conf` können nicht mit der hier angegebenen Kategorie protokolliert werden, da der `log-facility`-Parameter zu diesem Zeitpunkt noch nicht angeschaut wurde. Statt dessen wird die Standardvorgabe (`daemon`) verwendet. Wenn Sie das ändern wollen, müssen Sie den `dhcpd` neu übersetzen.

Mehr Informationen über den Protokolldienst finden Sie in der Linup-Front-Schulungsunterlage *Linux-Administration II*.

`option` ⟨*Name*⟩ ⟨*Wert*⟩ gibt eine DHCP-Option an, die an Clients übertragen werden soll (siehe `dhcp-options`(5)). Hier sind ein paar Beispiele:

```
option broadcast-address 192.168.66.255;
option dhcp-client-identifier "foobar";
option domain-name example.com;
option domain-name-servers 192.168.66.254, 192.168.65.254;
option domain-search "marketing.example.com", "example.com";
option interface-mtu 1400;
option netbios-name-servers 192.168.33.1, 192.168.34.17;
option nntp-server news.example.com;
option ntp-servers ntp0.example.com, ntp1.example.com;
option routers 192.168.66.254;
```


Die Optionen sind auch für DHCP-Clients von Bedeutung, die sich bestimmte Optionswerte vom Server wünschen dürfen. In diesem Fall müssen Sie sie natürlich auf dem Client konfigurieren.


Dass der Server dem Client eine Option schickt, heißt noch lange nicht, dass der Client deren Wert tatsächlich konkret umsetzt. Die Option `host-name` zum Beispiel wird von Clients gerne missachtet, die ihre eigenen Vorstellungen davon haben, wie sie heißen wollen.


Sie können vielen dieser Optionen Werte geben, die nur für bestimmte Subnetze oder einzelne Stationen gelten. Dies geht einerseits über geeignete Deklarationen wie `host` oder `subnet`, aber auch durch »berechnete« Optionswerte. Die Details darüber stehen in `dhcp-eval`(5).

Übungen


1.4 [!3] Installieren Sie einen DHCP-Server, der dynamisch IP-Adressen aus einem gewissen Vorrat vergibt. Überzeugen Sie sich, dass das korrekt funktioniert. (Hier ist es nützlich, eine Virtualisierungsumgebung einzusetzen, die »private Vernetzung« zuläßt, damit Sie nicht etwa andere Benutzer Ihres physikalischen Netzes stören.)


1.5 [!2] (Fortsetzung der vorigen Aufgabe.) Vergewissern Sie sich, dass die feste Adressenvergabe mit `host` funktioniert.


1.6 [2] Beobachten Sie, wie ein DHCP-Client die Gültigkeitsdauer seiner Netzparameter verlängert, zum Beispiel indem Sie die Parameter `default-lease-time` und `max-lease-time` niedrig genug setzen (zum Beispiel 60 Sekunden). Wann nimmt der Client die Verlängerung in Angriff?

1.5 dnsmasq

Das Programm `dnsmasq` von Simon Kelley ist eine Kombination aus DNS- und DHCP-Server, die sich vor allem für kleine Netze eignet, wo der Einsatz zum Beispiel von BIND und ISC-DHCP-Server einen übergroßen Aufwand darstellen würde.


Für `dnsmasq` müssen Sie keine DNS-Zonendateien anlegen, sondern der Inhalt von `/etc/hosts` wird per DNS zur Verfügung gestellt; wenn Sie weitere (ungewöhnliche) Ressourceneinträge benötigen, können Sie diese in die Konfigurationsdatei von `dnsmasq` eintragen. Details über die DNS-Funktionalität von `dnsmasq` finden Sie zum Beispiel in der Linup-Front-Schulungsunterlage *Linux-Netzwerkadministration*.

Natürlich bietet `dnsmasq` nicht alle Eigenschaften der anderen Programme; der deutlich geringere Installations- und Wartungsaufwand macht das aber in vielen Fällen wett.


Als DHCP-Server empfiehlt sich `dnsmasq` zum Beispiel, wenn Sie einen vorgefertigten Router (etwa eine FRITZ!box) zum Internetzugang nutzen wollen, aber mehr DHCP-Funktionalität benötigen, als die typischerweise doch rudimentären DHCP-Server der Router bieten – etwa feste Vergabe von IP-Adressen auf der Basis der MAC-Adressen. Sie können `dnsmasq` auf einem permanent laufenden PC in Ihrem Netz (zum Beispiel dem Dateiserver) oder einem sehr stromsparenden Gerät wie der Linksys NSLU2 laufen lassen und statt des DHCP-Servers im Router verwenden.


`dnsmasq` eignet sich auch sehr gut für Linux-basierte Router mit Betriebssystemen wie OpenWRT.

1.6 DHCP-Sicherheit

Wir wollen das Thema »DHCP« nicht verlassen, ohne noch ein paar Worte über DHCP-Sicherheit gesagt zu haben. Bitte beachten Sie das Folgende:

- Wie viele andere »traditionelle« Internet-Protokolle stammt DHCP aus einer Zeit, als Sicherheit noch kein besonders wichtiges Thema war. Entsprechend unterentwickelt sind die Maßnahmen, die DHCP anbietet, um Problemen aus dem Weg zu gehen.
- Normalerweise müssen Clients sich nicht beim DHCP-Server authentisieren – er verteilt entweder großzügig IP-Adressen aus einem dynamischen Vorrat oder prüft nur leicht auf der Clientseite fälschbare Daten wie den Client-Identifier oder die MAC-Adresse. Es ist also nicht wirklich schwierig, einen unautorisierten DHCP-Client in ein Netz einzuschleusen und sich gültige Netzparameter zu verschaffen. (Die möglichen Gegenmaßnahmen entstammen eher dem Bereich »Physikalische Sicherheit« und wurden schon weiter oben angedeutet.)
- Genausowenig müssen DHCP-Server sich bei ihren Clients authentisieren, die in der Regel den ersten halbwegs plausibel aussehenden Satz von Netzparametern in einer DHCP-Antwort akzeptieren und in Kraft setzen. Tatsächlich beruht DHCP, wie gezeigt, auf Broadcast, so dass Clients sich nicht einmal gezielt an einen Server mit einer bekannten Adresse wenden können[1]. Wenn Sie einen Rechner im lokalen Netz kontrollieren, ist es kein fundamentales Problem, einen unautorisierten DHCP-Server zu starten und Clients Netzparameter zuzuschanzen, die »interessante« Effekte haben können.

  
  Zum Beispiel könnten Sie per »`option domain-name-servers`« andere Rechner auf einen DNS-Server verweisen, der für bestimmte Namen (Banken?) gefälschte IP-Adressen ausliefert und die Benutzer dieser Rechner auf diese Weise anfällig für Phishing-Angriffe macht.
- Bei dynamischer Adressvergabe ist es möglich, dass Clients große Mengen von Adressen anfordern und so den Adressenvorrat ausschöpfen. Andere Clients können dann keine Adresse mehr bekommen. Hierbei handelt es sich um einen Dienstverweigerungs-Angriff (engl. *denial of service attack*).


[RFC3118] definiert ein Authentisierungsverfahren für DHCP-Nachrichten, das allerdings noch nicht von allen verbreiteten DHCP-Client- und -Serverprogrammen umgesetzt wird. Insbesondere der ISC-DHCP-Server kümmert sich noch nicht darum, so dass eine weitere Diskussion an dieser Stelle überflüssig scheint.

Kommandos in diesem Kapitel

`dhclient` Client für DHCP, konfiguriert einen Rechner, vom ISC `dhclient`(8) 19
`dhcpd` Server für DHCP, vom ISC `dhcpd`(8) 19
`dhcrelay` Relay-Agent für DHCP (reicht DHCP zwischen Netzen weiter) `dhcrelay`(8) 19
`dnsmasq` Ein einfacher DHCP- und cachender DNS-Server für kleine Installationen `dnsmasq`(8) 25

[1] Es gäbe da auch gewisse Henne-Ei-Probleme.

Zusammenfassung

- Das »Dynamic Host Configuration Protocol« (DHCP) dient zur zentralisierten Verwaltung von Netzkonfigurationsparametern.
- Einer der gebräuchlichsten DHCP-Server für Linux ist der ISC-DHCP-Server, der bei den meisten Distributionen Bestandteil des Paketrepertoires ist.
- Die Konfigurationsdatei für den ISC-DHCP-Server heißt `dhcpd.conf`.
- Die Konfigurationsdatei enthält Deklarationen (für Netztopologie, Rechner und ähnliches) und Parameter (mit konkreten Werten).
- DHCP ist nicht besonders sicher.

Literaturverzeichnis

RFC0951 Bill Croft, John Gilmore. »Bootstrap Protocol (BOOTP)«, September 1985. http://www.ietf.org/rfc/rfc0951.txt

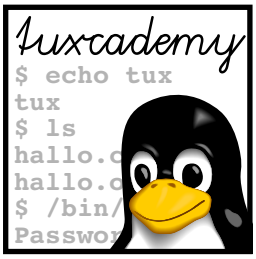
RFC1531 R. Droms. »Dynamic Host Configuration Protocol«, Oktober 1993. http://www.ietf.org/rfc/rfc1531.txt

RFC1918 Y. Rekhter, B. Moskowitz, D. Karrenberg, et al. »Address Allocation for Private Internets«, Februar 1996. http://www.ietf.org/rfc/rfc1918.txt

RFC2131 R. Droms. »Dynamic Host Configuration Protocol«, Mârz 1997. http://www.ietf.org/rfc/rfc2131.txt

RFC3118 R. Droms, W. Arbaugh. »Authentication for DHCP Messages«, Juni 2001. http://www.ietf.org/rfc/rfc3118.txt

Ste94 W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional Computing Series. Boston etc.: Addison-Wesley, 1994.



2

Authentisierung mit PAM

Inhalt

Lernziele

- Konzepte von PAM verstehen
- PAM konfigurieren können
- Die wichtigsten PAM-Module kennen und anwenden können
- Den »Name Service Switch« verstehen und konfigurieren können

Vorkenntnisse

- Kenntnisse der Linux-Systemadministration

2.1 Warum PAM?

Früher musste jedes Unix-Programm, das Benutzerauthentisierung benötigte – außer `login` auch Dienste wie TELNET und `rlogin`, FTP-Daemons, Bildschirmschoner und alles Mögliche andere –, die System-Datenbanken (`/etc/passwd` & Co.) direkt lesen. Das war natürlich furchtbar unflexibel; selbst für simple Änderungen wie die Einführung eines stärkeren Verschlüsselungsverfahrens für Kennwörter hätte man alle diese Programme finden und anpassen müssen, und daran, Benutzerdaten von anderswo zu holen als `/etc/passwd` (ein Thema, das mit der Einführung von NIS Mitte der 1980er Jahre akut wurde), war kaum zu denken, vor allem wenn man davon ausgeht, dass nicht alle Programme auf dem System tatsächlich im Quellcode vorlagen …

Der erste Schritt dazu, diese Situation zu verbessern, war die Einführung standardisierter Funktionen in der C-Bibliothek, die zum Beispiel die Benutzerdaten zu einem gegebenen Benutzerdaten oder einer gegebenen UID besorgen konnten (egal ob aus `/etc/passwd` oder von einem NIS-Server). Dies zusammen mit der Einführung von *sharable libraries* beschränkte das Problem zumindest darauf, »nur« die (dynamisch ladbare) C-Bibliothek umschreiben und auf dem System ersetzen zu müssen. Solche Änderungen kommen auch Programmen zugute, die mangels Quellcode nicht selbst neu übersetzt werden können. Allerdings reichte auch das natürlich noch nicht aus, um zum Beispiel verschiedene Programme mit unterschiedlichen Authentisierungsmethoden zu betreiben, etwa um Zugriffe über das Netz im Gegensatz zur lokalen Anmeldung nur bestimmten Benutzern zu erlauben – dafür waren immer noch die Anwendungsprogramme selbst zu ändern.

Pluggable Authentication Modules
PAM

Wirkliche Flexibilität bei der Konfiguration des Authentisierungsvorgangs erreichte man erst mit der Einführung der *Pluggable Authentication Modules*, kurz **PAM**. Die Idee hinter PAM ist, dass Anwendungsprogramme – etwa `login` oder der `ftpd` – zur Authentisierung von Benutzern nur noch auf die PAM-Bibliothek zugreifen, anstatt sich selbst darum zu kümmern. Die PAM-Bibliothek liefert dann (einfach gesagt) nur noch die Antwort »Zulassen« oder »Abweisen«; die Details des Anmeldevorgangs – also wo die Authentisierungsdaten herkommen, welche Anforderungen überhaupt aufgestellt werden und so weiter – werden innerhalb von PAM abgehandelt, ohne dass das aufrufende Programm sich darum kümmern muss. (Ein klassisches Beispiel für »Outsourcing«.)

PAM ist nicht zu verwechseln mit dem *Name Service Switch*, einer Infrastruktur, die innerhalb der C-Bibliothek für die Abstraktion von Quellen für die Benutzerdatenspeicherung (und ähnliches) dient. Die beiden arbeiten aber zusammen.

PAM kümmert sich um die Mechanik der Authentisierung, aber ist an ein paar Stellen auf die Mithilfe des aufrufenden Anwendungsprogramms angewiesen. Muss der Benutzer zum Beispiel ein Kennwort eingeben, dann sorgt PAM dafür, dass das aufrufende Programm es in angemessener Weise einfordert – etwa durch eine Texteingabe auf einem seriellen Terminal oder eine grafische Eingabe in einem Programm wie `xdm`.

PAM ist ursprünglich eine Erfindung von Sun Microsystems, wurde aber relativ früh in der Geschichte von Linux kompatibel nachprogrammiert.

Um mit PAM den Authentisierungsvorgang einer Anwendung zu ändern, müssen Sie nicht programmieren, sondern es genügt in der Regel, Einträge in einer Textdatei zu ändern, die den Authentisierungsvorgang für diese Anwendung konfiguriert. Dabei können Sie nicht nur beeinflussen, wo die Benutzerdaten herkommen, sondern sogar völlig neue Authentisierungsmechanismen integrieren, etwa Fingerabdruck- oder Smartcard-Lesegeräte oder Einmalkennwörter.

Es ist sogar möglich, mehrere alternative Authentisierungsmechanismen gleichzeitig zu konfigurieren. Einerseits können Sie so Benutzer zum Beispiel gegen ein LDAP-Verzeichnis authentisieren und nur, wenn das nicht gelingt, auf

die lokale /etc/passwd-Datei zurückfallen. Andererseits können Sie so eine »Mehrfaktoren-Authentisierung« realisieren, wo Benutzer nur dann zugelassen werden, wenn sie ihr Kennwort wissen *und* den richtigen Fingerabdruck vorweisen können.

PAM ist kein »Ersatz« für andere Authentisierungsmechanismen, sondern liefert die grundlegende Infrastruktur, um andere Mechanismen überhaupt flexibel benutzen zu können. Ob Sie mit PAM eine netzwerkweite zentralisierte Benutzerverwaltung mit »Single Sign-On« realisieren oder komplett auf jegliche Authentisierung verzichten, ist ganz Ihre Sache.

2.2 Struktur von PAM

2.2.1 Überblick

Wie schon angedeutet besteht PAM aus einer Bibliothek (sehr einfallsreich libpam.so genannt), die von Anwendungsprogrammen eingebunden wird. Außerdem gehört zu PAM eine Reihe von Modulen, die einzelne Schritte des Authentisierungsvorgangs durchführen. Wie die Authentisierung für ein Programm im Detail durchgeführt wird, hängt davon ab, welche PAM-Module in welcher Reihenfolge ausgeführt werden und wie PAM (die Bibliothek) auf die Ergebnisse dieser Ausführungsschritte reagiert. Die Modulauswahl und -reihenfolge wird pro Anwendung in Konfigurationsdateien beschrieben.

Module

Die PAM-Module finden Sie auf Ihrem System normalerweise im Verzeichnis /lib/security. (/lib/pam wäre vielleicht ein geschickterer Name gewesen, aber so ist es halt nicht gekommen.)

2.2.2 PAM-Funktionen

Die verschiedenen Funktionen, die PAM ausführt, sind in vier Gruppen einteilen. Jedes PAM-Modul ist mindestens einer dieser Gruppen zuzuordnen, wobei viele Module in mehreren der Gruppen Funktionen erbringen. Die Gruppen sind im Einzelnen:

account (Benutzerkonto) Module in dieser Gruppe prüfen, ob das angestrebte Benutzerkonto für eine Authentisierung zur Verfügung steht. Dies kann von Kriterien abhängen wie ob das Konto abgelaufen ist oder eine Anmeldung nur zu bestimmten Tageszeiten möglich sein soll, oder sogar vom Konto unabhängige Kriterien einbeziehen wie die aktuelle Anzahl von schon angemeldeten Benutzern.

auth (Authentisierung) Module in dieser Gruppe prüfen die Identität des Benutzers. Dazu können sie zum Beispiel ein geheimes Kennwort erfragen und gegen eine geeignete Benutzerdatenbank (etwa die lokale /etc/passwd-Datei, ein LDAP-Verzeichnis oder einen SQL-Datenbankserver) prüfen, die Gültigkeit eines Einmalkennworts (Stichwort »SecurID«) testen oder einen Netzhautscan erheben und auswerten.

Genaugenommen kümmern auth-Module sich nicht nur um Authentisierung (Feststellen, ob der Benutzer ist, wer er behauptet), sondern auch um Autorisierung (Rechtezuweisung an den Benutzer auf der Basis der festgestellten Identität). Bei Unix/Linux ergeben sich die Rechte eines Benutzers im einfachsten Fall direkt aus seiner festgestellten Identität, aber PAM-Module können einem Benutzer zum Beispiel zusätzliche Gruppen zuweisen, in die er nicht aufgrund von /etc/group aufgenommen würde.

```
# Konfiguration für login
auth     required   pam_issue.so issue=/etc/issue
auth     requisite  pam_securetty.so
auth     requisite  pam_nologin.so
auth     required   pam_unix.so nullok_secure
account  required   pam_unix.so
session  required   pam_env.so readenv=1 envfile=/etc/default/locale
session  required   pam_limits.so
session  optional   pam_motd.so
session  required   pam_unix.so
password required   pam_unix.so obscure sha512
```

Bild 2.1: Beispiel für eine PAM-Konfigurationsdatei

session (Sitzung) Module in dieser Gruppe führen Schritte durch, die nach einer erfolgreichen Authentisierung zum Aufbau einer Sitzung nötig sind. Zum Beispiel könnten sie die »Nachricht des Tages« (engl. *message of the day*) aus der Datei `/etc/motd` anzeigen, prüfen, ob ungelesene E-Mail vorliegt, und ähnliches.

password (Kennwort) Module in dieser Gruppe gestatten Benutzern das Ändern ihres Kennworts, unabhängig davon, wo das Kennwort gespeichert wird. Wenn Sie beispielsweise einen LDAP-Server zur Authentisierung verwenden, können Sie so dafür sorgen, dass Kennwortänderungen auf dem LDAP-Server aktenkundig gemacht werden. Außerdem können Sie mit `password`-Modulen zum Beispiel prüfen, ob Benutzerkennwörter offensichtlich vom Benutzernamen abgeleitet oder anderweitig leicht zu raten sind.

2.2.3 Konfigurationsdateien

Die PAM-Konfiguration für Ihr System finden Sie normalerweise im Verzeichnis `/etc/pam.d`, wo (theoretisch) für jedes Programm, das PAM verwendet, eine Datei steht, die die Authentisierungsschritte für dieses Programm angibt.

Wenn ein Programm zwar PAM benutzen möchte, aber keine Konfigurationsdatei für das Programm in `/etc/pam.d` hinterlegt ist, dann liest PAM ersatzweise die Datei `/etc/pam.d/other`.

Die Datei `/etc/pam.d/other` wird auch konsultiert, wenn ein Programm zwar eine eigene Konfigurationsdatei hat, aber für eine Modulgruppe (siehe voriger Abschnitt) in dieser Datei keine Module konfiguriert sind. In diesem Fall werden die in `/etc/pam.d/other` für die entsprechende Gruppe konfigurierten Module verwendet.

Bild 2.1 enthält ein (vereinfachtes) Beispiel für eine PAM-Konfigurationsdatei, hier für das Programm `login`, das sich um die Anmeldung auf Textkonsolen kümmert. Kommentarzeilen (mit »`#`« am Anfang) und Leerzeilen werden ignoriert; alle anderen Zeilen bestehen aus drei Spalten, deren erste die Modulgruppe (siehe oben) angibt. Die dritte Spalte benennt ein Modul (gegebenenfalls mit Parametern), während die zweite Spalte angibt, was mit dem Ergebnis des Moduls passieren soll.

Normalerweise reicht eine Zeile bis zum Zeilenende; überlange Konfigurationszeilen können Sie auf mehrere Zeilen in der Datei verteilen, indem Sie ein »\« ans Zeilenende setzen und in der nächsten Zeile in der Datei fortfahren.

Für die zweite Spalte gibt es zwei Arten von Werten. Die »traditionelle« Methode verwendet die folgenden Schlüsselwörter:

required Wenn ein mit `required` aufgerufenes Modul Misserfolg meldet, dann meldet PAM Misserfolg an das aufrufende Programm, aber erst, wenn die komplette Reihe von PAM-Modulen abgearbeitet wurde.

requisite Wenn ein mit `requisite` aufgerufenes Modul Misserfolg meldet, dann meldet PAM *sofort* Misserfolg an das aufrufende Programm.

sufficient Wenn ein mit `sufficient` aufgerufenes Modul Erfolg meldet, dann meldet PAM sofort Erfolg an das aufrufende Programm – es sei denn, dass schon weiter vorne ein mit `required` aufgerufenes Modul Misserfolg gemeldet hat (in diesem Fall wird der Erfolg des `sufficient`-Moduls ignoriert). Misserfolg eines mit `sufficient` aufgerufenen Moduls ist zunächst kein Problem.

optional Der Erfolg oder Misserfolg eines solchen Moduls ist nur wichtig, wenn es das einzige Modul in seiner Modulgruppe ist.

Der Unterschied zwischen `required` und `requisite` ist subtil, aber durchaus wichtig. Zum Beispiel fragt das System bei der Anmeldung *immer* nach Benutzername *und* Kennwort, selbst wenn es im Fall eines ungültigen Benutzernamens schon nach dessen Eingabe eine Fehlermeldung ausgeben könnte – aber das würde Angreifer in die Lage versetzen, gültige Benutzernamen zu raten. Der tatsächliche Ansatz läßt einen Angreifer im Unklaren, ob er nur das falsche Kennwort zu einem gültigen Benutzernamen angegeben hat oder ob der Benutzername schon selber ungültig war. Mit `required` und `requisite` können Sie dieses Verhalten modellieren und gleichzeitig dafür sorgen, dass Benutzer keine Kennwörter über möglicherweise unsichere Kommunikationskanäle eingeben: Im Beispiel in Bild 2.1 wird zum Beispiel für das Modul `pam_securetty.so`, das prüft, ob der Benutzer an einer »sicheren« Konsole sitzt[1], `requisite` angegeben, damit der Authentisierungsvorgang gegebenenfallls gleich endet, bevor der Benutzer seinen Benutzernamen und sein Kennwort auf einem möglicherweise unsicheren Terminal eintippt.

Außerdem gibt es noch die Schlüsselwörter `include` und `substack`. Beide lesen aus der als Parameter angegebenen Datei die Zeilen für die betreffende Modulgruppe aus und arbeiten sie so ab, als stünden sie anstelle der `include`- bzw. `substack`-Zeile in der ursprünglichen Konfigurationsdatei. Der Unterschied zwischen den beiden ist, dass bei `include` die Rückgabewerte (und zweiten Spalten) für den kompletten Authentisierungsvorgang gelten und bei `substack` nur für die zusätzlich eingelesene Datei.

Die von Debian GNU/Linux abgeleiteten Distributionen unterstützen einen alternativen Mechanismus zum Einlesen untergeordneter Konfigurationsdateien. Eine Zeile der Form

```
@include common-auth
```

liest dabei die Datei `/etc/security/common-auth` ein. (Diese sollte dann nur Zeilen für die Modulgruppe `auth` enthalten.) Diese Methode ist ein Debian-spezifisches Überbleibsel aus den düsteren Zeiten, als PAM selbst noch keinen solchen Mechanismus enthielt; in aktuellen Versionen von Debian und Ubuntu können (und sollten) Sie den nativen Mechanismus von PAM verwenden.

Alternativ zu den »traditionellen« Schlüsselwörtern können Sie auch eine weitaus differenziertere Methode verwenden, die es gestattet, genau festzulegen, wie PAM mit den verschiedenen Rückgabewerten von Modulen umgehen soll. Dabei hat der Eintrag in der zweiten Spalte die Form

[1] Typischerweise sind das die »virtuellen« Textkonsolen `/dev/tty0` bis `/dev/tty63` und allfällige direkt angeschlossene serielle Terminals, die unmittelbar neben dem Rechner stehen … naja, letzteres heute nicht mehr so oft.

Traditionell	Modern
required	[success=ok new_authtok_reqd=ok ignore=ignore default=bad]
requisite	[success=ok new_authtok_reqd=ok ignore=ignore default=die]
sufficient	[success=done new_authtok_reqd=done default=ignore]
optional	[success=ok new_authtok_reqd=ok default=ignore]

Tabelle 2.1: Traditionelle und moderne Rückgabewerte bei PAM

```
[wert_1 = aktion_1 wert_2 = aktion_2 …]
```

und $wert_i$ entspricht jeweils einem der möglichen Rückgabewerte der im Modul aufgerufenen Funktion (die sich wiederum von der Modulgruppe ableitet) oder `default` (als Standardvorgabe), etwa so:

```
[success=ok new_authtok_reqd=ok ignore=ignore default=bad]
```

Die $aktion_i$-Werte können wie folgt sein:

`ignore` Der Rückgabewert des Moduls wird ignoriert

`bad` Der Rückgabewert wird als Anzeichen dafür angesehen, dass der Authentisierungsvorgang (später) fehlschlagen sollte (dies entspricht vage dem, was bei `required` passieren soll – aber siehe unten!).

`die` Entspricht `bad`, aber liefert den Fehler sofort an das aufrufende Programm zurück (denken Sie an `requisite`).

`ok` Gibt an, dass dieser Rückgabewert zum Rückgabewert des gesamten Authentisierungsvorgangs beitragen sollte. Das heißt, wenn bis hierher Erfolg zurückgemeldet werden würde, dann überschreibt dieser Rückgabewert das, aber wenn bis hierher ein Fehlschlag zurückgemeldet werden würde, wird er ignoriert.

`done` Entspricht `ok`, aber liefert den Erfolg (oder früheren Fehlschlag) sofort an das aufrufende Programm zurück (denken Sie an `sufficient`).

n (eine positive ganze Zahl) Entspricht `ok`, aber überspringt die folgenden n Konfigurationszeilen. ($n = 0$ ist verboten.)

`reset` Alles Bisherige soll vergessen und mit der nächsten Zeile von vorne weitergemacht werden.

Tabelle 2.1 zeigt die Korrespondenz zwischen den »traditionellen« Schlüsselwörtern und ihren »modernen« Äquivalenten. Sie können ungeniert weiter die Schlüsselwörter verwenden; sie werden nicht verschwinden.

In der dritten Spalte steht ein Modulname – entweder als absoluter Pfad oder als Dateiname relativ zu `/lib/security` – gefolgt von allfälligen (modulspezifischen) Argumenten. (Bei `include` und `substack` enthält die dritte Spalte den Namen – absolut oder relativ zu `/etc/pam.d` – der einzulesenden Datei.)

Grundsätzlich können Sie auch die komplette PAM-Konfiguration Ihres Systems in der Datei `/etc/pam.conf` ablegen – Sie müssen dann nur den Namen des Programms an den Zeilenanfang stellen: Die Zeile

```
login auth required pam_unix.so shadow nullok
```

entspricht der Zeile

```
auth required pam_unix.so shadow nullok
```

in der Datei /etc/pam.d/login. Allerdings wird das in aktuellen Distributionen nicht mehr so gemacht. Wenn das Verzeichnis /etc/pam.d existiert, wird /etc/pam.conf von PAM nicht mehr angeschaut.

Übungen

2.1 [2] Das PAM-Modul pam_group.so kann Benutzern eine Gruppe zum Beispiel auf der Basis des Namens des Programms zuweisen, über das gerade eine Authentisierung stattfindet (etwa login oder xdm). Wofür könnte das gut sein?

2.2 [1] Warum bevorzugen Distributionen heutzutage die PAM-Konfiguration über /etc/pam.d gegenüber der über /etc/pam.conf?

2.3 Wichtige PAM-Module

2.3.1 Überblick

Die PAM-Infrastruktur ist nützlich, aber der wirkliche Witz sind die Module. In diesem Abschnitt beschreiben wir einige der gängigeren Module – die PAM-Distribution enthält schon drei Dutzend oder so, und in anderen Quellen läßt sich auch noch jede Menge Interessantes finden.

Für LPI-Kandidaten: In den Prüfungszielen besonders hervorgehoben sind die PAM-Module pam_unix, pam_cracklib, pam_limits und pam_listfile.

2.3.2 Ganz wichtige Module

Die folgenden Module werden Sie mit einiger Sicherheit in einer »normalen« PAM-Konfiguration antreffen. Nach dem Modulnamen ist angegeben, zu welcher Modulgruppe das Modul gehört: »A« steht für auth, »C« für account, »S« für session und »P« für password. Die meisten PAM-Module haben auch ihre eigenen Handbuchseiten, die so heißen wie das betreffende Modul – etwa pam_deny(8).

pam_deny (ACSP) Dieses Modul meldet immer Misserfolg (und ist damit das Gegenteil von pam_permit, siehe unten). Sie können es zum Beispiel in einer /etc/pam.d/other-Datei verbauen, etwa wie folgt:

```
auth     required   pam_warn.so
auth     required   pam_deny.so
account  required   pam_warn.so
account  required   pam_deny.so
password required   pam_warn.so
password required   pam_deny.so
session  required   pam_warn.so
session  required   pam_deny.so
```

Das Modul pam_warn (s. u.) protokolliert dabei den Zugriff.

Passen Sie auf, wenn Sie mit pam_deny operieren – Sie können sich leicht komplett aus dem System aussperren!

pam_env (AS) Dieses Modul erlaubt das kontrollierte Setzen oder Löschen von Umgebungsvariablen und richtet sich dabei nach der Datei /etc/security/pam_env.conf (sofern keine andere Datei angegeben wurde). Außerdem liest es Umgebungsvariable in der Form »VARIABLE=WERT« aus der Datei /etc/environment. Die wichtigsten Optionen sind

conffile=⟨*Dateiname*⟩ Gibt eine andere Datei an, die statt `/etc/security/pam_env.conf` gelesen werden soll.

debug Gibt über `syslog`(3) umfangreiche Informationen zur Fehlersuche aus.

envfile=⟨*Dateiname*⟩ Gibt eine andere Datei an, die statt `/etc/environment` gelesen werden soll.

readenv=0|1 Gibt an, ob die mit `envfile` benannte Datei (ersatzweise `/etc/environment`) überhaupt gelesen werden soll.

pam_group (A) Gibt einem Benutzer zusätzliche Gruppenzugehörigkeiten, basierend auf dem Namen des Programms, über das die Authentisierung abgewickelt wird, sowie anderen Kriterien wie dem Terminalnamen, dem Benutzernamen, Wochentag und Uhrzeit. Die genauen Regeln dafür stehen in der Datei `/etc/security/group.conf` (was nicht geändert werden kann). Nähere Informationen finden sich in `pam_group`(8) und `group.conf`(5).

pam_issue (A) Gibt den Inhalt der Datei `/etc/issue` aus, bevor nach einem Benutzernamen gefragt wird. In der Datei können diverse Steuersequenzen stehen, die bei der Ausgabe durch die jeweils passenden Angaben ersetzt werden – Details stehen in `issue`(8). Die wichtigsten Optionen sind:

noesc Verhindert die Ersetzung von Steuersequenzen

issue=⟨*Dateiname*⟩ Benennt eine andere Datei anstelle von `/etc/issue`

pam_lastlog (S) Gibt aus, wann der Benutzer sich zuletzt angemeldet hatte, und kümmert sich um die Datei `/var/log/lastlog`. Die wichtigsten Optionen sind:

silent Unterdrückt die Ausgabe über frühere Logins und aktualisiere nur `/var/log/lastlog`.

never Begrüßt den Benutzer mit einer Willkommensnachricht, wenn `/var/log/lastlog` keine alten Einträge für ihn enthält.

nodate (und `noterm`, `nohost`) Unterdrückt Teile der Ausgabe.

nowtmp Aktualisiert *nicht* die `wtmp`-Datei.

noupdate Aktualisiert überhaupt keine Dateien.

showfailed Gibt die Anzahl fehlgeschlagener Anmeldeversuche aus.

pam_limits (S) Setzt Ressourcenlimits für bestimmte Benutzer (gegebenenfalls sogar `root`). Die Limits stehen in der Datei `/etc/security/limits.conf`; außerdem werden alle Dateien in `/etc/security/limits.d` in der lexikografischen Reihenfolge ihrer Namen gelesen und so interpretiert, als sei ihr Inhalt hinten an `limits.conf` angehängt. Die gängigsten Optionen sind:

conf=⟨*Dateiname*⟩ Benennt eine alternative Datei anstatt `limits.conf`. Wenn Sie diese Option verwenden, wird `/etc/security/limits.d` nicht abgearbeitet.

Mehr über die Spezifikation von Ressourcenlimits finden Sie in `limits.conf`(5).

pam_mail (AS) Gibt eine Nachricht aus, wenn der Benutzer neue E-Mail in seinem Postfach hat. Details darüber, wo Postfächer zu finden sind, und ähnliches lassen sich über Optionen angeben; in `pam_mail`(8) stehen die Einzelheiten.

pam_motd (S) Gibt den Inhalt der Datei `/etc/motd` aus (maximal 64 KiB). Mit der Option `motd` können Sie eine andere Datei benennen.

pam_nologin (AC) Hindert Benutzer (außer `root`) daran, sich anzumelden, wenn die Datei `/etc/nologin` existiert – statt dessen wird der Inhalt von `/etc/nologin` ausgegeben. Die gängigsten Optionen sind:

file=⟨*Dateiname*⟩ Benennt eine andere Datei statt `/etc/nologin`.

successok Das Modul liefert Erfolg, wenn die Datei nicht existiert (der Standardwert ist »Ignorieren«).

pam_permit (ACSP) Dieses Modul liefert immer Erfolg zurück (und ist damit das Gegenteil von `pam_deny`, siehe oben). Wenn nötig, setzt es den Namen des »Benutzers« auf `nobody`.

Dieses Modul ist sehr, *sehr*, **sehr**, **SEHR** gefährlich!!! Benutzen Sie es mit der gebotenen Vorsicht.

pam_securetty (A) Erlaubt es `root` nur eine Anmeldung auf bestimmten Geräten (Terminals). Die Namen der zugehörigen Gerätedateien stehen in der Datei `/etc/securetty`. Das Modul prüft auch, dass `/etc/securetty` eine gewöhnliche Datei ist und nicht von allen Benutzern geschrieben werden kann. Sie sollten das Modul mit `required` oder `requisite` aufrufen, bevor Sie irgendwelche Module mit `sufficient` aufrufen.

pam_time (C) Erlaubt das Anmelden nur zu bestimmten Wochentagen und Tageszeiten und auf bestimmten Terminals. Die genauen Regeln dafür stehen in `/etc/security/time.conf` und sind in `time.conf`(5) dokumentiert.

pam_unix (ACSP) Führt die »normale« Unix-artige Benutzerauthentisierung durch, indem es die üblichen Bibliotheksfunktionen verwendet, um Benutzerdaten abzurufen (und gegebenenfalls zu setzen). Normalerweise werden also Daten aus `/etc/passwd` und `/etc/shadow` verwendet. Als `auth`-Modul prüft `pam_unix` das Kennwort des Benutzers (es darf im Standardfall nicht leer sein). Als `account`-Modul stellt es sicher, dass die Parameter in `/etc/shadow` einen Zugriff nicht verhindern und erzwingt gegebenenfalls das Ändern des Kennworts, wenn es abgelaufen ist. Als `session`-Modul protokolliert es den Zeitpunkt des An- und Abmeldens des Benutzers, und als `password`-Modul ändert es das Benutzerkennwort. Die wichtigsten Optionen sind:

nullok Erlaubt eine Anmeldung, wenn das Kennwort des Benutzers in der Benutzerdatenbank leer ist.

nullok_secure Erlaubt eine Anmeldung, wenn das Kennwort des Benutzers in der Benutzerdatenbank leer ist, nur, wenn der Benutzer sich auf einem der in `/etc/securetty` aufgezählten Terminals anzumelden versucht.

try_first_pass Wenn PAM in diesem Anmeldevorgang schon ein Kennwort gelesen hat, wird zuerst jenes ausprobiert und dann notfalls nach einem anderen gefragt.

use_first_pass Verwendet ein schon vorher in diesem Anmeldevorgang eingegebenes Kennwort; wenn keins existiert oder das Kennwort inakzeptabel ist, wird der Benutzer abgewiesen.

use_authtok Wenn ein Kennwort geändert werden soll, dann soll dasjenige verwendet werden, das durch ein früheres Modul gesetzt wurde.

remember=⟨*Zahl*⟩ Speichert eine Anzahl von alten Kennwörtern in `/etc/security/opasswd` und läßt diese Kennwörter nicht zu (damit Benutzer nicht zwischen wenigen Kennwörtern hin und her wechseln).

md5 (und `bigcrypt`, `sha256`, `sha512`, `blowfish`) Wählt einen Verschlüsselungsalgorithmus für künftig geänderte Kennwörter. (Aktuelle Kennwörter können nicht »umverschlüsselt« werden, weil das System sie dazu im Klartext kennen müsste.)

min=⟨*Zahl*⟩ Setzt eine minimale Länge für Kennwörter.

obscure Prüft bei der Kennwortänderung neu eingegebene Kennwörter auf übermäßige Plumpheit. Zu den geprüften Kriterien gehören:

- Das neue Kennwort darf nicht »das bisherige Kennwort umgekehrt« oder eine »rotierte« Version des vorigen (etwa `eheimg` statt `geheim`) sein.
- Das neue Kennwort darf sich vom alten nicht nur durch Groß- und Kleinschreibung unterscheiden und auch sonst nicht zu ähnlich sein.
- Das neue Kennwort darf nicht »zu einfach« sein, muss also eine gewisse Länge haben und eine Mixtur verschiedener Zeichen (Buchstaben, Ziffern, …) enthalten.

`pam_warn` Gibt den aktuellen Programmnamen, das Terminal, den Benutzernamen und einen allfälligen entfernten Benutzer- und Rechnernamen über `syslog`(3) ins Systemprotokoll aus. Beeinflusst die Authentisierung nicht weiter.

Übungen

2.3 [!2] Sorgen Sie dafür, dass der Benutzer `hugo` maximal 10 Prozesse gleichzeitig laufen lassen kann.

2.4 [2] Angenommen, Sie möchten die Kennwortverschlüsselung auf Ihrem System komplett auf etwas Sicheres umstellen (zum Beispiel `sha512`). Wie können Sie das erreichen?

2.3.3 Andere interessante Module

Diese Module sind nützlich, aber nicht typischerweise Bestandteil einer PAM-Standardkonfiguration (auch wenn das von der konkreten Distribution abhängen kann):

`pam_access` (ACSP) Prüft Zugriffsrechte auf der Basis von Regeln, die Netzwerkadressen (IPv4 und IPv6), DNS-Namen und ähnliches beschreiben. Diese Regeln stehen in `/etc/security/access.conf` (siehe `access.conf`(5)).

`pam_cracklib` (P) Prüft beim Ändern von Kennwörtern das neue Kennwort und lehnt es ab, wenn es zu »schwach« aussieht. Neben den bei `pam_unix` genannten Kriterien werden noch weitere geprüft:

- Die zulässige »Ähnlichkeit« des neuen Kennworts mit dem alten kann konfiguriert werden.
- Das Kennwort darf nicht zu viele aufeinanderfolgende gleiche Zeichen enthalten.
- Das Kennwort darf den Benutzernamen weder »normal« noch umgekehrt enthalten.

Außerdem wird (als erstes) geprüft, ob das Kennwort in einem Wörterbuch zu finden ist.

`pam_listfile` (ACSP) Prüft Zugriffsrechte auf der Basis einer beliebigen Datei. Zum Beispiel können Sie testen, ob der gewünschte Benutzername auf einer »weißen« oder »schwarzen« Liste steht und den Benutzer entsprechend zulassen oder abweisen. Die wichtigsten Optionen sind:

`file=`⟨*Dateiname*⟩ Die Datei, die durchsucht werden soll. Hierbei muss es sich um eine einfache Datei handeln (keine FIFOs, Sockets oder ähnliches), die nicht von »allen Benutzern« geschrieben werden darf.

`item=tty|user|rhost|ruser|group|shell` Welches Attribut aus dem Anmeldevorgang in der Datei gesucht werden soll. `tty` beschreibt das Terminal, auf dem der Anmeldevorgang stattfindet, `user` den gewünschten Benutzernamen, `rhost` den Namen des entfernten Rechners, von dem aus der Anmeldevorgang gestartet wurde (falls vorhanden), und `ruser` entsprechend den Namen des anfragenden Benutzers dortselbst (falls verfügbar).

onerr=succeed|fail Was passieren soll, wenn etwas Unvorhergesehenes passiert, also zum Beispiel die mit `file` angegebene Datei nicht gefunden oder geöffnet werden kann.

sense=allow|deny Was passieren soll, wenn das Gesuchte in der Datei gefunden wird – `allow` meldet Erfolg (im Sinne von PAM), `deny` meldet Misserfolg. Wenn das Gesuchte nicht gefunden wird, wird jeweils das Gegenteil gemeldet.

apply=user|@group Erlaubt es, die Prüfung auf bestimmte Benutzer oder die Mitglieder einer bestimmten Gruppe zu beschränken. Das funktioniert allerdings nur für `item=tty`, `item=rhost` und `item=shell`.

Das folgende Beispiel illustriert, wie Sie die Funktion der Datei `/etc/ftpusers` nachahmen können (*Wichtig:* In der Datei `/etc/ftpusers` konnten Sie beim kanonischen FTP-Server die Benutzer aufzählen, die FTP *nicht* benutzen dürfen – wenn Sie jetzt denken, dass die Datei dann eigentlich `/etc/ftpnonusers` heißen sollte, geben wir Ihnen völlig recht, aber man hat damals leider versäumt, uns zu fragen). Mit anderen Worten, wenn in `/etc/ftpusers` etwas steht wie

```
richard
hugo
adelheid
```

und der Benutzer `hugo` versucht, eine FTP-Sitzung zu öffnen, soll er abgewiesen werden. Sie erreichen das mit dem folgenden Eintrag in der Datei `/etc/pam.d/ftpd` (oder welche auch immer für Ihren FTP-Server gilt):

```
auth required pam_listfile.so onerr=succeed item=user \
                              sense=deny file=/etc/ftpusers
```

pam_mkhomedir (S) Legt das Heimatverzeichnis für einen Benutzer an, falls es nicht existiert. Dieses Modul macht Ihnen als Administrator die Arbeit leichter, wenn Sie Benutzer in einem zentralen Verzeichnisdienst (etwa LDAP) eintragen. Wenn der Benutzer sich zum ersten Mal auf einem Rechner anmeldet und dort kein Heimatverzeichnis für ihn vorhanden ist, können Sie es automatisch anlegen und mit einer Grundausstattung von Dateien versehen (so wie `useradd` das auch machen würde). Die wichtigsten Optionen sind:

skel=⟨*Verzeichnisname*⟩ Die Dateien im benannten Verzeichnis (anstatt von `/etc/skel`, dem Standardwert) werden als Grundausstattung ins neue Heimatverzeichnis kopiert.

umask=⟨*Maske*⟩ Setzt die `umask`. Der Standardwert ist `0022`.

pam_pwhistory (P) Dieses Modul merkt sich die letzten n Kennwörter, die jeder Benutzer verwendet hat, in der Datei `/etc/security/opasswd` und zwingt ihn, bei Änderungen ein frisches Kennwort zu verwenden, das nicht in der Liste auftaucht. Dies ist nur dann sinnvoll, wenn Sie kein zentrales Verzeichnis für Benutzer haben (etwa LDAP), da die alten Kennwörter nur lokal gespeichert werden und darum nicht zur Verfügung stehen, wenn der Benutzer sein Kennwort von einem anderen Rechner aus ändert. Die wichtigsten Optionen sind:

enforce_for_root Wenn diese Option gesetzt ist, gilt die Einschränkung auch für `root`.

remember=n Anzahl der zu merkenden Kennwörter. Standard ist $n = 10$.

retry=k Fragt den Benutzer maximal k mal, bevor Misserfolg zurückgemeldet wird. Standard ist $k = 1$.

use_authtok Bringt das Modul dazu, das von einem vorher in der Konfiguration aufgetretenen anderen Modul abgefragte Kennwort zu verwenden, statt selbst nach einem zu fragen.

Wenn Sie bis hierhin aufgepasst haben, dann ist Ihnen sicher aufgefallen, dass das Modul pam_unix eine ganz ähnliche Funktionalität aufzuweisen scheint. Siehe hierzu Übung 2.6.

pam_rootok (A) Dieses Modul liefert genau dann Erfolg, wenn die UID des aufrufenden Benutzers 0 ist. Damit können Sie root von weiteren Authentisierungshürden (etwa der Kennwortabfrage bei su) ausnehmen.

pam_rootok prüft die »reale UID« des aufrufenden Benutzers. Wenn das Programm, das PAM verwendet, ein Set-UID-Programm ist, dann läuft es mit der Identität des *Eigentümers* seiner Programmdatei als »effektiver UID«, aber pam_rootok orientiert sich trotzdem an der ursprünglichen UID des aufrufenden Benutzers.

pam_tally (AC) Dieses Modul erfüllt zwei verwandte Zwecke. In der Modulgruppe auth zählt es fehlgeschlagene Anmeldeversuche pro Benutzer und weist einen Benutzer ab, wenn bei seinem Konto zu viele davon vorgekommen sind. In der Modulgruppe account setzt es die Anzahl der fehlgeschlagenen Anmeldeversuche auf Null zurück (das passiert grundsätzlich zwar auch bei verschiedenen anderen PAM-Modulen, aber mit diesem Modul können Sie sichergehen). Die folgenden Optionen (unter anderen) können Sie in beiden Fällen verwenden:

file=⟨*Dateiname*⟩ In dieser Datei merkt das Modul sich Informationen über die fehlgeschlagenen Anmeldeversuche. Standard ist /var/log/faillog.

onerr=fail|succeed Was passieren soll, wenn etwas Unvorhergesehenes auftritt, etwa die Datei mit den Zählern nicht geöffnet werden konnte.

Für auth gelten unter anderem die folgenden Optionen:

deny=*n* Weist den Benutzer ab, wenn mehr als *n* fehlgeschlagene Anmeldeversuche für ihn registriert wurden.

lock_time=*n* Weist den Benutzer ab, wenn der letzte fehlgeschlagene Anmeldeversuch höchstens *n* Sekunden her ist.

unlock_time=*n* Wenn Sie diese Option verwenden und ein Benutzer die maximale Anzahl von Fehlversuchen überschritten hat, kann er sich nach *n* Sekunden wieder anmelden. Ansonsten muss der Systemadministrator (Sie) ihn manuell wieder freischalten.

magic_root Wenn diese Option gesetzt ist, dann wird der Zähler nur erhöht, wenn das Modul *nicht* von root (oder genauer gesagt einem Benutzer mit der UID 0) aufgerufen wird. Verwenden Sie das für Dienste wie su, die normale Benutzer aufrufen können.

no_reset Setzt die Anzahl der Fehlversuche bei einem erfolgreichen Anmeldevorgang nicht auf Null, sondern zählt sie nur herunter.

even_deny_root_account Sperrt gegebenenfalls auch den Zugang für root (ist normalerweise nicht der Fall).

Bei account können Sie die Optionen magic_root und no_reset verwenden.

Normalerweise kümmert pam_tally sich nicht um root, um Angriffen vorzubeugen, bei denen freche Benutzer versuchen, den Systemadministrator auszusperren. Dies ist unbedenklich, sofern Benutzer keinen interaktiven Shellzugang bekommen und root sich sowieso nicht direkt oder nur über die Konsole des Rechners anmelden kann.


Zum PAM-Modul `pam_tally` gehört auch ein Kommandozeilenprogramm namens `pam_tally`, mit dem Sie die Zähler für einzelne Benutzer zurücksetzen (oder auf beliebige Werte setzen) können, etwa so:

```
# pam_tally --user hugo --reset          Zurücksetzen
# pam_tally --user hugo --reset=5        Fünf Fehlversuche
```

(Rufen Sie als `root` mal »`pam_tally --help`« auf.)


Sie können auch das Programm `faillog` verwenden, um die Datei `/var/log/faillog` zu bearbeiten. `faillog` kann mehr als `pam_tally`, aber funktioniert nur mit dieser Datei. Lesen Sie `faillog`(8).

`pam_wheel` (AC) Dieses Modul implementiert die Gruppe `wheel`. Traditionell können bei BSD-artigen Unix-Systemen nur die Mitglieder dieser Gruppe mit `su` die Identität von `root` annehmen[2].


Wenn es auf dem System keine Gruppe namens `wheel` gibt, dann wird statt dessen die Gruppe mit der GID 0 verwendet (egal wie sie heißt).

Die gängigsten Optionen sind:

`group=`⟨*Gruppenname*⟩ Benutzt die benannte Gruppe anstatt `wheel` (oder der Gruppe mit der GID 0).

`deny` Verkehrt die Prüfung ins Gegenteil: Wenn der Benutzer `root`-Rechte anstrebt und Mitglied der Gruppe `wheel` ist, dann wird er abgewiesen.

`trust` Das Modul liefert direkt Erfolg zurück, anstatt »Ignorieren« zu liefern (der Standardfall – bei dem es einer Kennwortabfrage zumindest nicht im Weg steht). Wenn Sie geschickt sind, können Sie so erreichen, dass Mitglieder der Gruppe `wheel` ohne weitere Kennwortabfrage `root` werden können. (Siehe hierzu Übung 2.8.)

Übungen


2.5 [!2] Angenommen, in der Datei `/etc/ssh/sshusers` stehen einige lokale Benutzernamen (einer pro Zeile). Wie können Sie erreichen, dass sich nur die in dieser Datei benannten Benutzer über die Secure Shell anmelden können? (Vergessen Sie für den Moment mal die `sshd`-Konfigurationseinstellung `AllowUsers`.) Entwerfen Sie eine PAM-basierte Lösung und probieren Sie sie aus. (Sorgen Sie dafür, dass der `sshd` tatsächlich PAM verwendet, indem Sie sicherstellen, dass in der Datei `/etc/ssh/sshd_config` eine Zeile der Form »`UsePAM yes`« existiert.) (*Für Extrapunkte:* Realisieren Sie dasselbe sinngemäß auch für Gruppen.)


2.6 [2] Warum gibt es das Modul `pam_pwhistory`, wo doch schon das Modul `pam_unix` die Option `remember` unterstützt?


2.7 [2] Wie können Sie das Modul `pam_tally` verwenden, um den Zugang für Benutzer zu sperren, ohne ihre Kennwörter zu ändern?


2.8 [!2] Wie würden Sie PAM für das Programm `su` konfigurieren, damit Mitglieder der Gruppe `wheel` ohne Kennwortabfrage die Identität von `root` (oder eine andere Identität) annehmen, andere Benutzer hingegen `su` gar nicht verwenden können?

[2] Bevor Sie fragen: Wir wissen auch nicht, wo der Name `wheel` herkommt. Die GNU-Version von `su` unterstützt die Gruppe `wheel` übrigens gar nicht, da Richard M. Stallman die Idee vehement ablehnt; nach Ansicht vieler einer seiner weniger cleveren Standpunkte.

2.4 PAM und der Name Service Switch

PAM ist nicht zu verwechseln mit dem »Name Service Switch« (NSS), der Infrastruktur, mit der Sie festlegen können, wo das System (zum Beispiel) Benutzerdaten speichert. Nochmal zum Mitschreiben:

- NSS dient dazu, dass Programme Zugriff auf die Benutzerdatenbank bekommen können. Das ist nötig dafür, dass zum Beispiel »`ls -l`« oder »`ps au`« die Namen von Benutzern anzeigen können – Dateisystem und Kernel speichern ja nur (numerische) UIDs, die über `/etc/passwd` (oder einen LDAP-Server, oder …) in textuelle Namen zurückübersetzt werden müssen.

  
  NSS kann auch den Zugriff auf verschiedene andere Systemdatenbanken und (wichtig!) die Auflösung von Rechnernamen in IP-Adressen regeln. Für unsere Zwecke hier sind aber nur die Benutzer- und Gruppendaten interessant.

- PAM dient dazu, Programme zu konfigurieren, die in irgendeiner Form Authentisierung machen müssen. Dazu gehören natürlich das Besorgen von identifizierenden Informationen (wie Kennwörtern) aus beliebigen Quellen und deren Prüfung auf Gültigkeit, aber auch noch viele andere mögliche Schritte (wie oben gezeigt).

  
  Insbesondere erlaubt PAM auch das *Ändern* von Kennwörtern – darum kümmert NSS sich überhaupt nicht.

Zwischen PAM und NSS gibt es viele Berührungspunkte, aber sie sind nicht identisch und können sich auch nicht gegenseitig ersetzen.


Wenn der NSS richtig konfiguriert ist, können Sie zum Beispiel Ihre Benutzerinformationen (vor allem Benutzername und Kennwort) in einem LDAP-Verzeichnis ablegen, und `pam_unix` wird das Richtige tun, nämlich Benutzer mit ihren LDAP-basierten Kennwörtern authentisieren. Das PAM-Modul `pam_ldap` ist deswegen aber nicht überflüssig – damit können Sie nicht nur die Verschlüsselung von Kennwörtern komplett dem LDAP-Server überlassen, sondern auch Ihre Benutzer ihre Kennwörter direkt auf dem LDAP-Server ändern lassen.

Kommandos in diesem Kapitel

`faillog` Verwaltet die Datei `/var/log/faillog`. `faillog`(8) 41
`pam_tally` Erlaubt die Verwaltung von Zählerdateien für `pam_tally` `pam_tally`(8) 40

Zusammenfassung

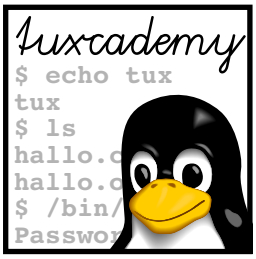
- Mit PAM können Programme Authentisierung flexibel konfigurierbar machen.
- PAM besteht aus einer Bibliothek, die von Programmen eingebunden wird, und einer Vielzahl von Modulen, die einzelne Authentisierungsschritte durchführen.
- PAM-Module kümmern sich um die Authentisierung, die authentisierungsunabhängige Kontenprüfung, den Aufbau von Sitzungen und das Ändern von Kennwörtern.
- Die Konfigurationsdateien für die verschiedenen Programme, die PAM benutzen, stehen in `/etc/pam.d`. Wenn dieses Verzeichnis nicht existiert (heute unüblich), wird die Datei `/etc/pam.conf` angeschaut.
- PAM-Module stehen in `/lib/security`.
- PAM ist nicht zu verwechseln mit NSS, dem »Name Service Switch«.

Literaturverzeichnis

Mor01 Andrew G. Morgan. »The Linux-PAM Application Developers' Guide«. Bestandteil der PAM-Distribution, August 2001.

Mor02a Andrew G. Morgan. »The Linux-PAM Module Writers' Guide«. Bestandteil der PAM-Distribution, September 2002.

Mor02b Andrew G. Morgan. »The Linux-PAM System Administrators' Guide«. Bestandteil der PAM-Distribution, Juni 2002.



3

Linux als LDAP-Client

Inhalt

Lernziele

- Wissen, wie Linux als Client an einen existierenden LDAP-Server angebunden werden kann
- Einfache Verzeichnisoperationen über die Kommandozeile auslösen können

Vorkenntnisse

- Kenntnisse der Linux-Systemadministration
- LDAP-Grundlagenwissen ist sehr hilfreich

3.1 Linux und LDAP

LDAP ist ein Verfahren, mit dem Rechner auf entfernte Verzeichnisdienste (engl. *directory services*) zugreifen können. Als LDAP-Server können Verzeichnisdienste wie »eDirectory« von Novell oder »Active Directory« von Microsoft dienen, aber auch nativ auf Linux laufende LDAP-Implementierungen wie OpenLDAP.

Im Rahmen dieser Schulungsunterlage ist es nicht möglich, die Grundlagen von LDAP sowie die Installation und Konfiguration eines LDAP-Servers zu erklären. Bitte beziehen Sie sich hierfür zum Beispiel auf die Linup-Front-Schulungsunterlage *LDAP und OpenLDAP*.

Ein Linux-Rechner kann einen LDAP-Verzeichnisdienst für unterschiedliche Zwecke nutzen. Auf Systemebene ist es möglich, Benutzer statt in die lokale Benutzerdatenbank (die Dateien `/etc/passwd`, `/etc/shadow` und `/etc/group`) ins Verzeichnis einzutragen und das Authentisierungssystem so zu konfigurieren, dass Benutzerinformationen, Kennwörter usw. vom Verzeichnis geholt werden. Es ist sogar möglich, direkt eine Authentisierung über den Verzeichnisdienst zu konfigurieren. Auf Anwendungsebene kann zum Beispiel ein Mail-Programm bei der Adressierung von E-Mail auf das Verzeichnis zurückgreifen und dort die Adressen einzelner Empfänger finden oder Verteilerlisten verwenden.

Die Konfiguration von System und Anwendungen für den Zugriff auf ein LDAP-basiertes Verzeichnis ist nicht völlig standardisiert. In der Regel werden aber die folgenden Informationen benötigt:

- Rechnername (oder IP-Adresse) und möglicherweise die Portnummer eines oder mehrerer anzusprechender LDAP-Server. LDAP verwendet standardmäßig den TCP-Port 389.

  LDAP erlaubt Redundanz und ermöglicht es, dass mehrere LDAP-Server denselben Datenbestand vorhalten. Clients können Informationen über alle diese Server zur Verfügung haben und probieren dann mehrere aus, falls der erste, den sie fragen, ihnen nicht antwortet.

  Oft können diese Informationen auch als »LDAP-URL« angegeben werden, etwa wie

  ```
  ldap://ldap.example.com:389
  ```

  (Die Portnummer ist optional, wenn sie 389 ist.)

- Einen »Basis-DN«, der den Punkt im (konzeptuellen) Verzeichnisbaum angibt, wo der Client mit Suchoperationen anfangen soll. Dies sieht typischerweise aus wie

  ```
  dc=example,dc=com
  ```

  oder (möglicherweise, aber seltener)

  ```
  o=Linup Front GmbH,l=Weiterstadt,st=Hessen,c=DE
  ```

  Der Basis-DN kann etwas mit dem Namen des LDAP-Servers zu tun haben, aber muss nicht.

  »DN« ist im LDAP-Jargon die Abkürzung von »Distinguished Name« (»ausgezeichneter Name«). DNs dienen dazu, Einträge im Verzeichnisbaum zu identifizieren, etwa so, wie Rechnernamen Rechner im DNS identifizieren oder Pfadnamen Dateien im Linux-Dateisystem.

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE dc=example,dc=com
URI  ldap://ldap.example.com ldap://ldap2.example.com:4711

SIZELIMIT 100
TIMELIMIT 15
```

Bild 3.1: Beispiel für eine ldap.conf-Datei

- Zugangsdaten für den LDAP-Server. Nicht jeder LDAP-Server stellt seine Informationen jedem zur Verfügung, der fragt – und in diesem Fall müssen Sie möglicherweise einen Benutzernamen und ein Kennwort bereithalten und außerdem wissen, welche Art von Authentisierung der LDAP-Server anbietet. LDAP kennt eine »simple« (aber unsichere) Authentisierung à la HTTP-»Basic«-Authentisierung, wo Benutzernamen und Kennwörter im Klartext übertragen werden (nicht schön – SSL/TLS sind Ihre Freunde), und eine ausgefeiltere Authentisierung auf der Basis von SASL.

Wie Sie diese Daten einem LDAP-benutzenden Programm schmackhaft machen können, hängt von dem betreffenden Programm ab. Typischerweise müssen Sie sie in geeigneter Form in eine Datei oder einen grafischen Dialog eintragen.

Die meisten Linux-Distributionen bieten zum elementaren Zugriff auf LDAP-basierte Verzeichnisse einen Satz von Kommandozeilenwerkzeugen an, die dem Dunstkreis des OpenLDAP-Projekts entstammen (aber auch mit anderen LDAP-Servern umgehen können – LDAP ist ja standardisiert). Diese Werkzeuge können Sie mit Optionen komplett über die Kommandozeile steuern, aber zu Ihrer Bequemlichkeit ist es möglich, Voreinstellungen in einer Konfigurationsdatei zu hinterlegen. Davon handelt der nächste Abschnitt.

3.2 Die Datei ldap.conf

Voreinstellungen für die kommandozeilenorientierten LDAP-Werkzeuge können Sie in der Datei ldap.conf ablegen, die Sie (je nach Distribution) in einem Verzeichnis wie /etc/ldap oder /etc/openldap finden können. Passen Sie auf – es könnte in Ihrem System auch andere ldap.conf-Dateien geben, die zum Beispiel von den PAM- und NSS-Modulen für LDAP verwendet werden; auch hier hängen die Details von Ihrer Distribution ab.

Eine beispielhafte ldap.conf-Datei sehen Sie in Bild 3.1. Wie üblich werden Leerzeilen und Zeilen, die mit »#« anfangen, ignoriert. Die restlichen Zeilen stellen Konfigurationseinträge dar. BASE bezeichnet den Basis-DN und URI einen oder mehrere URLs für LDAP-Server (sie müssen mit ldap:// anfangen). »SIZELIMIT 100« besagt, dass Suchoperationen maximal 100 Verzeichniseinträge als Ergebnis zurückliefern sollen, und »TIMELIMIT 15« sagt, dass Suchoperationen maximal 15 Sekunden dauern sollen.

Statt URI können Sie auch HOST und PORT verwenden. Dabei gibt PORT eine Portnummer vor, die verwendet wird, wenn die einzelnen bei HOST angegebenen Rechner keine eigenen Portnummern mitbringen. Sie könnten also in unserem Beispiel statt

```
URI  ldap://ldap.example.com ldap://ldap2.example.com:4711
```

etwas schreiben wie

```
PORT 389                                                    Eigentlich nicht nötig
HOST ldap.example.com ldap2.example.com:4711
```

HOST und PORT sind allerdings zugunsten von URI verpönt.

Der Wert 0 bei SIZELIMIT und TIMELIMIT steht für »unbegrenzt«.

Die Einstellungen bei SIZELIMIT und TIMELIMIT stellen die Perspektive des lokalen Clients dar. Es kann durchaus sein, dass der Server(administrator) seine eigenen Vorstellungen darüber hat, was geeignete Werte für diese Grenzen sind, und wenn die unterhalb von dem liegen, was der Client sich hier wünscht, dann behält am Schluss der Server recht. (Das Leben ist nicht fair.)

Als regulärer Benutzer können Sie ferner Voreinstellungen für die LDAP-Kommandozeilenwerkzeuge auch in der Datei .ldaprc in Ihrem Heimatverzeichnis ablegen (ldaprc – ohne Punkt – ist ebenso erlaubt). Zu guter Letzt lesen die Werkzeuge auch noch die Datei ldaprc im aktuellen Verzeichnis, wenn sie existiert.

Und wenn Ihnen diese reichhaltige Auswahl noch nicht genügt, können Sie in der Umgebungsvariable LDAPCONF den Pfad einer alternativen Konfigurationsdatei vorgeben oder in Umgebungsvariablen wie LDAPURI oder LDAPBASE gleich die Voreinstellungen selbst hinterlegen.

Die benutzerspezifischen Dateien sind vor allem für die Angabe von Authentisierungsinformationen wichtig, die in der systemweiten ldap.conf-Datei nicht zulässig sind.

Nähere Informationen über die Konfigurationsdatei(en) für die kommandozeilenbasierten LDAP-Werkzeuge finden Sie in ldap.conf(5).

3.3 Einfache Verzeichnisoperationen

Die folgenden Beispiele setzen voraus, dass Sie Zugriff auf einen installierten OpenLDAP-Server haben. In einem Kurs wird Ihr Trainer dafür sorgen.

3.4 Daten suchen mit `ldapsearch`

Die einfachste Operation ist das Suchen nach Einträgen in einem LDAP-Verzeichnis. Hierzu dient das Kommando ldapsearch. Ein Aufruf der Form

```
$ ldapsearch -x
```

zum Beispiel gibt alle Einträge aus, die im Verzeichnisbaum unter dem DN »dc=example,dc=com« enthalten sind:

```
$ ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
dn: dc=example,dc=com
```

Tabelle 3.1: Ausgabeformate für ldapsearch

Option	Ausgabe
keine	Ausgabe in einem erweiterten LDIF-Format
-L	Ausgabe im LDIFv1-Format
-LL	Kommentare werden unterdrückt
-LLL	Kommentare und LDIF-Versionsnummer werden unterdrückt

```
objectClass: organization
objectClass: dcObject
dc: example
o: Beispiel
description: Beispiel

# postmaster,example.com
◁◁◁◁◁
```

Die Option -x wird hier gebraucht, um »einfache« (PLAIN) Authentisierung zu aktivieren. (Für Ihren LDAP-Server kann das richtig sein, muss aber nicht – sollte Ihr LDAP-Server Sie kalt lächelnd abblitzen lassen, versuchen Sie etwas wie

```
$ ldapsearch -x -D "cn=Hugo Schulz,dc=example,dc=com" -W
```

Hierbei gibt die Option -D einen DN an, mit dem Sie sich beim Verzeichnis authentisieren wollen – das moralische Äquivalent zu einem Linux-Benutzernamen –, und die Option -W sorgt dafür, dass ldapsearch Sie nach dem dazu passenden Kennwort fragt.)

Wenn Sie die Option -x weglassen, nimmt ldapsearch an, dass Sie eine SASL-basierte Authentisierung vornehmen möchten. Das geht bei einem nicht dafür konfigurierten OpenLDAP-Server schief, aber wenn Ihr LDAP-Server Sie mit PLAIN-Authentisierung nicht zuläßt, dann könnten Sie es durchaus mal ohne das -x probieren und schauen, was passiert.

Das Ergebnis wird in einem Format namens LDIF (»LDAP Data Interchange Format«) ausgegeben, dessen Details wir hier aus Platzgründen nicht diskutieren können (zum Glück erklärt es sich weitgehend selbst).

Wenn Ihr Verzeichnisbaum nicht bei »dc=example,dc=com« anfängt, können Sie mit der Option -b einen alternativen Basis-DN angeben:

```
$ ldapsearch -x -b dc=linupfront,dc=de
```

Das funktioniert natürlich auch, wenn Sie nur in einem Teilbaum des DIT suchen wollen.

Versuchen Sie das Beispiel besser nicht mit dem Verzeichnisbaum Ihres multinationalen Unternehmens mit drei Millionen Einträgen.

Die Option -L gibt Ihnen eine gewisse Kontrolle über die Details des Ausgabeformats; Tabelle 3.1 enthält einen Überblick.

Das erste ldapsearch-Argument ist ein Filter, der das Resultat auf bestimmte Einträge einschränken kann: Filter

```
$ ldapsearch -LLL -x 'cn=Hugo Schulz'
dn: cn=Hugo Schulz, dc=example, dc=com
objectClass: person
cn: Hugo Schulz
sn: Schulz
```

gibt den Eintrag für den Benutzer Hugo Schulz aus.

Weitere Argumente benennen die Attribute, die ausgegeben werden sollen:

```
$ ldapsearch -LLL -x 'cn=Hugo Schulz' sn
dn: cn=Hugo Schulz, dc=example, dc=com
sn: Schulz
```

3.5 Daten hinzufügen, ändern und löschen

Suchen können im Verzeichnis ist eine nette Sache, aber hin und wieder ist es auch nötig, ins Verzeichnis zu schreiben. Auch dafür gibt es Werkzeuge, von denen wir im Folgenden ein paar vorstellen.

Daten hinzufügen Mit `ldapadd` können Sie neue Daten in ein LDAP-Verzeichnis aufnehmen. Diese Daten müssen im LDIF vorliegen, typischerweise in einer Datei, und können dem Verzeichnis zum Beispiel wie folgt hinzugefügt werden:

```
$ ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f beispiel.ldif
```

Hier binden wir uns mit dem DN des Administrators, `cn=Manager,dc=example,dc=com`, an das Verzeichnis (das `-W` erwartet wie bei `ldapsearch`, dass Sie auf Anfrage das Kennwort eingeben, und `-x` verlangt PLAIN-Authentisierung), was für unser einfaches Beispiel stimmt, aber in der freien Wildbahn natürlich nicht notwendigerweise zutreffen muss.

Mit der Option `-w` können Sie das Kennwort auch direkt auf der Kommandozeile angeben (etwa »`-w geheim`«). Das ist aber nicht empfehlenswert.

Die Einträge in der Datei `beispiel.ldif` dürfen im Verzeichnis noch nicht existieren, sonst gibt es eine Fehlermeldung. Sie sollten auch darauf achten, den DIT »von oben nach unten« aufzubauen; wenn Sie zum Beispiel den Eintrag `cn=Hugo Schulz,ou=People,dc=example,dc=com` hinzufügen wollen, dann muss in dem betreffenden Moment der Eintrag `ou=People,dc=example,dc=com` schon existieren, sonst gibt es ebenfalls eine Fehlermeldung.

Daten ändern Daten im Verzeichnis ändern können Sie mit dem Kommando `ldapmodify`. Ähnlich wie `ldapadd` übernimmt es Verzeichnisdaten gemäß LDIF, diesmal allerdings in einem »Folge von Operationen«-Format. Auch bei `ldapmodify` entsprechen die Optionen, die sich zum Beispiel mit Authentisierung befassen, denen von `ldapsearch`.

Genau genommen ist `ldapadd` nichts anderes als »`ldapmodify -a`«. Die beiden Kommandos werden tatsächlich vom selben Programm implementiert, das sich anhand seines Aufrufnamens für eine der beiden Geschmacksrichtungen entscheidet.

Daten löschen Zum Löschen dient das Kommando `ldapdelete` – zwar können Sie das auch mit `ldapmodify`, aber `ldapdelete` übernimmt die DNs der zu entfernenden Einträge direkt auf der Kommandozeile (oder der Standardeingabe, wenn keine auf der Kommandozeile stehen) und enthebt Sie daher der Pflicht, LDIF-Dateien mit den entsprechenden Löschanforderungen zu konstruieren:

```
$ ldapdelete -x "uid=goner,dc=example,dc=com"
```

Kennwörter ändern Kennwörter können Sie im Verzeichnis mit dem Kommando `ldappasswd` ändern. Sie könnten zum Beispiel versuchen, das Kennwort für den Benutzer `cn=Hugo Schulz,dc=example,dc=com` zu setzen. Um Schwierigkeiten zu vermeiden, machen Sie das am besten als Verzeichnis-Administrator:

```
$ ldappasswd -W -x -D "cn=Manager,dc=example,dc=com" \
>   -S "cn=Hugo Schulz,dc=example,dc=com
New password: abc123
Re-enter new password: abc123
Enter LDAP Password: geheim                                    Für den rootdn
```

(Die Option `-S` fragt das Kennwort interaktiv ab – wie üblich zweimal. Schlagen Sie in `ldappasswd`(1) nach, um die anderen Möglichkeiten zum Setzen von Kennwörtern zu finden.)

Kommandos in diesem Kapitel

`ldapadd` Fügt Einträge aus einer LDIF-Datei in ein LDAP-Verzeichnis ein `ldapadd`(1) 50

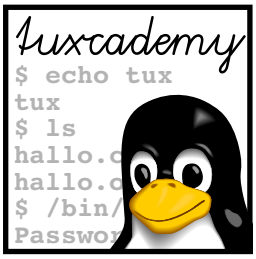
`ldapdelete` Löscht Einträge aus einem LDAP-Verzeichnis `ldapdelete`(1) 50

`ldapmodify` Ändert Einträge in einem LDAP-Verzeichnis anhand einer LDIF-Datei `ldapmodify`(1) 50

`ldappasswd` Ändert das Kennwort eines LDAP-Eintrags `ldappasswd`(1) 50

Zusammenfassung

- LDAP ist ein Verfahren, mit dem Rechner auf entfernte Verzeichnisdienste (engl. *directory services*) zugreifen können.
- Ein Linux-Rechner kann einen LDAP-Verzeichnisdienst für unterschiedliche Zwecke nutzen, etwa für die Benutzerverwaltung und -authentisierung oder zur Unterstützung von Mail-Programmen.
- Für die Konfiguration eines LDAP-Clients brauchen Sie normalerweise mindestens einen URL für den LDAP-Server und einen Basis-DN für Suchoperationen.
- Die kanonischen LDAP-Kommandozeilenwerkzeuge werden systemweit über die Datei `/etc/ldap/ldap.conf` (mitunter auch `/etc/openldap/ldap.conf`) konfiguriert.
- Sie lesen außerdem noch die Datei `.ldaprc` im Heimatverzeichnis des aufrufenden Benutzers und die Datei `ldaprc` im aktuellen Verzeichnis.
- Die wichtigsten Einträge in `ldap.conf` sind `URI` und `BASE`.
- `ldapsearch` erlaubt die Suche in LDAP-Verzeichnissen.
- `ldapadd`, `ldapmodify`, `ldapdelete` und `ldappasswd` dienen zur Manipulation von Verzeichnisinhalten.



4

Einführung in Samba

Inhalt

Lernziele

- Samba kennenlernen
- Einfache Verzeichnisfreigaben mit Samba erstellen können

Vorkenntnisse

- Ein Linuxsystem in ein lokales Netz einbinden könne
- Grundlegendes Wissen über Dateizugriffe in Netzen
- Kenntnisse aus der Windows-Welt sind hilfreich

4.1 Dateizugriffsverfahren in Netzen

gemeinsame Nutzung von Sekundärspeicher

Eine der Hauptaufgaben eines Rechnernetzes ist die gemeinsame Nutzung von Ressourcen. Dazu gehören insbesondere Daten auf Sekundärspeicher wie Platten oder RAID-Systemen. Für die gemeinsame Nutzung von Sekundärspeicher gibt es transparente und nicht-transparente Methoden. »Transparent« bezieht sich dabei darauf, ob ein Benutzer bewußt eine Verbindung zu einem anderen Rechner aufbaut oder ob der Netzzugriff vom Benutzer unbemerkt passiert. Typische nicht-transparente Methoden sind FTP oder die »Secure Shell« (SSH) mit dem `scp`-Programm.

Samba

Dieser Kurs beschäftigt sich mit transparenten Methoden zur gemeinsamen Nutzung von Sekundärspeicher. In erster Linie geht es um Samba, eine freie Software-Lösung zur Erzeugung von Windows-Freigaben auf Unix-artigen Betriebssystemen. Als klassische Unix-Lösung für die gemeinsame Sekundärspeichernutzung werden Sie das *Network File System* NFS kennenlernen.

NFS

Gemeinsame Nutzung von Druckern

Eine zweite Ressource, die für die gemeinsame Nutzung durch mehrere Rechner prädestiniert ist, stellen Drucker dar. Im Rahmen des Samba-Teils der Unterlage erfahren Sie, wie Sie mittels Samba Drucker netzwerkweit zur Verfügung stellen können, auch wenn sie an Unix/Linux-Maschinen angeschlossen sind.

LPI-Prüfung 201

Diese Kursunterlage soll unter anderem Kenntnisse vermitteln, die für das Bestehen der LPI-Prüfung 201 notwendig sind. Seit Anfang 2006 setzen die Prüfungsziele sowohl Samba 2 als auch Samba 3 voraus; da Sie heutzutage sowieso Samba 3 einsetzen sollten (Samba 2 ist offiziell abgekündigt), redet diese Unterlage vornehmlich über Samba 3. Dort, wo es nötig ist, gehen wir jedoch auch auf die Unterschiede zu Samba 2 ein.

4.2 Was ist Samba?

Samba

Samba ist eine Implementierung des SMB-Protokolls und ermöglicht damit:

- die gemeinsame Verwendung von Festplatten und Druckern im lokalen Netz
- benutzerspezifische Authentisierung und Rechteverwaltung
- Namensauflösung für Windows-Netze (WINS)
- Vorhalten von Ressourcenlisten für Windows-Netze (Suchdienst)

SMB kommt hauptsächlich in Windows-Netzen zum Einsatz – mit Samba existiert eine komfortable Möglichkeit, SMB auch mit anderen Betriebssystemplattformen, z. B. unter Unix oder VMS zu verwenden. Samba ist damit die erste Wahl, wenn es darum geht, gemeinsam genutzte Ressourcen in heterogenen Netzwerken bereit zu stellen. Samba ist freie Software und unterliegt der GPL.

Geschichte

Die erste Version von Samba wurde 1992 von Andrew Tridgell entwickelt. Der Australier wollte ein Programm unter DOS mit Zugriff auf die Netzwerkschnittstelle (genauer gesagt NetBIOS-Schnittstelle) laufen lassen und gleichzeitig einen NFS-Client für DOS verwenden. Das ist unter DOS nicht möglich, er versuchte daher das Programm unter Unix zum Laufen zu bringen. Dazu brauchte er aber eine NetBIOS-Implementierung für Unix. Durch *Reverse Engineering* mit Hilfe eines Paketsniffers entwickelte er eine solche. Jetzt konnte er SMB-Freigaben auf seinem Unix-Rechner erzeugen und kam so ohne den NFS-Client unter DOS aus.

smbd

Tridgell verwendete die neue Software erst zwei Jahre später wieder, diesmal auf einem Linux-System. Er entwickelte das Projekt unter dem Namen »smbd« weiter. Wegen Probleme mit dem Eigentümer der Namensrechte an »SMB« (wer das wohl sein kann?) war er später gezwungen, das Projekt umzubenennen – ein `grep` über die Wortliste einer Rechtschreibprüfung nach Wörtern mit den Buchstaben »S«, »M« und »B« in dieser Reihenfolge förderte »Samba« zu Tage.

Mittlerweile arbeitet ein Kernteam aus ca. 30 Personen weltweit an Samba und hat es zu dem gemacht, was es jetzt ist – einer Standard-Anwendung und einem der Aushängeschilder freier Software.

Warum sollten Sie Samba einsetzen wollen? Einige Gründe dafür:

- Sie benötigen die Funktionalität eines Windows-basierten Servers, aber eine Windows-Serverlizenz ist zu teuer. (Für Samba sind keine Lizenzgebühren fällig, egal wie viele Server Sie damit betreiben.)
- Die für einen Windows-basierten Server benötigten Client-Zugriffslizenzen sind zu teuer. (Für Samba sind keine Client-Zugriffslizenzen notwendig.)
- Sie möchten eine schrittweise Migration von Windows auf Linux durchführen und brauchen dafür die Möglichkeit gemeinsam benutzten Sekundärspeichers. (Der typische Ansatz besteht darin, zunächst – für die Clients transparent – die Server auf Linux zu migrieren. Später werden dann die Clients nachgeholt.)
- Sie möchten Linux- und Windows-Benutzerverwaltung und -Authentisierung vereinheitlichen. (Samba gibt Ihnen die Möglichkeit, ein LDAP-Verzeichnis zur Benutzerdatenhaltung für Windows und Linux einzusetzen – eine effiziente, skalierbare und dank der verschiedenen frei verfügbaren, hochqualitativen LDAP-Server auch kostengünstige Lösung.)
- Sie wünschen eine Stabilität und Performance, die mit Windows-basierten Systemen nicht zu erreichen ist. (Mit Samba und Unix haben Sie potentiell Zugriff auf hochleistungsfähige Hardwareplattformen, auf denen Windows nicht zur Verfügung steht. Selbst auf identischer Hardware bietet Linux mit Samba normalerweise höhere Dateiserver-Performance als Windows.)

4.3 Versionen und Bestandteile von Samba

Die aktuelle Samba-Version (im Sommer 2010) ist Samba 3. Samba 3 enthält wesentliche Neuerungen zur Vorgängerversion Samba 2.2, unter anderem:

- Erst mit Samba 3 ist es wirklich möglich, einen *Primary Domain Controller* für eine Windows-NT-Domäne aufzusetzen, inklusive *Trusted Domains*, etc.
- Ein Samba 3-Server kann Mitgliedsserver einer Windows-2000/2003-Domäne werden, da *Active Directory* (ADS) unterstützt wird.
- Samba 3 ist modular erweiterbar. Besipielsweise müssen Sie Samba nicht mehr neu übersetzen, wenn Sie die LDAP-Unterstützung verwenden möchten. Das Laden des Moduls beim Start reicht aus.
- Samba 3 verwendet standardmäßig Unicode für die Zeichencodierung, auch eine Konvertierung findet automatisch statt.
- Zu den bisherigen Samba-Werkzeugen sind `net` (bekannt aus der Windows-Welt) und `rpcclient` hinzugekommen.

Samba 2.2 sollten Sie allerdings heutzutage nicht mehr verwenden.

Die aktuelle Version von Samba (August 2010) ist Samba 3.5.4. Samba 3.6 befindet sich in Vorbereitung.

Die nächste Version, Samba 4, beginnt sich bereits am Horizont abzuzeichnen – die ersten Alpha-Testversionen sind bereits erschienen. Bis Samba 4 produktiv eingesetzt werden kann, wird es aber noch ein bisschen dauern. Der wesentliche für Sie als Administrator sichtbare Fortschritt gegenüber Samba 3 ist die Möglichkeit, ein Active Directory auf Samba-Basis implementieren zu können.

Samba besteht als Implementierung des SMB-Protokolls nicht nur aus einem Programm, sondern aus einer ganzen Sammlung. Hier eine kurze Übersicht über die einzelnen Programme. Viele davon werden in den späteren Kapiteln ausführlich besprochen. Im Wesentlichen muss zwischen Serverprogrammen und sonstigen Hilfs- und Clientprogrammen unterschieden werden. Als Serverprogramme gelten:

Serverprogramme

nmbd Der `nmbd` dient vor allem der NetBIOS-Namensauflösung, aber kümmert sich auch um die Suchdienst-Protokolle (die »Netzwerkumgebung« von Windows). Außerdem kann er als WINS-Server fungieren. Wenn Sie ihn verwenden wollen (meistens der Fall), sollten Sie ihn vor `smbd` starten.

smbd Der `smbd` ist die eigentliche Implementierung von SMB und damit für Authentisierung, Datentransfer und Ähnliches zuständig.

winbindd Möchten Sie Ihren Sambaserver in eine Windows-Domäne oder ein ADS integrieren, benötigen Sie dafür den `winbindd`. Er ermöglicht Windows-Benutzern, die sich an der Domäne angemeldet haben, den Zugriff auf den Samba-Server, ohne dass dort entsprechende Linux-Benutzer angelegt werden müssen. `winbindd` wird dabei (ähnlich wie z. B. NIS) in die Datei `/etc/nsswitch.conf` als Quelle für Benutzerdaten eingetragen; ebenso ist über PAM eine Linux-Anmeldung mit Authentisierung in der Domäne möglich. Auch für die Unterstützung von *Trusted Domains* ist `winbindd` unumgänglich. Diese Funktionalität ist ab Samba 3 integriert.

Außer den Server-Diensten enthält eine Samba-Installation üblicherweise folgende Hilfsprogramme:

Hilfsprogramme

smbclient Eine Client-Implementierung von SMB; erlaubt den Zugriff auf SMB-Freigaben, unter anderem auch das Drucken auf Druckerfreigaben. Auf Dateien können Sie über Kommandos ähnlich denen des `ftp`-Programms zugreifen.

smbmount Eine weitere Client-Implementierung von SMB, die es Ihnen ermöglicht, SMB-Freigaben unter Linux direkt als Dateisysteme einzuhängen (zu »mounten«). Andere Unix-Systeme bieten diese Möglichkeit oft nicht an; hier sind Sie für Dateizugriffe auf den `smbclient` angewiesen.

nmblookup Ermöglicht NetBIOS-Namensabfragen, ähnlich dem DNS-Werkzeug `host`.

smbpasswd Mit diesem Programm können Sie SMB-Benutzer anlegen und ihre Kennwörter verwalten.

testparm Überprüft Ihre Samba-Konfiguration auf syntaktische und zum Teil auch konzeptionelle Fehler.

smbstatus Zeigt aktuelle Zugriffe auf einen Samba-Server an.

smbtree Eine kommandozeilenbasierte »Netzwerkumgebung«.

rpcclient Ermöglicht das Ausführen von MS-RPC-Kommandos auf entfernten SMB-Rechnern. Dazu gehört unter anderem das Herunterfahren entfernter Rechner. Nur Samba 3!

net Entspricht im Wesentlichen dem `net`-Befehl der Windows-Welt. Wird unter anderem für den Beitritt einer Domäne benötigt. Nur Samba 3!

swat `swat` ist ein Netzwerkdienst, der üblicherweise über den `(x)inetd` gestartet wird und die webbasierte Administration eines Samba-Servers ermöglicht.

Bei den beschriebenen Programmen handelt es sich lediglich um die wichtigsten. Diese und alle weiteren Programme einer Samba-Installation werden in der Handbuchseite `samba`(7) aufgeführt.

Übungen

4.1 [!1] Was sind die gerade aktuellen Samba-Versionen (möglicherweise hat sich seit der Drucklegung dieser Unterlage ja etwas getan)? Schauen Sie auf `www.samba.org` nach.

4.2 [!1] Welche Samba-Version ist in Ihrer Distribution enthalten? Schauen Sie nach, etwa mit `rpm` oder `dpkg`.

4.4 Samba-Dokumentation

Als freies Software-Projekt ist Samba nicht nur wegen des offenen Quellcodes außerordentlich gut dokumentiert. Erster Anlaufpunkt für Samba-Dokumentation ist `http://www.samba.org/samba/docs/`. Hier finden Sie eine Vielzahl von Dokumenten. Dazu gehören:

- Die Handbuchseiten der verschiedenen Samba-Programme.
- Das offizielle Samba-3-HOWTO- und Referenzhandbuch [TV03], siehe auch `http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/`. Hierbei handelt es sich um ein englisches Dokument, das Samba unter anderem mit Hilfe praktischer Beispiele erklärt. Leider merkt man dieser Dokumentation an, dass es sich um eine »*Collection*« handelt, durch das Zusammenstückeln diverser Einzeltexte wird der Lesefluss etwas gehemmt. Insbesondere als PDF ein gutes Nachschlagewerk, weniger eine Nachttischlektüre zum Einstieg in Samba.
- Dasselbe Dokument finden Sie unter `http://gertranssmb3.berlios.de/` auch als deutsche Übersetzung, ebenso im Buchhandel als [TV05].
- Ebenfalls sehr praktisch orientiert und eher als durchgehende Lektüre gedacht ist *Samba-3 by Example* von John H. Terpstra [Ter04], einem der Samba-Entwickler. Dieses Buch bekommen Sie natürlich im Buchhandel, aber auch kostenlos unter `http://www.samba.org/samba/docs/man/Samba-Guide/`.
- Ein (leider nicht mehr ganz aktuelles) HOWTO ist das *Unofficial Samba Howto*, gedacht als schnelle Anleitung zum Aufsetzen eines Samba-Servers, zu finden unter `http://www.samba.org/samba/docs/unofficial-samba-howto.html`
- Eine Art Standardwerk zum Thema Samba in Buchform ist *Using Samba* von Jay Ts, Robert Eckstein und David Collier-Brown, erschienen im O'Reilly-Verlag [TECB03b]. Auch dieses Buch gibt es im WWW, allerdings nicht in der neuesten Version, unter `http://www.samba.org/samba/docs/using_samba`. Die deutsche Übersetzung ist [TECB03a].

Auch anderswo im Internet oder im Buchhandel gibt es Interessantes:

- Eine schöne Einleitung in die Theorie, also in die Welt der Protokolle SMB/CIFS ist *Implementing CIFS* von Christopher R. Hertel, erschienen bei *Prentice-Hall* [Her03]. Auch dieses Buch wieder im WWW unter `/www.ubiqx.org/cifs/`.
- Nicht in elektronischer Form erhältlich, dafür aber in deutscher Sprache und ziemlich aktuell ist *Samba 3* von Jens Kühnel, erschienen beim mitp-Verlag [Kü04].
- Nicht zuletzt bietet auch das Webfrontend `swat` sehr brauchbare Informationen zu den einzelnen Konfigurationsparametern eines Samba-Servers.

4.5 Installation von Samba

zwei Möglichkeiten

Wie in der Welt der freien Software üblich, haben Sie zwei Möglichkeiten für den Weg zur eigenen Samba-Installation; Einerseits können Sie die Programme selbst aus dem Quelltext übersetzen oder Sie verwenden das Paket-Management-System Ihrer Linux-Distribution, um Samba in vorgefertigter Form zu installieren. Letzteres ist einfacher und für den Einsteiger durchaus empfehlenswert. Für eine Firmenumgebung mit speziellen Ansprüchen ist das Selbst-Übersetzen die bessere Lösung. Nur so können Sie besondere Eigenschaften in Samba integrieren und nicht erwünschte Eigenschaften außen vor lassen, was der Sicherheit und Leistung dienlich sein kann.

Paket-Management-System

Auf das Installieren von Samba mit dem Paket-Management-System Ihrer Linux-Distribution soll hier nicht weiter eingegangen werden. Entsprechende Pakete liegen Ihrer Distribution entweder bei oder können von `http://www.samba.org/samba/ftp/Binary_Packages/` bezogen werden.

Für die Installation aus dem Quellcode benötigen Sie zuerst den Quellcode – `ftp://de.samba.org/samba.org/`. Dort finden Sie das gepackte `tar`-Paket der von Ihnen gewünschten Version, z. B. `samba-3.5.4.tar.gz`. Dieses müssen Sie auspacken:

```
$ tar -xzf samba-3.5.4.tar.gz -C /tmp
```

Wechseln Sie danach in das entstehende Verzeichnis, genauer gesagt in das Unterverzeichnis `source`:

```
$ cd /tmp/samba-3.5.4/source/
```

Konfiguration, Übersetzung und Installation erfolgen wie bei heutiger freier (Linux-)Software üblich mit `configure`, `make` und `make install`. Für das Erzeugen des `Makefile` reicht prinzipiell ein einzelnes `./configure`, dafür machen Sie sich aber nicht die Mühe – Das `configure`-Skript ermöglicht diverse Anpassungen, die Sie mit `./configure --help` aufrufen können. Besonders wichtig sind die Wahl der Installationspfade (`prefix=`⟨*Verzeichnis*⟩ und `*dir=`⟨*Verzeichnis*⟩) sowie die Aktivierung und Deaktivierung bestimmter Eigenschaften:

`--with-ldap` Kompiliert Samba mit LDAP-Unterstützung – außerordentlich wichtig beim Einsatz in MS-ADS und für die Implementierung größerer Domänenstrukturen.

`--with-ads` Kompiliert Samba mit ADS-Unterstützung.

`--with-krb5=`⟨*Verzeichnis*⟩ Kompiliert Samba mit Kerberos-Unterstützung (wichtig für ADS) und gibt gleichzeitig an, wo Kerberos installiert ist.

`--with-pam` Kompiliert Samba mit PAM-Unterstützung für die Benutzerauthentisierung. Bei modernen Linux-Systemen sollte das Standard sein.

`--with-quotas` Ermöglicht den Einsatz von Samba mit Disk-Quota-Unterstützung. (Damit kann der Festplattenplatz für einzelne Benutzer eingeschränkt werden.)

`--with-acl-support` ACL-Unterstützung benötigen Sie unter anderem, wenn Sie es Windows-Benutzern ermöglichen wollen, erweiterte Windows-Rechte (ACLs) für Dateien auf dem Samba-Server zu vergeben.

`--with-shared-modules=`⟨*Modulliste*⟩ Ermöglicht es Ihnen, eine Reihe von Funktionen als ladbare Module zu kompilieren – sehr empfehlenswert.

`--with-smbmount` Auch das Programm `smbmount` wird miterzeugt. Diese Option steht stellvertretend für andere; nicht alle Bestandteile von Samba werden automatisch in die Übersetzung eingeschlossen, häufig müssen sie von Hand ausgewählt werden.

Haben Sie mit `./configure` erfogreich ein `Makefile` erstellt, können Sie Samba übersetzen und installieren:

```
$ make
<<<<<
$ /bin/su
# make install
<<<<<
```

Samba ist eines der »wohlerzogenen« Programmpakete, die eine Deinstallation erlauben, auch wenn Sie das Paket aus dem Quellcode installiert haben. Dazu müssen Sie lediglich ein `make uninstall` im Quellcode-Verzeichnis, respektive im dort befindlichen `source`-Unterverzeichnis ausführen. Alternativ können Sie aus dem Quellcode auch ein eigenes RPM- oder Debian-Paket erstellen, was die Verwaltung der Software insbesondere auf mehreren Rechnern gleichzeitig stark vereinfacht. Informationen über den Selbstbau von Software-Paketen bekommen Sie uner anderem in der Linup-Front-Schulungsunterlage *Linux-Systemanpassungen – Kernel, Software und Hardware*.

Übungen

4.3 [*4] Auch wenn Sie es als Benutzer einer Linux-Distribution eigentlich nicht müssten: Versuchen Sie sich mal an einer Installation von Samba aus dem Quellcode, ähnlich wie eben diskutiert. Vor allem, wenn Sie das LPIC-2-Zertifikat anstreben, ist dies eine sinnvolle Übung.

4.6 Starten der Samba-Server-Programme

Auf einem einfachen Samba-Server, der lediglich Windows-Freigaben im Netzwerk anbietet, muss theoretisch nur der `smbd` laufen. Beim `smbd` handelt es sich um die eigentliche Implementierung von SMB; mit ihm können Sie Verzeichnisse und Drucker freigeben, sowie die Authentisierung der Benutzer durchführen. Möchten Sie außerdem eine NetBIOS-Namensauflösung anbieten oder durchführen und/oder am Suchdienst für eine NT-Domäne bereitstellen, müssen Sie auch den `nmbd` starten – und zwar vor dem `smbd`! Für das Anbinden an eine Windows-NT- bzw. eine Windows-2000/2003-ADS-Domäne brauchen Sie zusätzlich den `winbindd`.

Am besten starten Sie auf jeden Fall `nmbd` und `smbd`. Der Verzicht auf den `nmbd` bringt keinen großen Effizienzgewinn und kann sich außerdem nachteilig auf die Funktion des `smbd` auswirken.

Start mit dem Internet-Daemon

Alle drei Dienste können sowohl als freistehende Dienste als auch über den »Internet-Daemon« `inetd` bzw. `xinetd` gestartet werden. Der Start mit dem Internet-Daemon ist nur dann empfehlenswert, wenn nur selten Zugriffe auf den Server zu erwarten sind, da Samba bei jedem Zugriff erst gestartet werden muss. In fast allen Fällen bedeutet das eine zu große Zeitverzögerung. Wie das Starten über den `inetd/xinetd` im Detail funktioniert, erfahren Sie aus der Linup-Front-Schulungsunterlage *Linux-Netzwerkadministration I* oder den entsprechenden Handbuchseiten `xinetd`(8) und `xinetd.conf`(5) bzw. `inetd`(8) und `inetd.conf`(5).

Meistens werden die Samba-Dienste als freistehende Dienste gestartet. Auch hier gibt es wieder zwei Möglichkeiten; entweder Sie starten die Dienste jedesmal von Hand oder Sie schreiben den Aufruf der Programme in ein Shellskript. Letzteres ist dringend zu empfehlen, denn als Init-Skript in die Runlevel-Konfiguration Ihres Unix/Linux-Systems integriert ermöglicht ein Shellskript den automatischen Start der Samba-Dienste beim Hochfahren des Rechners.

Aus irgendwelchen Gründen liefern manche Distributionen, etwa die von SUSE/Novell, für die beiden Daemons getrennte Init-Skripte mit, was in der Praxis den Umgang mit Samba eher erschwert statt erleichtert. Debian GNU/Linux zum Beispiel verwendet ein einziges Init-Skript für beide.

Starten von nmbd

Der nmbd wird mit dem Kommando

```
# nmbd -D
```

als Daemon im Hintergrund gestartet. Danach lauscht er auf den UDP-Ports 137 und 138. Die Konfiguration von nmbd erfolgt über die Datei smb.conf.

Starten von smbd

Den smbd können Sie analog zum nmbd mit dem Aufruf

```
# smbd -D
```

als Daemon im Hintergrund starten. Seine Ports sind in der Regel die TCP-Ports 139 und 445. Auch er wird über die Datei smb.conf konfiguriert.

Der smbd kennt einige interessante Kommandozeilenparameter (siehe auch smbd(8)):

-S bewirkt, dass der smbd alle Ausgaben auf der Standardausgabe ausgibt. Das ist für den ersten Start und die Fehlerbehebung durchaus sinnvoll. Dazu muss der Daemon allerdings im Vordergrund gestartet werden

-i ermöglicht es Ihnen den smbd im Vordergrund (also nicht als Daemon) zu starten. -i impliziert außerdem die Option -S, alle Ausgaben landen also auf der Standardausgabe.

-d *n* ermöglicht es Ihnen die Geschwätzigkeit des smbd einzustellen. Mit einem Wert von 0 für *n* ist er quasi stumm während ein Wert von 10 wirklich nur zum Debuggen sinnvoll ist. Ein vernünftiger Wert für den täglichen Gebrauch ist 1; die Werte 2 und 3 liefern ausführlichere Daten für die Problemanalyse bei der Konfiguration, und höhere Werte sind nur für Samba-Entwickler zum Debugging interessant.

-b gibt aus, mit welchen Optionen smbd übersetzt wurde. Besonders interessant ist die Ausgabe von

```
# smbd -b | grep '^[ ]*WITH'
  WITH_UTMP
  WITH_ADS
  WITH_AUTOMOUNT
Elided
```

Init-Skript

Das Starten von Hand erfolgt üblicherweise nur zu Testzwecken. Normalerweise wird das Programm über ein entsprechendes Init-Skript gestartet und gestoppt (z. B. /etc/init.d/smb bei den Linux-Distributionen von Novell/SUSE, /etc/rc.d/smb bei RedHat/Fedora oder /etc/init.d/samba bei Debian GNU/Linux und Ubuntu). Das Init-Skript kann manuell aufgerufen werden:

```
# /etc/init.d/nmb start
Starting Samba NMB daemon                              done
# /etc/init.d/smb start
Starting Samba SMB daemon                              done
```

Automatischer Start

Im Rahmen der Runlevel-Konfiguration können Sie den Dienst automatisch beim Systemstart starten. Klassischerweise legen Sie dazu symbolische Links auf das Skript in die Konfigurationsverzeichnisse der entsprechenden Runlevel, z. B. für Runlevel 3:

```
# cd /etc/init.d/rc3.d/
# ln -s ../nmb S49smb
# ln -s ../smb K04smb
# ln -s ../smb S50smb
# ln -s ../smb K03smb
```

Sie müssen dabei darauf achten, dass der nmbd vor dem smbd gestartet und nach diesem gestoppt wird. Das können Sie über eine entsprechende Numerierung der S-Links (engl. *start*) und K-Links (engl. *kill*) realisieren.

Vorsicht: In Ihrer Distribution sind manuelle Änderungen dieser symbolischen Links möglicherweise verpönt – etwa bei aktuellen SUSE-Distributionen, wo das nur über einen Runlevel-Editor oder Programme wie insserv oder chkconfig passieren soll. Siehe dazu den nächsten Absatz.

Zur Vereinfachung bringen viele Linux-Distributionen ein Kommando mit, das das Anlegen der Links automatisiert. Statt der oben gezeigten Befehle tut es bei SUSE/Novell- oder Red-Hat-artigen Distributionen ein

```
# chkconfig nmb 3
nmb    0:off   1:off   2:off   3:on    4:off   5:off    6:off
# chkconfig smb 3
smb    0:off   1:off   2:off   3:on    4:off   5:off    6:off
```

Im nächsten Abschnitt erfahren Sie, wie Sie Samba – insbesondere die Daemons smbd und nmbd konfigurieren können. Ohne Konfiguration gibt es keine Freigabe und damit ergibt auch der Start des Dienstes keinen Sinn …

Übungen

4.4 [!2] Wie sind die Init-Skripte für Samba bei Ihrer Distribution organisiert? Schauen Sie sich in /etc/init.d um und versuchen Sie, ihren Inhalt zu verstehen.

4.5 [2] Wo innerhalb des Startvorgangs sollte Samba gestartet werden? Welche Systembestandteile müssen initialisiert sein und welche Dienste müssen laufen, damit Samba funktioniert?

4.7 Die Samba-Konfigurationsdatei smb.conf

Das Erstellen von Freigaben und damit die Konfiguration des Samba-Servers erfolgt über die Konfigurationsdatei smb.conf, die üblicherweise im Verzeichnis /etc/samba liegt. Es handelt sich um eine Textdatei, die von der Syntax her den »*.ini-Dateien« aus der Windows-Welt ähnelt.

Abschnitte

Die Datei besteht aus einzelnen Abschnitten, die jeweils durch den Namen des Abschnitts, eingefasst in eckige Klammern (»[« und »]«), eingeleitet werden. Ein Abschnitt endet am Beginn des nächsten Abschnitts oder am Ende der Datei. Leere Zeilen sowie Kommentarzeilen, die mit einer Raute (»#«) beginnen[1], werden ignoriert; Sie können sie zur Strukturierung der Datei verwenden. Jeder Abschnitt definiert eine (teilweise auch mehrere) Freigaben, wobei der Name des Abschnitts in den meisten Fällen dem Namen der Freigabe entspricht.

Den Namen einer Freigabe können Sie beliebig wählen. Allerdings müssen Sie dabei folgende Regeln berücksichtigen:

- Groß- und Kleinschreibung spielt keine Rolle.

[1] Achtung: das Kommentarzeichen gehört wirklich nur an den Anfang einer Zeile; »Kommentare«, die nicht am Anfang einer Zeile beginnen, sind keine und gelten als Bestandteil der Konfiguration

- Die Abschnitts-Namen `global`, `homes` und `printers` haben eine Sonderbedeutung und dürfen nicht anderweitig vergeben werden.
- Am besten verwenden Sie nur Buchstaben. Insbesondere Sonderzeichen wie »`$`« können Probleme verursachen. Leerzeichen sind zwar laut SMB-Protokoll erlaubt, sollten aber bei der Konfiguration eines Unix/Linux-Dienstes möglichst vermieden werden.

`global`-Abschnitt

Der `global`-Abschnitt legt mit seinen Parametern das allgemeine Verhalten der verschiedenen Samba-Programme fest. Hier können Sie beispielsweise den Namen und Kommentar des Rechners oder das Authentisierungsverfahren für SMB-Benutzer festlegen. Der `homes`-Abschnitt ermöglicht eine elegante Freigabe der Heimatverzeichnisse aller Benutzer. Der `printers`-Abschnitt ermöglicht die Freigabe von Druckerwarteschlangen und deren Konfiguration.

`homes`-Abschnitt

`printers`-Abschnitt

Variablen

Die Abschnitte selbst bestehen aus Variablen, die wie folgt definiert werden können:

```
# Mögliche Methoden der Variablenzuweisung in smb.conf
VARIABLE=Wert
VariAbLe=Wert
variable=Wert
     variable =   Wert
v ar  i   able= Wert
```

Für Samba ist es irrelevant, ob der Name der Variable groß oder klein geschrieben wird oder ob sich Leerzeichen vor oder hinter dem Gleichheitszeichen befinden. Selbst Leerzeichen innerhalb des Variablennamens werden toleriert. (Ob das Sinn ergibt, ist eine andere Frage – am besten halten Sie sich an die Schreibweise in der Samba-Dokumentation.) Selbstverständlich sind auch Einrückungen erlaubt.

Achtung: Bei dem *Wert* der Variablen ist es in der Regel *nicht* egal, ob Sie Leerzeichen einfügen oder große oder kleine Buchstaben verwenden – beispielsweise kann der Wert einer Variablen eine Pfadangabe enthalten, und wie Sie wissen, kennen Linux-Systeme da keine Toleranz.

Beim Wert der Variablen kann es sich um spezielle Angaben, wie etwa Benutzernamen oder Pfadangaben handeln. Viele Variable steuern aber einfache Ja-Nein-Entscheidungen. Hier haben Sie die Qual der Wahl, was den Wert angeht – Sie können sowohl `Yes` und `No` als auch `True` und `False` oder einfach `0` und `1` verwenden. Bei diesen Variablen ist auch im Wert die Groß- und Kleinschreibung irrelevant.

Makros

Im Wert der Variablen können Sie je nach Anwendung Makros der Form `%`⟨*Buchstabe*⟩ verwenden. Die Samba-Programme ersetzen diese bei der Auswertung von `smb.conf` durch passende Ersatzwerte. Einige häufig verwendete Makros sehen Sie in Tabelle 4.1.

Standardwerte

Viele Variable haben Standardwerte. Welche das jeweils sind, können Sie unter anderem der Handbuchseite `smb.conf`(5) entnehmen. Um für alle Freigaben einen abweichenden Standardwert zu setzen, müssen Sie diesen im `global`-Abschnitt angeben – das ist eine der Hauptaufgaben dieses Abschnitts.

Die Variablen in `smb.conf` sind entweder »global« oder auf einzelne Freigaben bezogen (dürfen aber, wie erwähnt, trotzdem im `global`-Abschnitt auftauchen, um Vorgaben zu definieren). Ob eine Variable zur einen oder anderen Klasse gehört, erfahren Sie aus `smb.conf`(5); die einzelnen Variablen sind dort entweder mit »`(G)`« oder »`(S)`« gekennzeichnet, kurz für *global* oder *share* (Freigabe).

Sollten Sie übrigens dieselbe Variable in der `smb.conf`-Datei oder gar im selben Abschnitt mehrmals verwenden, kommt die jeweils unterste Zeile in der Datei zum Zug. Anders formuliert: Die Datei wird von oben nach unten bearbeitet. Da-

Einlesen der Konfiguration

Tabelle 4.1: Makros in der `smb.conf`-Datei

Makro	Bedeutung
`%D`	Domänen- bzw. Arbeitsgruppenname
`%h`	DNS-Name des Samba-Rechners
`%L`	NetBIOS-Name des Samba-Rechners
`%M`	DNS-Name des zugreifenden SMB-Clients
`%m`	NetBIOS-Name des zugreifenden SMB-Clients
`%I`	IP-Adresse des zugreifenden SMB-Clients
`%d`	PID des `smbd`
`%T`	Die aktuelle Zeit
`%S`	SMB-Service-Name der Verbindung – hiermit kann eine Verbindung eindeutig spezifiziert werden
`%u`	Name des auf eine Freigabe zugreifenden Benutzers
`%H`	Heimatverzeichnis des auf eine Freigabe zugreifenden Benutzers

zu müssen Sie die Samba-Daemons übrigens im Gegensatz zu anderen Unix/Linux-Diensten nicht zwingen! Einmal pro Minute lesen `smbd` und `nmbd` die Konfigurationsdatei automatisch neu ein. Falls Ihnen das beim Einrichten der Programme zu lange dauert, tut es beim `smbd` auch ein `SIGHUP`, etwa mit

```
# killall -1 smbd
```

Bestehende Verbindungen bleiben davon unangetastet.

Der `nmbd` lässt sich von einem `SIGHUP` nicht so sehr beeindrucken, dass er seine Konfiguration neu liest – er gibt nur den Zustand seiner internen Namens-Datenbank in eine Datei aus (siehe `nmbd`(8)). Bei Konfigurationsänderungen hilft beim `nmbd` nur Warten oder Neustart.

Übungen

4.6 [2] Wie können Sie sich leicht ein Bild von den in Ihrer `smb.conf`-Datei definierten Variablen machen, ohne dass Leerzeilen oder Kommentarzeilen davon ablenken?

4.8 Erste Schritte: Einrichten einer einfachen Freigabe mit Samba

Im Folgenden soll beispielhaft aufgezeigt werden, wie einfach es ist, mit Samba simple Verzeichnisfreigaben zu erstellen. Ausgehend von der denkbar primitivsten Konfiguration bis hin zu spezielleren Einstellungen sollen Sie ein Gefühl dafür bekommen, wie Samba funktioniert. Die weiteren Einzelheiten, die für den praktischen Einsatz von Belang sind, werden Sie in den späteren Kapiteln kennen lernen.

Primitivste Konfiguration

Hier ist ein sehr einfaches Beispiel für eine `smb.conf`-Datei:

```
# Ein (sehr) einfaches Beispiel einer smb.conf-Datei
[global]

[test]
    path = /testfreigabe
```

Diese Beispielsdatei enthält zwei Abschnitte. Der erste Abschnitt (`[global]`) bezieht sich auf das allgemeine Verhalten des Servers; Im Beispiel ist er leer (und die im Programm fest vorgegebenen Standardwerte treten in Kraft). Sie müssen keinen `global`-Abschnitt anlegen, in der Praxis wird sich das allerdings nicht vermeiden lassen – irgendwie wollen Sie das Verhalten des Servers ja konfigurieren … Der zweite Abschnitt (`[test]`) ist der Eintrag für eine Freigabe namens `test`. Mit der Variablen `path` wird festgelegt, welches Verzeichnis freigegeben werden soll.

Nach dem Editieren der Datei sollten Sie sinnvollerweise noch das freizugebende Verzeichnis erstellen und mit etwas Inhalt füllen. Falls noch nicht geschehen, ist es jetzt an der Zeit, `nmbd` und `smbd` zu starten.

Anzeige von Freigaben

Zum Anzeigen der Freigabe können Sie das Kommando `smbclient` verwenden:

```
$ smbclient -NL 192.168.0.100
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.5.4]

        Sharename      Type      Comment
        ---------      ----      -------
        test           Disk
◁◁◁◁◁
```

`smbclient` ist eine von vielen Möglichkeiten, auf SMB-Freigaben zuzugreifen. Die Option `-L` zählt alle auf dem Rechner `192.168.0.100` sichtbaren Freigaben auf, die Option `-N` unterdrückt die in diesem Fall nicht notwendige Passwortabfrage.

Arbeitsgruppe `WORKGROUP`

`smbclient` zeigt den gefragten Rechner als Mitglied der Arbeitsgruppe (oder Domäne) `WORKGROUP`. Das ist ebenso wie »`Server=[Samba 3.5.4]`« die Standardeinstellung. Im `global`-Abschnitt können Sie speziellere Einstellungen vornehmen:

```
[global]
    workgroup = TEST
[test]
    path = /testfreigabe
    comment = Testfreigabe für den Samba-Kurs
```

Entsprechend anschaulicher sieht dann die Ausgabe von `smbclient` aus:

```
$ smbclient -NL 192.168.0.100
Anonymous login successful
Domain=[TEST] OS=[Unix] Server=[Samba 3.5.4]

        Sharename       Type      Comment
        ---------       ----      -------
        test            Disk      Testfreigabe für den Samba-Kurs
◁◁◁◁◁
```

Sehen können Sie die Freigabe jetzt, und mit `smbclient` ist es prinzipiell auch möglich, auf Dateien auf der Freigabe zuzugreifen:

```
$ smbclient -N //192.168.0.100/test
Anonymous login successful
Domain=[TEST] OS=[Unix] Server=[Samba 3.5.4]
tree connect failed: NT_STATUS_ACCESS_DENIED
```

Im vorliegenden Beispiel schlägt das allerdings fehl. Selbstverständlich ist es in der Samba-Standardkonfiguration nicht einfach möglich, auf irgendwelche Freigaben zuzugreifen. Samba kennt diverse Möglichkeiten, Zugriffe einzuschränken und Benutzer zu authentisieren. Auf dieses Thema kommen wir in Kapitel 7 zurück.

Einschränken von Zugriffen

Dieser Abschnitt erläutert lediglich die Konfiguration, die notwendig ist, damit ein anonymer (Gast-)Benutzer lesend auf den Inhalt einer Freigabe zugreifen kann:

Gastbenutzer

```
[global]
    workgroup = TEST
    security = share
    guest account = nobody
[test]
    path = /testfreigabe
    comment = Testfreigabe für den Samba-Kurs
    guest ok = yes
```

Der Parameter `guest ok = yes` erlaubt prinzipiell anonyme Zugriffe. Das alleine reicht aber noch nicht. Die letzte Instanz für Dateizugriffe ist das Unix-Dateisystem – und ob ein Zugriff auf das Unix-Verzeichnis der Freigabe erlaubt ist, hängt von der Unix-UID ab, unter der der Zugriff erfolgt. Also muss dem Gast-Benutzer noch ein Unix-Benutzerkonto zugeordnet werden. Dazu dient die Variable `guest account`, die einen Unix-Benutzernamen als Wert zugewiesen bekommt. Hierfür können Sie ein eigenes Benutzerkonto anlegen (der betreffende Benutzer muss sich nicht interaktiv anmelden können, kann also ohne Kennwort oder mit `/bin/false` als Login-Shell erzeugt werden) oder den Benutzer `nobody` verwenden, der schon für NFS die »Gastrolle« spielt.

`guest ok`

`guest account`

Je nachdem, wie Samba übersetzt wurde, ist `nobody` die Standardeinstellung für `guest account` und muss gar nicht angegeben werden.

Beachten Sie, dass `guest account` ein globaler, `guest ok` aber ein freigabespezifischer Parameter ist. Sie können also nur einen Unix-Benutzernamen für *alle* Freigaben mit Gastzugriff angeben.

Wie Sie noch sehen werden, ist es außerdem notwendig, die Variable `security = share` anzugeben, damit der Samba-Server nicht entsprechend der Standardeinstellung nach einem Benutzernamen und Kennwort fragt. Zugegebenermaßen ist das eigentlich nicht nötig, wenn Sie `smbclient` verwenden. Neuere Windows-Systeme haben aber Probleme, auf solche Freigaben zuzugreifen; Sie fragen immer nach einem Kennwort – was ein Problem ist, weil es keines gibt …

Mit `smbclient` können Sie jetzt wie mit einem FTP-Client auf die Freigabe zugreifen. Unter Windows müssten Sie den Pfad zur Freigabe in SMB-Manier angeben:

»URL« für Freigaben

```
\\⟨Rechnername⟩\⟨Freigabename⟩
```

Unter Linux/Unix dagegen gilt der *Backslash* in der Regel als Maskierungszeichen, deshalb wird hier zur Definition der Freigabe der einfache Schrägstrich verwendet:

```
//⟨Rechnername⟩/⟨Freigabename⟩
```

Im Beispiel sähe das dann so aus:

```
$ smbclient -N //192.168.0.100/test
Anonymous login successful
Domain=[TEST] OS=[Unix] Server=[Samba 3.5.4]
smb: \> ls
  .                           D        0  Tue Jan 11 10:03:51 2005
  ..                          D        0  Tue Jan 11 10:03:41 2005
  profile                             6878  Tue Jan 11 10:03:51 2005

                53589 blocks of size 65536. 11353 blocks available
smb: \> get profile
```

```
smb: \\> quit
$ ls -lh profile
-rw-r--r--  1 tux users 6,8K 2005-01-11 10:48 profile
```

Mit »ls« lassen Sie sich den Inhalt der Freigabe anzeigen, mit »get« übertragen Sie eine Datei vom entfernten Rechner in Ihr momentanes lokales Verzeichnis, mit »put« können Sie Dateien auf den entfernten Rechner schieben und mit »quit« das Programm verlassen.

Auf Linux-Rechnern gibt es statt des etwas umständlichen smbclient auch die Möglichkeit, SMB-Freigaben direkt (auf Kernel-Ebene) ins Dateisystem einzuhängen, so dass Sie auf die Freigabe im Wesentlichen so zugreifen können, als läge sie auf einer lokalen Platte des Rechners. Grundlage dafür ist das smbmount-Kommando, mit dem Sie ähnlich wie mit mount eine Freigabe wie ein Dateisystem einbinden können:

```
# smbmount //192.168.0.100/test /mnt -o guest
# ls /mnt
profile
```

smbmount unterstützt diverse Optionen für das Einhängen, etwa um Benutzername und Kennwort anzugeben oder einen lokalen Benutzer anzugeben, dem alle Dateien auf dem eingehängten Dateisystem gehören sollen. Diese Optionen werden *hinten* ans Kommando angehängt und bestehen (meist) aus einer durch Kommata getrennten Folge von Angaben der Form »⟨*Schlüssel*⟩=⟨*Wert*⟩«, zum Beispiel:

```
# smbmount //192.168.0.100/test /mnt -o username=hugo,password=geheim
```

Eine ausführliche Liste aller Optionen finden Sie in smbmount(8).

Gängiger ist statt smbmount allerdings der direkte Aufruf von mount mit der Option »-t smbfs«. Sie können dabei alle Optionen aus mount(8) und smbmount(8) angeben:

```
# mount -t smbfs -o guest,ro //192.168.0.100/test /mnt
```

Damit können Sie natürlich auch in /etc/fstab auf SMB-Freigaben verweisen.

Der letzte Schrei ist »mount -t cifs« statt »-t smbfs«. Hiermit greifen Sie auf eine Neuimplementierung des SMB-Dateisystems im Linux-Kernel zu, das an die Eigenschaften ganz moderner Server (etwa Samba) angepasst ist und mit diesen besser zusammenarbeiten kann. Dazu gehören zum Beispiel Verbesserungen beim Umgang mit Dateisperren und Internationalisierung bei den Dateinamen. Eine ausführliche Liste der verfügbaren Optionen finden Sie in mount.cifs(8) und weitere Informationen in der Datei Documentation/filesystems/cifs.txt im Quellcode des Linux-Kernels.

Nur um es gesagt zu haben: Es ist eine ausgesprochen dumme Idee, das SMB-Kennwort auf der Kommandozeile von smbmount oder mount anzugeben, da andere Benutzer davon möglicherweise mit ps oder ähnlichem Kenntnis nehmen können. Sie tun besser daran, mit der Option »credentials=⟨*Dateiname*⟩« den Namen einer (angemessen lesegeschützten) Datei anzugeben, die Zeilen der Form

```
username=hugo
password=geheim
```

enthält.

Alternativ können Sie selbstverständlich auch von einem Windows-Rechner aus auf die Beispiel-Freigabe zugreifen. Dazu öffnen Sie Ihre »Netzwerkumgebung« und verwenden dort »Netzwerkressource hinzufügen«, um den Pfad zur Freigabe anzugeben – diesmal mit »\\«! Es spielt dabei keine Rolle, als welcher Benutzer Sie angemeldet sind, es handelt sich dank »`security = share`« und »`guest ok`« ja um eine Freigabe für anonyme Benutzer.

Wie Sie gesehen haben, ist es nicht wirklich schwierig, eine simple SMB-Freigabe mit Hilfe von Samba zu erstellen und darauf zuzugreifen. Das ist allerdings nur ein Bruchteil dessen, was mit Samba alles möglich (und oft auch nötig) ist. Im nächsten Kapitel lernen Sie zuerst die theoretischen Grundlagen für Samba kennen, in den darauf folgenden Kapiteln werden einzelne Aspekte, etwa spezielle Freigaben und Rechteverwaltung, insbesondere aber auch Benutzerverwaltung und Sicherheit besprochen.

Übungen

4.7 [!2] Legen Sie nach Angabe des Trainers eine Samba-Freigabe an und greifen Sie testweise darauf zu. Kopieren Sie sich eine Datei von der Freigabe auf Ihren lokalen Rechner. Sie können die Übung übrigens auch komplett auf dem lokalen Rechner durchführen.

4.8 [2] Hängen Sie Ihre Freigabe testhalber auch mit `smbmount` bzw. `mount` ein.

4.9 [2] Was passiert mit besonderen Dateitypen, etwa symbolischen Links oder Gerätedateien, die in dem Ihrer Freigabe zugrundeliegenden Verzeichnis auf dem Linux-Rechner stehen?

4.9 Testen und Überwachen von Samba

Wie bei jedem Netzwerkdienst ist es auch bei Samba wichtig, regelmäßig, insbesondere nach Neustarts zu überprüfen, ob die Daemons richtig laufen oder ob irgendwelche Fehler aufgetreten sind. Auch schadet es nichts, ab und an zu überprüfen, wer gerade auf den Server zugreift.

Speziell nach dem Erstellen oder dem Verändern einer Konfiguration sollten Sie sich von deren Korrektheit überzeugen. Noch vor dem Start der Daemons steht hierfür das Programm `testparm` zur Verfügung. Ein einfacher Aufruf genügt: `testparm`

```
# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[test]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
        workgroup = TEST
```

Syntaktische Fehler werden entsprechend angezeigt:

```
# testparm
Load smb config files from /etc/samba/smb.conf
Unknown parameter encountered: "workkroup"
Ignoring unknown parameter "workkroup"
Processing section "[test]"
```

Achtung: Fehler in den *Werten* der Variablen werden nicht unbedingt angezeigt. Geben Sie z. B. bei der Definition einer Freigabe einen nicht existierenden Pfad an, so ist das dem Programm `testparm` egal.

Interessant ist übrigens auch die Ausgabe von

```
# testparm -v
Load smb config files from /etc/samba/smb.conf
Processing section "[test]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
        dos charset = CP850
        unix charset = UTF-8
        display charset = LOCALE
        workgroup = TEST
◁◁◁◁◁
```

Mit der Option `-v` zeigt `testparm` alle Variable an, also auch die, deren Standardeinstellung gilt, was zumindest einen Eindruck davon verschafft, was mit Samba alles möglich ist. Wie auch immer Sie `testparm` aufrufen, Ihre Daemons sollten danach zumindest starten. Allerdings gibt ein erfolgreicher Ablauf von `testparm` noch keine Gewähr, dass alle Freigaben so funktionieren, wie Sie sich das vorgestellt haben.

Meldungen von `smbd`

Beim ersten Start von `smbd` kann es sinnvoll sein, die Meldungen auf der Standardausgabe auzugeben:

```
# smbd -iS
smbd version 3.5.4 started.
Copyright Andrew Tridgell and the Samba Team 1992-2010
Unknown parameter encountered: "workkroup"
Ignoring unknown parameter "workkroup"
```

Protokolldatei

Natürlich möchte der `smbd` seine Meldungen auch loswerden, wenn er als Daemon gestartet wird. In der Standardkonfiguration schreibt er dabei in die Protokolldatei `/var/log/samba/log.smbd`. Eine alternatives Verzeichnis für die Datei können Sie bei Aufruf des Daemons mit der Option `-l` angeben. Der Inhalt der Datei könnte beispielsweise so aussehen:

```
# cat /var/log/samba/log.smbd
[2005/01/07 20:01:35, 0] smbd/server.c:main(757)
  smbd version 3.5.4 started.
  Copyright Andrew Tridgell and the Samba Team 1992-2010
◁◁◁◁◁
[2005/02/03 16:46:32, 1] smbd/server.c:open_sockets_smbd(348)
  Reloading services after SIGHUP
[2005/02/03 16:46:32, 0] param/loadparm.c:map_parameter(2424)
  Unknown parameter encountered: "workkroup"
[2005/02/03 16:46:32, 0] param/loadparm.c:lp_do_parameter(3114)
  Ignoring unknown parameter "workkroup"
```

`syslogd`

Wie viele Dienste kann der `smbd` seine Meldungen auch an den `syslogd` weitergeben. Allerdings erst ab Samba 3 und auch nur dann, wenn er mit der `configure`-Option `--with-syslog` übersetzt wurde. In diesem Fall können Sie mit dem Parameter `syslog only = yes` in der Datei `smb.conf` festlegen, dass ausschließlich dieses Protokollziel verwendet werden soll.

Die Geschwätzigkeit, die smbd gegenüber seinen Protokollzielen an den Tag legt, können Sie beim Start mit der Option -d bestimmen: *Debuglevel*

```
# smbd -iS -d 2
smbd version 3.5.4 started.
Copyright Andrew Tridgell and the Samba Team 1992-2010
uid=0 gid=0 euid=0 egid=0
Unknown parameter encountered: "workkroup"
Ignoring unknown parameter "workkroup"
Processing section "[test]"
added interface ip=192.168.0.1 bcast=192.168.0.255 nmask=255.255.255.0
added interface ip=192.168.0.2 bcast=192.168.0.255 nmask=255.255.255.0
Registered MSG_REQ_POOL_USAGE
Registered MSG_REQ_DMALLOC_MARK and LOG_CHANGED
waiting for a connection
```

Sie haben dabei die Möglichkeit, *Debuglevel* zwischen 1 und 10 festzulegen. Sorgen Sie bei *Debuglevels* größer 3 aber dafür, dass Ihnen die Protokolldateien nicht die Systempartition zumüllen.

Im laufenden Betrieb haben Sie mehrere Möglichkeiten, die SMB-Verbindungen zu überwachen, die andere Rechner zu Ihrem Server aufgebaut haben. Eine erste Möglichkeit wäre netstat:

```
# netstat -tpn | grep smbd
tcp  0  0 192.168.0.100:445  192.168.0.1:33253  ESTABLISHED 7708/smbd
```

Etwas spezifischer und aussagekräftiger ist der Befehl smbstatus: smbstatus

```
# smbstatus
Samba version 3.5.4
◁◁◁◁◁
Service      pid     machine       Connected at
-------------------------------------------------------
test         7708    192.168.0.1   Thu Jan 19 17:05:04 2005
◁◁◁◁◁
```

Er zeigt an, welcher Rechner auf welche Freigabe zugreift und welche Prozess-ID die für die Verbindung zuständige smbd-Instanz hat. Letzteres ermöglicht einen Abbruch der Verbindung:

```
# kill 7708
```

Je nachdem, welche Geschmacksrichtung von SMB-Client an der beendeten Verbindung beteiligt war, hat ein Beenden der smbd-Instanz verschiedene Konsequenzen; Einen aktuellen Windows-Client (ab Windows 2000) lässt das Ganze ziemlich kalt – er erstellt eine neue Verbindung, ohne das der Benutzer etwas merkt. Einfachere Client-Implementierungen wie smbclient beenden sich nach einer gewissen Zeit.

Ab Samba 3 steht Ihnen alternativ zu smbstatus auch das Programm net zur Verfügung. Mit den Parametern status shares zeigt er alle offenen SMB-Verbindungen an: net

```
# net status shares
Service      pid     machine       Connected at
-------------------------------------------------------
test         7795    192.168.0.1   Thu Jan 19 17:15:23 2005
```

Zu guter Letzt gibt es noch die Möglichkeit, sich mit swat die bestehenden Verbindungen anzeigen zu lassen. Auch ein Beenden der Verbindung ist möglich.

Übungen

4.10 [!2] Greifen Sie mit einer SMB-Client-Implementierung auf eine Ihrer Testfreigaben zu und lassen Sie sich die Verbindung anzeigen. Versuchen Sie die Verbindung zu beenden.

Kommandos in diesem Kapitel

nmbd NetBIOS-Nameserver (unter anderem) nmbd(8) 59
rpcclient Werkzeug zum Ausführen von MS-RPC-Funktionen rpcclient(1) 56
smbd Server für SMB/CIFS-Dienste smbd(8) 59
smbmount Programm zum Einhängen von Windows- oder Samba-Freigaben als Dateisysteme smbmount(8) 56, 66
smbstatus Zeigt die aktiven Verbindungen auf einem Samba-Server an smbstatus(1) 69
smbtree Ein textbasierter SMB-Umgebungs-Browser (à la Windows-„Netzwerkumgebung“) smbtree(1) 56
testparm Prüft eine Samba-Konfiguration und gibt sie aus testparm(1) 67

Zusammenfassung

- Samba ist ein frei verfügbarer Server für Dateizugriffs- und Druckdienste nach dem SMB-Protokoll.
- Samba besteht aus den Serverprogrammen `smbd`, `nmbd` und `winbindd` sowie diversen Hilfsprogrammen.
- Samba ist leicht entweder über Linux-Distributionspakete oder den Quellcode zu installieren.
- Samba wird normalerweise über Init-Skripte gestartet (die von Distribution zu Distribution verschieden aussehen können).
- Zur Konfiguration der Samba-Programme dient die Datei `smb.conf`.
- `testparm`, `smbstatus` und `net` erlauben die Kontrolle einer Samba-Konfiguration bzw. eines laufenden Samba-Servers.

Literaturverzeichnis

Her03 Christopher R. Hertel. *Implementing CIFS—The Common Internet FileSystem*. Bruce Perens' Open Source Series. Prentice-Hall, 2003. ISBN 0-13-047116-X. http://ubiqx.org/cifs/

Kü04 Jens Kühnel. *Samba 3 – Wanderer zwischen den Welten*. Bonn: mitp-Verlag, 2004. ISBN 3-8266-0985-9.

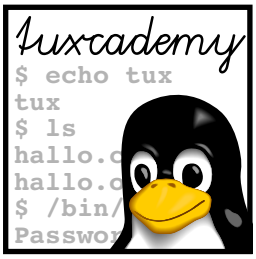
TECB03a Jay Ts, Robert Eckstein, David Collier-Brown. *Samba*. Sebastopol, CA: O'Reilly & Associates, 2003, 2. Auflage. ISBN 3-89721-359-1. http://www.oreilly.de/catalog/samba2ger/

TECB03b Jay Ts, Robert Eckstein, David Collier-Brown. *Using Samba*. Sebastopol, CA: O'Reilly & Associates, 2003, 2. Auflage. ISBN 0-596-00256-4. http://www.oreilly.com/catalog/samba2/

Ter04 John H. Terpstra. *Samba-3 by Example. Practical Exercises to Successful Deployment*. Prentice Hall PTR, 2004. ISBN 0-13-147221-6.

TV03 John H. Terpstra, Jelmer R. Vernooij. *The Official Samba-3 HOWTO and Reference Guide*. Prentice Hall PTR, 2003. ISBN 0-13-145355-6.

TV05 John H. Terpstra, Jelmer R. Vernooij. *Samba 3 – das offizielle Handbuch*. München: Addison-Wesley, 2005. ISBN 3-82732152-2.



5

Die Theorie – Protokolle und Domänen

Inhalt

Lernziele

- Die Protokolle hinter den Kulissen von Samba kennenlernen
- Verstehen des Windows-Arbeitsgruppen- und Domänen-Konzepts
- Einordnen der Begriffe ADS und LDAP
- Kennenlernen der verschiedenen Namensauflösungsmethoden in Windows-Netzwerken

Vorkenntnisse

- Kenntnisse über das TCP/IP-Schichtenmodell und Netzwerkprotokolle
- Grundlegendes Wissen über Namensauflösung und DNS

5.1 Überblick

Das folgende Kapitel ist stark theorielastig und mag dem ein oder anderen Leser zu detailliert erscheinen. Für die bloße Verwaltung eines Samba-Servers könnte das zutreffen – allerdings nur bis etwas schief geht. Vor allem bei Kommunikationsproblemen zwischen Windows-Clients und Samba-Server kommen Sie auch mit ausführlicher Dokumentation nicht weiter, wenn Sie mit Begriffen wie »SMB«, »NetBIOS« oder »Direct-Hosted TCP« nichts anfangen können. Auch aus Gründen der Sicherheit ist es wichtig, sich nicht nur mit der Terminologie, sondern auch mit den genauen Abläufen der Netzwerkkommunikation zu beschäftigen – insbesondere, wenn es um Authentisierung und Verschlüsselung geht.

Im folgenden Kapitel werden verschiedene Eigenschaften von SMB vorgestellt und zum Teil auch detailliert besprochen. Das hier Erlernte wird dann in den späteren Kapiteln mit Samba in die Praxis umgesetzt.

Die meisten Informationen über die beschriebenen Protokolle wurden übrigens vom Samba-Team durch Beobachten des Netzwerk-Verkehrs erschlossen und nicht etwa der offiziellen SMB-Dokumentation entnommen. Es ist wahrscheinlich, dass es nicht einmal bei Microsoft selbst vernünftige Dokumentation gibt: Microsoft wurde 2005 nachdrücklich von der Europäischen Kommission aufgefordert, eine solche zur Verfügung zu stellen, und lieferte tatsächlich etliche tausend Seiten aus, die von dem (von Microsoft selbst bestellten) externen Experten aber nicht als nützlich angesehen wurden. SMB ist sicherlich kompliziert und unübersichtlich genug, dass diese Geschichte zumindest plausibel scheint; über die Jahre hat jede Windows-Version ihre eigenen Fehler und Eigenheiten beigesteuert, die von den neueren Versionen dann aus Kompatibilitätsgründen mit unterstützt werden mussten.

Über diese Unterlage hinausgehende Informationen zu den einzelnen Protokollen und zu Samba selbst, finden Sie in erster Linie unter `http://www.samba.org`. Die erwähnten RFCs finden Sie unter `http://www.ietf.org/rfc/`.

5.2 Die Protokolle – SMB, NetBEUI & Co.

5.2.1 Was ist SMB?

SMB (*Server Message Block*) ist ein von allen Windows-Systemen unterstütztes Netzwerk-Protokoll. Es ermöglicht mehreren Rechnern die gemeinsame Nutzung von Dateien bzw. Verzeichnissen und Druckern. Gemeinsam genutzte Ressourcen heißen in diesem Zusammenhang auch »**Freigaben**« (*shares*).

Freigaben

Neben der gemeinsamen Nutzung von Verzeichnissen und Druckern ermöglicht SMB auch die von seriellen Schnittstellen sowie von abstrakten Kommunikationswegen wie *named pipes*. Unter Windows werden im Gegensatz zu Unix *named pipes* nicht nur lokal, sondern mit Hilfe von SMB auch zur Kommunikation von Programmen auf verschiedenen Rechnern verwendet. Diese *named pipes* dienen zur Realisierung der *Microsoft Remote Procedure Calls* (MS-RPCs).

Seit 1983

SMB wurde ursprünglich 1983 von Barry A. Feigenbaum bei IBM entwickelt und hieß deswegen zuerst auch »BAF«. Später wurde es bei Microsoft in »SMB« umbenannt. Mit dem Protokoll sollten DOS-Rechnern miteinander kommunizieren können.

Schichtenmodell

Bei SMB handelt es sich um ein klassisches Client-Server-Protokoll, das man im TCP/IP-Schichtenmodell in der Applikationschicht ansiedeln könnte. Allerdings können Sie SMB nicht nur mit IP als Transportdienst, sondern auch auf der Basis von NetBEUI oder IPX (Novell) verwenden. Nach dem Aufbau einer Verbindung (z. B. über TCP), können die beteiligten Rechner sich SMB-Kommunikationspakete schicken - eben die besagten *Server Message Blocks*.

Tabelle 5.1: Portzuordnung von NetBIOS-Diensten bei NetBT

Dienst	Name	Port
NetBIOS-Name-Dienst	`netbios-ns`	`137/udp`
NetBIOS-Datagramm-Dienst	`netbios-dgm`	`138/udp`
NETBIOS-Sitzungs-Dienst	`netbios-ssn`	`139/tcp`

Aufbau des Protokollheaders

Ein »*Server Message Block*« besteht aus einem Header fixer Größe und einem Datenteil variabler Größe. Der entscheidende Teil des SMB-Headers ist das 8 Bit breite *Kommando*. Momentan sind 66 verschiedene Kommandos bekannt und in Samba implementiert. Typische Kommandos sind beispielsweise `open` (Nummer 2) und `mkdir` (Nummer 0). Der 32 Bit breite *Status*-Bereich dient zur Übermittlung von Fehlermeldungen.

Identifikationsnummer

Die Felder `TID`, `PID`, `UID` und `MID` enthalten eindeutige Identifikationsnummern für die Freigabe (*Tree ID*), den Prozess auf dem Client (*Process ID*), den jeweiligen Benutzer (*User ID*) bzw. die SMB-Verbindung selbst (*Message ID*). Die Identifikationsnummern sind u. a. notwendig für die Rechteverwaltung und das Sperren (*locking*) von Dateien.

5.2.2 Welche Rolle spielt NetBIOS?

NetBIOS

NetBIOS (engl. *Network Basic Input/Output System*) wurde Anfang der 1980er Jahre von IBM und Sytec entwickelt, damit PC-Programme mit Netzwerkhardware kommunizieren konnten (so wie sie seinerzeit das BIOS verwendeten, um mit Tastatur, Bildschirm, Platten usw. zu kommunizieren). NetBIOS war Teil einer simplen PC-Netzwerk-Infrastruktur – *kein* Protokoll! Ein wichtiger Bestandteil von NetBIOS ist das Adressierungsschema, das 16 Byte-Namen für Rechner und netzwerkfähige Programme vorsieht – die NetBIOS-Namen.

NetBIOS-Namen

NetBIOS-Namenstypen

NetBIOS kennt verschiedene NetBIOS-Namenstypen. Dabei handelt es sich im Prinzip um verschiedene NetBIOS-Dienste. Einer davon ist der oben beschriebene NetBIOS-Name-Service (`netbios-ns`). Ein weiterer, der NetBIOS-Datagramm-Service (`netbios-dgm`) ist für die Übertragung der Daten zuständig. Der NetBIOS-Session-Service (`netbios-ssn`) schließlich dient zum Aufbau und Abbau von Verbindungen.

NetBIOS und SMB

Wo ist der Zusammenhang mit SMB? Microsoft integrierte NetBIOS-Routinen in DOS und ermöglichte es DOS, Dateisysteminhalte an die NetBIOS-Schnittstelle und damit an die Netzwerkhardware weiterzugeben. Alles was für gemeinsame Festplattenressourcen im Netz noch fehlte, war ein Netzwerkprotokoll zur Übertragung der Daten und Adressen – SMB.

Im Prinzip funktioniert SMB allein mit NetBIOS, was allerdings nur primitive Netzwerke erlaubt und mit den sich damals etablierenden Netzwerk-Standards (Ethernet, Token-Ring, TCP/IP) nicht viel gemein hatte. Deshalb entwickelte IBM NetBEUI (*NetBIOS Enhanced User Interface*), ein Protokoll, um NetBIOS auf Ethernet und Token-Ring zu verwenden.

NetBEUI

Ein NetBEUI-basiertes Netz kann maximal 256 Knoten in einem einzigen physikalischen Netz enthalten. TCP/IP-Eigenschaften wie Subnetting und Routing werden nicht unterstützt. Bei NetBEUI handelt sich also um ein reines Protokoll für lokale Netze.

Der nächste logische Schritt nach oben (im Schichtenmodell) war das Verwerfen von NetBEUI und der Transport von NetBIOS über die etablierten Protokolle für Adressierung, Wegleitung und Verbindungsaufbau. Ein Ansatz war NBIPX (*NetBIOS over IPX*). IPX ist eine von Novell entwickelte Alternative zu TCP/IP, die heute in der Praxis kaum noch vorkommt.

NBIPX

Der zweite, heute noch häufig verwendete Ansatz, war NetBT (*NetBIOS over TCP/IP*), auch NBT oder RFCNB genannt. Hierbei handelt es sich um eine in [RFC1001] und [RFC1002] beschriebene Methode zur NetBIOS-Nutzung in

NetBT

TCP/IP-Netzen. NetBIOS-Namen werden dabei zu IP-Adressen umgesetzt, und den einzelnen NetBIOS-Namenstypen sind bestimmte Ports zugeordnet (Tabelle 5.1). Die Namensauflösung erfolgt über WINS, NetBIOS-Broadcasts oder DNS (siehe auch Abschnitt 6.1).

Direct Hosted TCP

Der nächste (letzte?) Schritt war *Direct Hosted TCP*. Hierbei handelt es sich im Wesentlichen um eine Formulierung der Microsoft-Marketing-Abteilung, um die vollständige Eliminierung von NetBIOS zu beschreiben. SMB wird direkt über Port `445/tcp` realisiert. Die Namensauflösung erfolgt hier per Standard über DNS.

Ältere Windows-Systeme wie Windows NT verwenden genau wie Samba 2.2 NetBT. Moderne Windows-Versionen wie Windows 2000, Windows XP oder Windows 2003 Server erlauben wie auch Samba 3 die Wahl zwischen NetBT und *Direct Hosted TCP*. NetBIOS kann unter Samba 3 mit dem Eintrag »`disable netbios = yes`« im globalen Abschnitt der Datei `smb.conf` abgeschaltet werden. Das sollten Sie im Regelfall allerdings nicht tun – Sie könnten sich ansonsten Kompatibilitätsprobleme mit älteren Systemen einhandeln.

5.2.3 Versionen und Fähigkeiten von SMB

Seit seiner Entwicklung Anfang der 1980er Jahre wurden SMB-Implementierungen verschiedenster Ausprägung entwickelt, um der immer weiter fortschreitenden Komplexität in Netzen Herr zu werden. Welche SMB-Variante bei einer Verbindung verwendet wird, handeln Client und Server mit ihren ersten Paketen aus.

Core Protocol

Die ursprüngliche SMB-Variante war das »*Core Protocol*« (z. B. im »PC NETWORK PROGRAM 1.0«). Es beherrscht die wesentlichen Vorgänge, um den Zugriff auf Verzeichnis- und Drucker-Freigaben zu ermöglichen:

Fähigkeiten

- Aufbau und Abbau von Verbindungen zu Freigaben
- Öffnen und Schließen von Dateien
- Lesen und Schreiben von Dateien
- Erstellen und Löschen von Dateien
- Anzeige von Verzeichnis-Inhalten
- Lesen und Schreiben von Datei-Attributen
- Sperren (engl. *locking*) von Dateien

Die folgenden Protokoll-Varianten, nämlich das »*Core Plus Protocol*« sowie die verschiedenen Versionen des Windows-internen »*LAN Manager*s« erfuhren im Wesentlichen Verbesserungen dieser Grundfunktionalität. Dabei wurden zum Teil neue Typen von SMBs eingeführt oder einfach bestehende abgeändert. Zu den wichtigen Neuerungen späterer Versionen gehörten der »Suchdienst« (*Browsing*, siehe Abschnitt 6.6) und die »*Windows NT Domain Control*«, eine Methode zur Authentisierung von Benutzern (siehe Abschnitt 5.5).

5.2.4 Ablauf einer SMB-Sitzung

Im Folgenden ein kurzer Abriss der Abläufe einer SMB-Sitzung bei der Verwendung von NetBT:

1. Zum Verbindungsaufbau findet ein herkömmlicher TCP-3-Wege-Handschlag auf dem TCP-Port 139 (im Falle von Direct Hosted TCP dem Port 445) statt. Aktuelle Client-Implementierungen versuchen an beiden Ports einen Verbindungsaufbau. Beide Rechner sind jetzt bereit, Informationen mittels der SMBs auszutauschen.

2. Bei der *SMB-Negotiate-Protocol*-Aushandlung entscheiden die beteiligten Rechner über die zu verwendende SMB-Variante und damit auch über das Authentisierungs-Verfahren (siehe Abschnitt 5.5).

3. Bei der *NetBIOS-Session*-Anfrage werden die NetBIOS-Namen der beteiligten Rechner übertragen (dieser Schritt entfällt im Fall von Direct Hosted TCP).

4. Mittels *Session-Setup*-SMBs kann sich der Client, wenn erwünscht, authentisieren. Im Falle benutzerorientierter Authentisierung tragen alle folgenden Pakete eine eindeutige Benutzernummer (UID) zur Identifizierung. Diese UID ist temporärer Natur und sollte nicht mit Unix-UIDs verwechselt werden.

5. Beim *SMB Tree Connect* wird der Name der gewünschten Freigabe übermittelt und eine Verbindung hergestellt – sofern der Benutzer entsprechende Rechte hat. Zum Kennzeichnen der Verbindung erstellt der Server eine eindeutige TID.

6. Während der Verbindung werden SMBs zum Erstellen von Verzeichnissen, zum Öffnen von Dateien, usw. übertragen. SMB unterscheidet dabei *keine* Groß- und Kleinschreibung.

Groß- und Kleinschreibung

7. Über zu *Session Setup* und *SMB Tree Connect* korrespondierende Vorgänge kann eine bestehende SMB-Verbindung beendet werden.

5.2.5 Und was ist CIFS?

CIFS (engl. *Common Internet File System*) ist im Grunde dasselbe wie SMB – 1996 entschied Microsoft, dass SMB das Wort »Internet« im Namen bräuchte, und änderte den Namen in CIFS.

Sie sollten sich von Microsofts Marketing nicht irreführen lassen – einen Samba-Server sollten Sie (genausowenig wie einen Windows-Server) keinesfalls direkt dem Internet aussetzen! Wenn Sie wirklich Samba-Freigaben über das Internet zur Verfügung stellen müssen, dann verwenden Sie unbedingt ein VPN mit starker Verschlüsselung, etwa auf der Basis von OpenVPN (mehr hierzu in der Linup-Front-Schulungsunterlage *Linux-Sicherheit*).

Übungen

5.1 [5] Verfolgen Sie mit Hilfe eines Netzwerksniffers, z. B. mit `ethereal`, was beim Zugriff auf eine SMB-Freigabe mit `smbclient` für Pakete entstehen. Vergleichen Sie das Ganze, wenn möglich, mit dem Netzwerkverkehr unter Verwendung eines Windows-Clients.

5.3 Arbeitsgruppen und Domänen

Arbeitsgruppen und Domänen sind logische Konstrukte unter Windows bzw. NetBIOS, um mehrere Rechner zu Gruppen zusammenzufassen (insbesondere Gruppen mit einem gemeinsamen Suchdienst – Abschnitt 6.6). In einer Arbeitsgruppe hält jeder Rechner seine eigenen Authentisierungsinformationen vor. Bei Windows 95/98/ME kann allerdings auch ein zentraler Kennwort-Server existieren, der von den anderen Rechnern verwendet wird.

Arbeitsgruppe

Eine Erweiterung des Arbeitsgruppenkonzepts stellen NT-Domänen dar. In einer **NT-Domäne** werden die Authentisierungsinformationen immer zentral verwaltet. Jede Domäne hat einen oder mehrere Rechner, die als Domänencontroller (DC) fungieren. Üblicherweise gibt es einen *Primary Domain Controller* (PDC) und – in größeren Netzen – einen oder mehrere *Backup Domain Controller* (BDC). Auf den DC liegen sämtliche für die Authentisierung von Benutzern notwendigen Informationen, also die Benutzernamen, Kennwörter, Gruppenzugehörigkeiten und Ähnliches.

NT-Domäne
Domänencontroller

Im Prinzip ist eine Windows-NT-Domäne nichts anderes als eine Arbeitsgruppe, in der die beteiligten Rechner die Authentisierung von Benutzern an den Domänencontroller delegieren. Während im ursprünglichen Arbeitsgruppenkonzept jeder Rechner die Zugriffe auf seine Ressourcen (Freigaben) selber authentisiert, übernimmt das in einer Domäne zentral der Domänencontroller. Ein Domänencontroller ist in der Lage, quasi »Eintrittskarten« zu verteilen, die Clients ohne erneute Authentisierung Zugriff auf die Ressourcen anderer Server in der Domäne geben; in einer reinen Arbeitsgruppe wäre dafür eine separate Authentisierung erforderlich.

Auch hier gilt: Windows-Domänen haben nichts mit dem DNS (Domain Name System) zu tun. Sie heißen nur zufällig so ähnlich. Wir verwenden daher den Begriff »Domäne« nur im Zusammenhang mit Windows-Domänen und nicht, wenn wir über DNS sprechen.

In einem Windows-Netz können Samba-Server als PDC oder BDC auftreten. Die Replikation der Daten zwischen Samba-basierten PDC und BDC funktioniert allerdings nur mit Hilfe von LDAP vernünftig.

5.4 ADS und LDAP

In großen Netzen mit vielen Benutzern ist es problematisch, alle Benutzer mit den herkömmlichen »flachen« Authentisierungsverfahren zu verwalten. Was es für die Namensauflösung schon lange in Form von DNS gibt, steht mit LDAP jetzt unter anderem auch für die Benutzerverwaltung zur Verfügung. Das **LDAP** (engl. *Lightweight Directory Access Protocol*) ermöglicht es, Objekte (z. B. Benutzer) in einer baumförmigen Struktur, dem »Verzeichnis« (engl. *Directory*), zu speichern und netzweit abzurufen. Das »Verzeichnis« wiederum kann sich verteilt – in Stücken oder redundant – auf verschiedenen Rechnern befinden.

LDAP

Mit Windows 2000 hat Microsoft einen eigenen Verzeichnisdienst eingeführt, das **Active Directory** (AD), auch *Active Directory Service* (ADS) genannt. ADS besteht aus LDAP, dem verteilten Authentisierungsverfahren Kerberos, DDNS und weiteren bekannten Verfahren, allerdings nicht in der standardisierten, sondern in einer von Microsoft (zwecks Monopolbildung) veränderten Form. ADS wird von Windows-Maschinen unter anderem zur Authentisierung der Benutzer verwendet. Im Moment können Sie einen Unix-Rechner mit Samba als Mitglied in ein AD eingliedern, aber noch keinen Active-Directory-Server aufsetzen – das wird erst mit Samba 4 möglich sein.

Active Directory

5.5 Sicherheitsstufen, Zugriffsbeschränkung und ACLs

Üblicherweise ist es nicht erwünscht, dass jeder Benutzer im Netz auf beliebige Freigaben zugreifen darf. Das SMB-Protokoll ermöglicht Zugriffsbeschränkungen mittels verschiedener Methoden. Diese Methoden werden in Form sogenannter Sicherheitsstufen (engl. *security levels*) definiert:

Sicherheitsstufen

Share Level Zusammen mit dem »User level« Bestandteil der ersten SMB-Implementierungen. Mögliche Zugriffsbeschränkungen sind rein an den Freigaben orientiert – eine Unterscheidung zwischen verschiedenen Benutzern findet (im Gegensatz zu allen anderen Levels) nicht statt. Jeder Freigabe kann ein eigenes Kennwort zugewiesen werden (das dann aber alle Benutzer kennen müssen, die auf die Freigabe zugreifen dürfen sollen).

Den Share Level sollten Sie nicht wirklich verwenden – bekanntlich ist ein Geheimnis, das mehr als eine Person kennt, keins mehr. Im übrigen gibt es spezielle Programme, die in der Lage sind, die Share-Level-Freigaben in einem Netz zu finden und die Kennwörter dafür zu bestimmen.

User Level Die Zugriffsbeschränkungen sind datei- und benutzerspezifisch – jeder Benutzer, der auf eine Datei zugreifen will, muss sich authentisieren und entsprechende Zugriffsrechte für die betreffende Freigabe überhaupt sowie die Datei selbst besitzen. Die notwendigen Authentisierungsinformationen liegen dabei lokal auf dem Rechner, auf dem auch die Freigabe liegt.

Server Level Entspricht im Wesentlichen dem »User Level«. Die Authentisierungsinformationen liegen hier auf einem anderen Rechner, dem Kennwort-Server der Arbeitsgruppe.

Kennwort-Server

Domain Level Entspricht im Wesentlichen dem »Server Level«, aber setzt statt einer Arbeitsgruppe eine NT-Domäne voraus. Die Authentisierungsinformationen liegen auf einem Domänencontroller. Zu diesem Rechner wird eine gesicherte Verbindung aufgebaut, um die Authentisierungsinformationen zu übertragen. Die Benutzer authentisieren sich nicht gegenüber einem Einzelrechner, sondern nur einmal gegenüber der kompletten Domäne, und bekommen so Zugriff auf alle Server in der Domäne.

ADS Entspricht im Wesentlichen dem »Domain Level«. Die Authentisierung erfolgt hier über ADS und Kerberos (siehe Abschnitt 5.4)

Entscheidend ist, dass jeder Server mit einer Freigabe erneut eine Authentisierung verlangt, es also statt wie bei NFS nicht ausreicht, wenn ein Benutzer sich bereits auf dem Client erfolgreich authentisiert hat. Die Authentisierung kann allerdings, wie beschrieben, erheblich vereinfacht werden, wenn die Benutzer auf dem Client sich nicht lokal, sondern gegenüber einer NT-Domäne authentisieren.

Welche Authentisierungsmethode verwendet wird, hängt von dem Ergebnis der *SMB-Negotiate-Protocol*-Aushandlung ab. Die eigentliche Authentisierung erfolgt während des *Session Setup*. Dabei werden der Name des Benutzers, sein Kennwort und seine Domäne übermittelt. Bei der Kennwortübermittlung kommen folgende Methoden in Frage:

`Null, Klartext` Übermittlung ohne Verschlüsselung (hu brr)

`LM, NTLM, NTLMv2` (*LAN Manager*) Challenge-Response-Verfahren

`NTLMSSP` (*NT LAN Manager Security Service Provider*) Aktuelle Version, verwendet ebenfalls das Challenge-Response-Verfahren

`Kerberos` Zentralisiertes Authentisierungsverfahren, funktioniert nur in Zusammenhang mit ADS

Seit Windows 98 bzw. Windows NT SP3 werden per Standard im *User Level* verschlüsselte Kennwörter übertragen, auch bei Samba 3 ist das der Standard. Klartext-Kennwörter sollten Sie auf keinen Fall verwendet. Client-Systeme, die keine Kennwortverschlüsselung kennen, würden aus Kompatibilitätsgründen ein Abschalten der Kennwortverschlüsselung bei allen anderen Systemen erfordern – und gehören damit auf den Betriebssystem-Schrotthaufen.

Eigentlich ist es falsch, bei dem erwähnten Challenge-Response-Verfahren von »Kennwortverschlüsselung« zu reden. Das Verfahren läuft wie folgt ab:

1. Beim Erstellen eines Benutzerkontos auf dem Server wird das Kennwort im Klartext eingegeben. Allerdings wird es nicht so, sondern in Form eines Hash-Wertes abgespeichert. Dabei wird immer derselbe Algorithmus verwendet – ein und das selbe Kennwort ergibt immer denselben Hash-Wert. (Genaugenommen handelt es sich bei den Windows 95/98/ME und Windows NT/2000/XP um zwei verschiedene Algorithmen, namentlich respektive DES und MD4.)

Hash-Wert

2. Möchte sich zu einem späteren Zeitpunkt ein Benutzer auf einem Client gegenüber dem Server authentisieren, muss er das entsprechende Kennwort im Klartext eingeben. Auf dem Client wird temporär der Hash-Wert erzeugt.

3. Für eine erfolgreiche Authentisierung des Benutzers muss der gerade gebildete Hash-Wert auf dem Client mit dem gespeicherten Hash-Wert auf dem Server übereinstimmen. Um diesen Vergleich durchzuführen, können aber keine der Hash-Werte über das Netz geschickt werden, wo ein Angreifer sie mithören könnte. Statt dessen erzeugt der Server eine Zufallszahl und schickt eine Kopie davon an den Client (»Parole?« – *challenge*). Der Client verschlüsselt diese Zufallszahl mit dem vom Benutzer eingegebenen Hash-Wert (unter anderem) als Schlüssel und schickt das Ergebnis zurück an den Server (*response*). Der Server versucht dies dann mit *seiner* gespeicherten Kopie des Benutzer-Hash-Werts zu entschlüsseln; wenn er die ursprüngliche Zufallszahl als Ergebnis erhält, wird angenommen, dass der Benutzer auf dem Client das richtige Kennwort gewusst hat.

challenge

response

Angreifer

Ein Angreifer, der im Netzwerk mitlauscht, sieht nur den Austausch von Zufallszahlen, mit denen er nichts anfangen kann (denn beim nächsten Anmeldevorgang wird eine andere Zufallszahl gewählt, die dann auch eine andere Antwort bedingt). Selbst ein Administrator, der die Hash-Werte sehen kann, kann nicht ohne weiteres das Kennwort eines Benutzers erraten, da man von einem Hash-Wert nicht auf den ursprünglichen Wert zurückschließen kann.

Das beschriebene Challenge-Response-Verfahren ist keineswegs SMB-spezifisch, sondern wird in der EDV-Welt recht häufig für die Authentisierung eingesetzt. Beispiele dafür finden sich in APOP, der Secure Shell, SMTP-AUTH und vielen anderen Protokollen.

Zugriffsrechte

Hat ein Benutzer sich für eine Freigabe erfolgreich authentisiert, entscheiden die Zugriffsrechte für die einzelnen Dateien, was der entsprechende Benutzer darf. Allerdings unterscheiden sich die klassischen Unix-Zugriffsrechte ganz erheblich von den Rechten, die man von Windows aus setzen kann. Unter Samba findet hier eine Abbildung der Windows-Rechte auf die Unix-Rechte statt. Mehr dazu erfahren Sie in Abschnitt 7.7.

ACLs

Sowohl bei Unix[1] als auch bei den Windows-Varianten gibt es ACLs (*Access Control Lists*), die es ermöglichen, über die Standardrechte hinaus festzulegen, welcher Benutzer was mit welcher Datei tun darf. Leider unterscheiden sich die Implementierungen ein wenig, auch innerhalb von Windows. Trotzdem ist es möglich, die ACLs von Windows auch unter Samba zu verwenden, wenn hinter dem Samba-Server ein Unix- oder Linux-System steht, das ACLs unterstützt. Mehr hierzu in Abschnitt 7.8.

Zusammenfassung

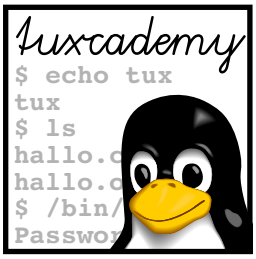
- SMB (*Server Message Block*) ist ein von allen Windows-Systemen unterstütztes Netzwerk-Protokoll. Es ermöglicht mehreren Rechnern die gemeinsame Nutzung von Dateien bzw. Verzeichnissen und Druckern.
- NetBIOS ist ein (einigermaßen) medienneutrales Netzwerkzugangsverfahren. Heutzutage wird es fast ausschließlich auf TCP/IP eingesetzt.
- Arbeitsgruppen und Domänen dienen dazu, mehrere Rechner zusammenzufassen. Domänen sind dabei spezielle Arbeitsgruppen, deren Server die Authentisierung an einen Domänencontroller delegieren.
- Active Directory und LDAP dienen zur strukturierten Speicherung von Verwaltungsdaten in großen Netzen.
- Es gibt diverse Authentisierungsmethoden für den Zugriff auf gemeinsame Ressourcen.

[1] Jedenfalls Linux, bei den meisten gängigen Dateisystemen.

Literaturverzeichnis

RFC1001 Network Working Group. »Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods«, Mârz 1987.
http://www.ietf.org/rfc/rfc1001.txt

RFC1002 Network Working Group. »Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications«, Mârz 1987.
http://www.ietf.org/rfc/rfc1002.txt



6

NetBIOS-Namensauflösung und Browsing

Inhalt

Lernziele

- Die verschiedenen Methoden der NetBIOS-Namensauflösung verstehen
- `nmbd` für die NetBIOS-Namensauflösung verwenden können
- Browsing (Suchdienst) verstehen
- Einen eigenen WINS-Server konfigurieren können

Vorkenntnisse

- Grundlegendes Wissen über Namensauflösung in Netzwerken
- Allgemeine Kenntnisse über die Konfiguration von Samba

6.1 Namen und Namensauflösung à la Windows

NetBIOS-Namen

In NetBIOS werden Rechner traditionell nicht über Adressen identifiziert, sondern über ihre Namen. **NetBIOS-Namen** sind 16 Byte lang. Die ersten fünfzehn Zeichen enthalten den eigentlichen Rechnernamen, dabei wird Groß- und Kleinschreibung nicht unterschieden. Erlaubt sind alphanumerische Zeichen und diverse Sonderzeichen; um Leerzeichen und Umlaute sollten Sie allerdings einen Bogen machen. Ist der eigentliche Name kürzer als 15 Zeichen, wird er NetBIOS-intern mit Leerzeichen aufgefüllt. Das sechzehnte Zeichen des NetBIOS-Namens kennzeichnet die Rolle eines Rechners, man nennt es auch »NetBIOS-Suffix« (Tabelle 6.1 zeigt eine Auswahl). Eine gängige Schreibweise setzt das sechzehnte Zeichen als Hexadezimalwert in spitzen Klammern ans Ende des Namens; (Folgen von) Leerzeichen unmittelbar vor dem sechzehnten Zeichen werden weggelassen.

NetBIOS-Suffix
Schreibweise

NetBIOS-Namen werden außerdem dazu herangezogen, alle Rechner mit einer bestimmten Eigenschaft zu finden, zum Beispiel alle Domänencontroller. Hierzu sucht ein Windows-Rechner nach allen Rechnern, die den Namen »`DOMAENE`⟨*1c*⟩« registriert haben. Eine Auswahl dieser speziellen Namen finden Sie in Tabelle 6.2.

Windows sieht verschiedene Formen der Namensauflösung vor:

Broadcast Name Resolution Die Rechner reservieren sich selbst einen Namen, testen aber vorher per Broadcast, ob ein Rechner gleichen Namens im Netzwerk schon existiert. Die Namensvergabe ist komplett dynamisch; Jeder Rechner kann ohne weitere Kontrolle einen Namen übernehmen. Namenskonflikte sind nicht ausgeschlossen (bis häufig).

Die Namensauflösung kann in NetBEUI-Netzen ebenfalls über Broadcast geregelt werden; ein Rechner *X*, der auf eine Freigabe auf Rechner *Y* zugreifen möchte, sucht *Y* über einen Broadcast. In NetBT-Netzen müssen NetBIOS-Namen zuerst in IP-Adressen aufgelöst werden, was analog funktioniert: Möchte ein Rechner mit einem anderen Rechner bestimmten Namens kommunizieren, fragt er per Broadcast, wer so heißt. Fühlt sich ein Rechner angesprochen, sendet er entsprechend eine Antwort, die seine IP-Adresse enthält. Diese Broadcasts sorgen bei größeren Netzen für erheblichen Netzwerkverkehr.

Alternativ können NetBIOS-Namen und die dazugehörigen IP-Adressen in der Datei `lmhosts` explizit vorgegeben werden.

WINS (*Windows Internetworking Name Server*) Hieß zuerst *NetBIOS Name Server*. Hierbei handelt es sich nicht um ein Broadcast-, sondern um ein Punkt-zu-Punkt-System. Ein bootender Rechner registriert seinen Namen und seine IP-Adresse bei einem zentralen WINS-Server, der die Informationen speichert. Der WINS-Server hat dabei ein Vetorecht. Möchte ein Rechner mit einem anderen Rechner bestimmten Namens kommunizieren, fragt er den WINS-Server.

WINS-Server

DNS Bei *Direct Hosted TCP* werden keine NetBIOS-Namen mehr verwendet – die Namensauflösung läuft ausschließlich über DNS. Wichtig für die Funktion der Windows-Rechner: Die »Netzwerkumgebung« im »Arbeitsplatz« zeigt

Tabelle 6.1: NetBIOS-Suffixe (Auswahl)

Name	Bedeutung
`RECHNER`⟨*00*⟩	`RECHNER` bietet Serverdienste an
`RECHNER`⟨*03*⟩	Generischer (NetBIOS-)Rechnername
`RECHNER`⟨*20*⟩	`RECHNER` bietet LANManager-Serverdienste (Freigaben) an
`GRUPPE`⟨*1b*⟩	Rechner ist Domain Master Browser

ohne NetBIOS keine Einträge mehr! Das Abschalten von NetBIOS empfiehlt sich daher nur in großen Netzen, wo die NetBIOS-Broadcasts zu viel Netzwerkverkehr verursachen würden. Sie können DNS aber auch parallel zu WINS verwenden.

Direct Hosted TCP vs. NT-Domänen

⚠ Beachten Sie, dass DNS im Gegensatz zu WINS keine Suche nach Namen mit einem bestimmten Suffix erlaubt. Dies hat die unangenehme Konsequenz, dass Sie bei *Direct Hosted TCP* keine NT-Domänen mehr verwenden können, denn diese benötigen die Namenssuche, damit Clients den Domänencontroller finden können. *Direct Hosted TCP* setzt einen Active-Directory-Domänencontroller voraus; darum gibt es bisher keine Möglichkeit, Samba als Domänencontroller zu verwenden und gleichzeitig NetBIOS über TCP/IP abzuschalten.

NetBIOS-Namen und DNS

Bei *NetBIOS over TCP* koexistieren zwei Namensräume, nämlich der von NetBIOS und der von DNS. Um Problemen aus dem Weg zu gehen, sollten unter *NetBIOS over TCP* der NetBIOS-Name und der DNS-Name eines Rechners identisch sein. Das können Sie konsequent nur mit WINS, nicht aber mit der *Broadcast Name Resolution* realisieren – bei letzterer ist jeder Rechner Herr über seinen eigenen Namen, während Sie bei WINS wenigstens theoretisch die Möglichkeit haben, Wildwuchs zu verhindern. Ein WINS-Server kann die bei ihm registrierten NetBIOS-Namen an einen DNS-Server weitergeben. Microsoft-DNS-Server übernehmen entsprechende Einträge automatisch, unter Samba wird diese Funktion über dynamisches DNS (DDNS) emuliert.

Zeichenvorrat

Ein weiteres Problem beim Nebeneinander von NetBIOS-Namen und DNS sind die in Rechnernamen erlaubten Zeichen; während DNS nur alphanumerische Zeichen und den Bindestrich zuläßt, gestattet NetBIOS sehr viel mehr Zeichen. Das ist vor allem dann ein mögliches Problem, wenn Sie ein bestehendes Windows-Netzwerk auf DNS umstellen wollen.

6.2 WINS im Detail

In heterogenen Netzen ist WINS die bevorzugte Methode zur Namensauflösung, um die Netzlast durch Broadcasts gering zu halten und trotzdem eine gemeinsame Sicht auf die Namen zu ermöglichen. Verschiedene Probleme der Broadcast-basierten Namensverwaltung werden vermieden:

- Bei der NetBIOS-Namensvergabe per Broadcast kann es passieren, dass sich ein Rechner einen Namen wünscht, der bereits vergeben ist. Damit wäre dieser Rechner zunächst vom NetBIOS-Netzverkehr ausgeschlossen.
- Bei der NetBIOS-Namensvergabe per Broadcast haben Sie keine wirkliche Kontrolle, wer welchen Namen bekommt. Insbesondere wird es schwierig, den NetBIOS-Namensraum mit dem (sicherlich auch vorhandenen) DNS-Namensraum abzugleichen.
- Eine reine DNS-Namensauflösung funktioniert nur bei *Direct Hosted TCP* und das wiederum funktioniert nicht mit älteren Systemen (Windows 9x, Windows NT, Samba 2.*x*).

Tabelle 6.2: NetBIOS-Gruppennamen (Auswahl)

Name	Bedeutung
`GRUPPE`⟨*03*⟩	Generischer Name, von allen Mitgliedern von `GRUPPE` registriert
`GRUPPE`⟨*1c*⟩	Domänencontroller bzw. Netlogon-Server
`GRUPPE`⟨*1d*⟩	Local Master Browsers

WINS vs. DNS

Auf den ersten Blick fällt Ihnen bei WINS vielleicht eine gewisse Ähnlichkeit zu DNS auf. Im Unterschied zum klassischen DNS fehlt aber unter anderem die Hierarchie im Namensraum – es gibt keine »WINS-Domains«. (Da WINS vom NetBEUI-Konzept eines nicht allzu großen Netzes ausgeht, ist das auch nicht weiter schlimm.)

WINS-Funktion

WINS basiert auf der Idee eines zentralen WINS-Servers, der eine dynamische WINS-Datenbank vorhält. Ein neu hinzukommender NetBIOS-Client meldet beim WINS-Server seinen Namen an und bekommt diesen für eine gewisse Zeit (TTL; *Time To Live*) fest zugeteilt – sofern der Name nicht schon anderweitig vergeben ist. Damit alle Rechner den WINS-Server zur Auflösung der NetBIOS-Namen zu IP-Adressen verwenden können, müssen Sie die Adresse des WINS-Servers auf jedem Rechner bekannt geben. Das kann per Hand oder per DHCP geschehen.

WINS ist nicht auf das lokale Netzsegment beschränkt, mehrere Subnetze können denselben WINS-Server verwenden. Das ist sogar zwingend nötig, wenn Ihr Netz mehrere physikalische Segmente umfasst, da Namensauflösung über Broadcast dann nicht alle Rechner erreichen würde.

Sie können auch mehrere WINS-Server im Netz haben, die ihre Daten untereinander replizieren. Allerdings sind Sie dabei für den Moment noch auf Windows angewiesen, oder Sie verwenden `samba4WINS`, einen replizierenden WINS-Server der Firma SerNet, zu finden unter `http://enterprisesamba.org/`.

Datei `lmhosts`

Alternativ zu WINS können Sie auch auf jedem SMB-Rechner eine Datei namens `lmhosts` pflegen, in der IP-Adressen zu NetBIOS-Namen aufgelöst werden, etwa:

```
# Beispiel für eine Datei "lmhosts"
192.168.0.1        albus
192.168.0.100      minerva
```

Unter Samba können Sie die `lmhosts` unter `/etc/samba/lmhosts` anlegen. In größeren Netzen macht es natürlich nicht wirklich Laune, alle Rechner immer mit der neuesten Version dieser Datei zu versorgen.

Samba spricht die verschiedenen Quellen für Namensinformationen – `lmhosts`-Datei, WINS, Broadcast und Unix-übliche Rechnernamensauflösung über DNS bzw. `/etc/hosts` – in einer Reihenfolge an, die vom Parameter »`name resolve order`« vorgegeben wird. Der Standardwert ist

```
name resolve order = lmhosts host wins bcast
```

Das heißt, Namen werden zuerst in der lokalen `lmhosts`-Datei nachgeschlagen, dann wird die Unix-Methode probiert, bevor auf den WINS-Server und das Broadcast-Verfahren zurückgegriffen wird.

6.3 NetBIOS-Namen eines Samba-Rechners

DNS-Name als NetBIOS-Name

Auch ein Samba-Rechner sollte einen eigenen NetBIOS-Namen haben. Normalerweise beansprucht ein Samba-Rechner seinen DNS-Hostnamen als NetBIOS-Name. Das sollte auch möglichst so bleiben, da es keine gute Idee ist, einem Rechner einen vom DNS-Namen abweichenden NetBIOS-Namen zuzuweisen. Trotzdem haben Sie die Möglichkeit, das zu tun:

```
netbios name = ALBUS
```

Der Samba-Rechner wird dann versuchen, den Namen ALBUS beim WINS-Server (wenn vorhanden) zu beantragen. Sie sollten darauf achten, dass Sie keine NetBIOS-Namen verwenden, die länger als 15 Zeichen sind.

Mehrere NetBIOS-Namen

Ein Samba-Rechner kann sogar mehrere NetBIOS-Namen haben:

```
netbios aliases = DUMBLEDORE, HEADMASTER
```

Clients können den Samba-Rechner dann unter mehreren Namen ansprechen, was unter anderem das Aufsetzen mehrerer »virtueller« Samba-Server auf demselben Rechner erlaubt.

Arbeitsgruppe oder Domäne

Normalerweise ist ein SMB-Rechner Mitglied einer Arbeitsgruppe oder Windows-NT-Domäne. Einem Samba-Rechner muss dafür mindestens der Name der Arbeitsgruppe oder Domäne mitgeteilt werden. Beides ermöglicht der Parameter workgroup in der Datei smb.conf, etwa so:

```
workgroup = HOGWARTS
```

Wenn der Rechner Mitglied einer NT- oder gar einer ADS-Domäne sein soll, ist allerdings ein bisschen mehr Aufwand nötig als nur ein Name in smb.conf!

Kommentar

Ebenfalls zur NetBIOS-Namensauflösung gehört die Angabe des Kommentars, der einem Rechner in der »Netzwerkumgebung« zugeordnet wird. Ein entsprechender Eintrag in der smb.conf sieht so aus:

```
server string = Server in der Arbeitsgruppe HOGWARTS
```

Standardmäßig wird die verwendete Samba-Version als Kommentar angegeben. Sinnvoller ist es, z. B. die Funktion des Rechners im Kommentar zu verewigen.

WINS-Server

Es bringt überhaupt nichts, auf einem Samba-Rechner diverse NetBIOS-Namen einzutragen, wenn nicht bekannt ist, bei welchem WINS-Server sie registriert werden sollen. Dafür steht Ihnen der smb.conf-Parameter »wins server« zur Verfügung. Er übernimmt die IP-Adresse des WINS-Servers:

```
wins server = 192.168.0.1
```

Theoretisch können Sie natürlich auch den DNS-Namen des WINS-Servers angeben. Damit machen Sie sich aber davon abhängig, dass DNS auf jeden Fall funktioniert – oft keine gute Idee.

mehrere WINS-Server

Haben Sie mehrere WINS-Server für Ihr Netzwerk konfiguriert (geht im Moment nur auf Windows-Basis oder mit samba4WINS), dann können Sie diese auch angeben:

```
wins server = 192.168.0.1 172.16.22.33
```

Kann der erste Server nicht erreicht werden, versucht Samba den zweiten zu kontaktieren und so weiter.

Gültigkeitsdauer

Beim Registrieren von NetBIOS-Namen auf dem WINS-Server kann Samba sich eine bestimmte Gültigkeitsdauer (*time to live*, TTL) wünschen:

```
max ttl = 259200
```

Die TTLs werden in Sekunden angegeben. Als Wert ist im Beispiel der Standardwert aufgeführt: 3 Tage.

6.4 Samba als WINS-Server

WINS-Server unter Unix/Linux

Schon lange ist es möglich, mit Samba einen WINS-Server unter Unix/Linux aufzusetzen. Verantwortlich für diese Funktion ist der `nmbd`. Möchten Sie ihn als WINS-Server verwenden, reicht im Prinzip der Eintrag

```
wins support = yes
```

in der Datei `smb.conf`. (Der Eintrag `wins server` darf nicht gleichzeitig aktiv sein!) Auf UDP-Port 137 bietet der `nmbd` danach seine Dienste als WINS-Server an. Den Eintrag sollten Sie allerdings nur auf einem einzigen Rechner im Subnetz aktivieren.

Datei `wins.dat`

Unter Samba legt der `nmbd` registrierte NetBIOS-Namen in der Datei `wins.dat` ab, die standardmäßig im Verzeichnis `/var/lock/` oder `/var/lib/samba/` zu finden ist. Es handelt sich um eine normale ASCII-Datei:

```
# cat /var/lib/samba/wins.dat
VERSION 1 0
"ALBUS#00" 1107768483 192.168.0.100 64R
"ALBUS#03" 1107768483 192.168.0.100 64R
"ALBUS#20" 1107768483 192.168.0.100 64R
"HOGWARTS#00" 1107768569 255.255.255.255 e4R
"HOGWARTS#1e" 1107768569 255.255.255.255 e4R
```

Die Datei enthält in erster Linie die IP-Adressen für einzelne Rechner. Angegeben ist jeweils der NetBIOS-Name (z. B. `ALBUS`) inklusive NetBIOS-Suffix (z. B. `#00`). Auch Arbeitsgruppen- oder Domänen-Namen tauchen auf (z. B. `HOGWARTS#1e`).

Die zweite Spalte in einem Namenseintrag in `wins.dat` ist das »Verfallsdatum« des Namens (unixtypisch in Sekunden seit dem 1.1.1970, 0 Uhr UTC). Sie können Namen permanent machen, indem Sie diesen Wert auf 0 setzen:

```
"ALBUS#00" 0 192.168.0.100 64R
```

Die letzte Spalte enthält die sogenannten *NetBIOS flags*. Diese geben im Prinzip an, um welche Sorte Namen es sich handelt und welche Arten von Namensauflösung der benannte Rechner verwendet (Broadcast, WINS oder Kombinationen von beidem).

Das direkte Ändern von `wins.dat` ist im Moment erlaubt und funktioniert; wenn künftige Versionen von Samba anfangen, WINS-Replikation zu unterstützen, kann es allerdings zu Änderungen kommen.

TTLs

Wie lange die entsprechenden Einträge gültig bleiben, hängt von der jeweiligen TTL ab. Die Grenzen für die zu vergebenden TTLs können Sie mit den folgenden Parametern definieren:

```
min wins ttl =  21600
max wins ttl = 518400
```

Die TTLs werden in der Einheit »Sekunden« angegeben. Als Werte sind im Beispiel die Standardwerte aufgeführt: 6 Stunden bzw. 6 Tage.

Rückgriff auf DNS

In manchen Fällen kann es passieren, dass der WINS-Server einen angefragten Namen nicht zur IP-Adresse auflösen kann. Dann ist es nützlich, wenn er seinerseits den DNS-Server fragt. Das können Sie über den Eintrag

```
dns proxy = yes
```

in der `smb.conf` realisieren. Der `nmbd` fragt dann den in der Datei `/etc/resolv.conf` eingetragenen Nameserver.

Microsoft sieht vor, dass die auf einem WINS-Server dynamisch eingetragenen Rechner mit dem lokalen DNS-Server abgeglichen werden, so dass auch dieser entsprechende Anfragen in Zukunft beantworten kann. Die Windows-Systeme verwenden dafür Protokolle, die keinem Standard entsprechen, so dass Samba nicht den Original-Mechanismus nachbilden kann. Allerdings ist es möglich, die Daten eines Samba-WINS-Servers mit einem DDNS-konformen DNS-Server (etwa BIND 9) abzugleichen. Der `smb.conf`-Parameter »`wins hook`« ermöglicht es, ein Programm anzugeben, das im Falle einer Änderung der WINS-Datenbank ausgeführt wird und den entsprechenden DDNS-Transfer durchführt. Der Transfer sollte mit den DDNS-typischen kryptografischen Methoden (z. B. DSA-Schlüssel) abgesichert werden. Weitere Informationen finden Sie in der Samba- und BIND-Dokumentation.

Übungen

6.1 [!3] Setzen Sie einen WINS-Server für Ihr Schulungsnetzwerk auf. Kooperieren Sie dabei mit den anderen Kursteilnehmern, damit Sie den Server testen können bzw. damit sich die Server nicht gegenseitig in die Quere kommen.

6.5 Überprüfen der NetBIOS-Namensauflösung

Wie Sie gesehen haben, ist die NetBIOS-Namensauflösung nicht ganz simpel. Daher sollten Sie insbesondere nach dem Aufsetzen eines WINS-Servers ausreichend Zeit für Tests investieren. Das zentrale Werkzeug für das Testen der Namensauflösung (auch unter Windows) ist das Kommando `nmblookup`:

```
# nmblookup minerva
querying minerva on 192.168.0.255
192.168.0.100 minerva<00>
```

Wie Sie sehen, schickt `nmblookup` dabei einen IP-Broadcast ins lokale Netz, der die entsprechende Nachfrage enthält. Eine positive Antwort zeigt zwar, dass die Namensauflösung für den betreffenden Rechnernamen grundsätzlich funktioniert. Allerdings können Sie nicht sicher sein, dass die Anfrage wirklich von Ihrem WINS-Server beantwortet wurde. Dafür müssen Sie dessen IP-Adresse explizit als Ziel angeben:

IP-Broadcast

Explizite Anfrage an den WINS-Server

```
# nmblookup -U 192.168.0.1 -R minerva
querying minerva on 192.168.0.1
192.168.0.100 minerva<00>
```

Außerdem ist es notwendig, die Anfrage an den WINS-Server mit der Option `-R` (engl. *recursion desired bit*) zu stellen, damit nicht eine normale Broadcast-Anfrage an den Server gestellt wird. Auf eine solche würden Sie keine Antwort bekommen.

Neben einfachen Namens-Anfragen können Sie mit `nmblookup` auch eine Rückwärtsauflösung durchführen:

Rückwärtsauflösung

```
# nmblookup -A 192.168.0.100
Looking up status of 192.168.0.100
        MINERVA         <00> -         H <ACTIVE>
        MINERVA         <03> -         H <ACTIVE>
◁◁◁◁◁
```

Mit `nmblookup` können Sie auch Suchmuster, wie sie von der Shell bekannt sind, verwenden. So können Sie zum Beispiel alle Rechner anzeigen, die Freigaben anbieten:

```
# nmblookup '*#20'
querying * on 192.168.0.255
192.168.0.1 *<20>
192.168.0.100 *<20>
```

Suchmuster sollten Sie dabei selbstverständlich mit Hochkommata vor der Shell verbergen.

Mit `nmblookup` könne Sie auch nach den NetBIOS-Suffixen suchen, und so herausfinden, welcher Rechner welche Funktion in Ihrem lokalen Netz hat (hierzu mehr in Abschnitt 6.6). Ein Sonderfall der Suche nach Rechnern ist die Suche nach allen Samba-Maschinen im Netz:

```
# nmblookup __samba__
uerying __samba__ on 192.168.0.255
92.168.0.1 __samba__<00>
92.168.0.100 __samba__<00>
```

Bei `__samba__` handelt es sich nicht um einen offiziellen NetBIOS-Namen, aber die Samba-Entwickler waren so nett und haben es trotzdem integriert …

Protokolldatei

Verwenden Sie die Protokolldatei des `nmbd` (meist `log.nmbd`) zur Fehlersuche. Es hilft oft, die Protokollierstufe auf 2 oder 3 zu setzen; höhere Werte führen zu großen Mengen kryptischer Ausgabe und sind vor allem für Samba-Entwickler interessant.

Übungen

6.2 [!2] Versuchen Sie, eine Liste aller Rechner im lokalen Netz aufzustellen, die zu Ihrer Arbeitsgruppe/Domäne gehören. (*Tipp:* Verwenden Sie `nmblookup` und schauen Sie in Tabelle 6.2.)

6.6 Suchdienst (*Browsing*)

NetBIOS sieht eine Suchfunktion für das Netzwerk vor, die es ermöglicht, auf jedem Rechner alle Rechner des Netzwerks im Blick zu behalten. Diese Funktion wird auch **Suchdienst** (*Browsing*) genannt. Jeder Rechner mit einer Freigabe verschickt Broadcasts an alle anderen Rechner im Netzwerksegment, um diesen seine Ressourcen bekannt zu machen. Jeder Rechner erzeugt aus den empfangenen Broadcasts eine lokale Liste der bekannten Freigaben (engl. *browse list*).

Suchdienst

Netzwerkumgebung

Die *browse list* kann unter Windows in der »Netzwerkumgebung« eingesehen werden. Die »Netzwerkumgebung« ist allerdings nicht immer aktuell, da das beschriebene Broadcastsystem seine Zeit braucht. Insbesondere kann ein Rechner, der sich nicht sofort meldet, nicht sofort aus der Suchliste entfernt werden.

Das Samba-Äquivalent zur Windows-Netzwerkumgebung ist das `smbtree`-Kommando. Hiermit können Sie zum Beispiel die Rechner der aktuellen Arbeitsgruppe und ihre Freigaben auflisten:

```
$ smbtree -N
HOGWARTS
  \\ALBUS                    Samba 3.5.4
        \\ALBUS\ADMIN$           IPC Service (Samba 3.5.4)
        \\ALBUS\IPC$             IPC Service (Samba 3.5.4)
        \\ALBUS\test
  \\MINERVA                  Samba-Testserver für den Samba-Kurs
```

```
\\MINERVA\ADMIN$          IPC Service (Samba-Testserver für den Samba-Kurs)
\\MINERVA\IPC$            IPC Service (Samba-Testserver für den Samba-Kurs)
\\MINERVA\test-2          Testfreigabe für den Samba-Kurs
```

Broadcast in großen Netzen: ein Problem

In großen Netzen würden Browsing-Listen sehr groß werden und die vielen Broadcasts das Netz lahmlegen, deshalb sieht SMB spezialisierte Rechner vor, die für alle anderen Netzwerkteilnehmer Browsinglisten vorhalten, die sogenannten *Browser*[1]. Sie registrieren sich unter dem NetBIOS-Namen `_MSBROWSER_`, damit sie von den anderen Systemen gefunden werden können. Es gibt *Local Master Browser* und *Domain Master Browser*.

Local Master Browser

Ein *Local Master Browser* (LMB) hält die Suchliste für ein physikalisches Netzsegment vor. Zum LMB wird ein Rechner über eine »Schönheitskonkurrenz«, bei dem die Art des Betriebssystems und die bisherige Rolle im Netz der verschiedenen Kandidaten verglichen werden. Aus diesen Parametern wird ein »OS-Level« bestimmt, der die Basis einer Art interner Hackordnung zwischen den verschiedenen Windows-Varianten darstellt. Die moderneren Systeme haben einen höheren OS-Level als die älteren, Server-Varianten dominieren Workstation-, Home- und Professional-Varianten. Bei Samba können Sie den OS-Level mit der Variablen »`os level`« einstellen; mit der Variablen »`local master`« (standardmäßig eingeschaltet) legen Sie fest, ob Ihr Samba-Server überhaupt am Wettbewerb teilnehmen soll.

OS-Level

Innerhalb von Samba kümmert sich der `nmbd` um die Teilnahme am Suchdienst und am Browser-Wettbewerb. Es ist mit Hilfe von »`os level`« immer möglich, den Wettbewerb zu gewinnen, indem Sie einen höheren Wert einstellen, als ein Windows-Rechner erreichen kann (34 ist genug).

Steht die Variable »`preferred master`« auf `yes`, dann erzwingt Samba beim Start einen neuen LMB-Wettbewerb (dessen Resultat dann schon festgelegt ist …). Achten Sie aber darauf, nur ein einziger Rechner im Netz (Samba oder Windows) sich für den *preferred master* hält; wenn es zwei sind, dann lösen sie ständig neue Wettbewerbe aus, um sich diese Ehre abzujagen.

Im normalen Betrieb wird der Schönheitswettbewerb etwa alle 11 bis 15 Minuten wiederholt. Üblicherweise gewinnt der aktuelle LMB, da er ja nicht nur vorher gewonnen hat, sondern außerdem noch einen »Amtsbonus« bekommt.

Der Standardwert für »`os level`« ist 20, was heißt, dass Samba alle Windows-Rechner mit der Ausnahme eines Windows-NT-4.0-Domänencontrollers aussticht. Wenn Sie also keinen solchen Rechner im Netz haben und Ihr Samba-Server nicht korrekt konfiguriert ist, kann es sein, dass Ihr Server sein lokales Netz von einem möglichen domänenweiten Suchdienst (siehe unten) abkoppelt. Beachten Sie, dass Samba sich standardmäßig um die Position des LMB bewirbt (»`local master`« ist eingeschaltet). (Siehe auch Übung 6.6.)

Suchdienst: Nur im selben Subnetz

Der Suchdienst basiert auf Broadcast und funktioniert darum nur zwischen Rechnern im selben physikalischen Subnetz (bei NetBEUI-basierten Netzen ist das irrelevant, weil es nur ein physikalisches Netz gibt). Um in NetBT-basierten Netzen die Suchdienste mehrerer physikalischer Subnetze zu vereinigen, können Sie einen Rechner zum »Domänen-Suchdienst« (*Domain Master Browser*, DMB) erklären. Dieser Rechner erhält die Suchlisten aller LMBs, vereinigt sie und schickt sie zurück, so dass die einzelnen Rechner in den Subnetzen Zugriff auf die komplette Liste bekommen. Die Aktualisierungsintervalle zwischen LMBs und DMBs sind berüchtigt – es dauert mitunter mehr als eine Stunde, bis eine neue Ressource im Nachbarnetz bekannt ist. Sie können einen Samba-Server als DMB konfigurieren, indem Sie die Variable »`domain master`« auf »`yes`« (oder ein Synonym davon) setzen.

Domain Master Browser

[1] Nicht zu verwechseln mit den Programmen, die zum Zugriff auf das World Wide Web eingesetzt werden.

Tabelle 6.3: Samba-Parameter für den Suchdienst (*browsing*)

Parameter	DMB	LMB	kein MB
`domain master`	`yes`	`no`	`no`
`local master`	`yes`	`yes`	`no`
`preferred master`	`yes`	`yes`	`no`
`os level`	`34`	`34`	`0`

Die Verwendung des Begriffs *Domain* in »*Domain Master Browser*« hat nichts mit dem Konzept der Windows-NT-Domänen (siehe Abschnitt 5.3) zu tun – Sie können auch in einer einfachen Arbeitsgruppe einen DMB haben.

Der DMB ist meistens auch LMB in seinem physikalischen Subnetz, aber das ist nicht strikt vorgeschrieben. Samba besteht allerdings darauf – setzen Sie also »`domain master`« nur dann, wenn Sie auch alles Nötige getan haben, damit der Rechner LMB wird.

Sollten Sie allerdings eine NT-Domäne verwenden, dann muss der PDC der NT-Domäne gleichzeitig auch DMB sein. Insbesondere müssen Sie verhindern, dass Ihr Samba-Server sich zum DMB erklärt, bevor der (Windows-basierte) PDC dazu kommt – ein Rechner wird DMB, indem er einen bestimmten NetBIOS-Namen beansprucht, und dabei gilt »wer zuerst kommt, mahlt zuerst«. Umgekehrt wird ein Samba-Server, der selber als PDC konfiguriert ist, automatisch immer auch versuchen, sich zum DMB zu machen.

Suchdienst und WINS

Der DMB registriert seine IP-Adresse beim WINS-Server unter dem Namen der Arbeitsgruppe gefolgt von `<1B>`. Die LMBs dagegen registrieren ihre IP-Adresse unter dem Namen der Arbeitsgruppe gefolgt von `<1D>`. Da es pro physikalischem Subnetz einen LMB gibt, gibt es entsprechend pro Subnetz einen `<1D>`-Namen; der DMB sucht nach diesen Namen und kann so die LMBs finden.

Suchdienst braucht WINS

Der Domänen-Suchdienst setzt zwingend voraus, dass Sie WINS zur Auflösung von NetBIOS-Namen verwenden. Sollten Sie statt dessen NetBIOS-Namen über Broadcast auflösen, werden Sie zwei Effekte bemerken: Zum einen werden die meisten LMBs keinen DMB finden können, da sie nur auf ihrem eigenen physikalischen Subnetz suchen können, der DMB aber in einem anderen steht. Zum anderen kann ein Client, der einen interessanten Namen in einer domänenweiten Suchliste findet, nicht auf den betreffenden Rechner zugreifen, falls sich dieser in einem anderen physikalischen Subnetz befindet – weil er dessen NetBIOS-Namen nicht in eine IP-Adresse auflösen kann.

Nur Samba-Server: kein DMB nötig

In einem Netz, das nur einen Samba-Server verwendet, können Sie auch ohne DMB auskommen, da Samba-basierte LMBs sich direkt austauschen können. Dies können Sie über den Parameter »`remote browse sync`« aktivieren, indem Sie entweder gezielt die IP-Adresse des oder der entfernten LMB oder die Broadcast-Adresse des oder der entfernten Subnetze(s) angeben:

```
remote browse sync = 192.168.1.1 192.168.2.1          Gezielt die LMBs …
remote browse sync = 192.168.1.255 192.168.2.255      … oder per Broadcast
```

In diesem Fall müssen Sie außerdem noch dafür sorgen, dass die Namensauflösung funktioniert – über WINS, DNS, `/etc/hosts` oder andere Methoden.

In Tabelle 6.3 finden Sie eine Zusammenstellung der für den Suchdienst relevanten Konfigurationsparameter.

Suchdienst und `nmblookup`

Sie können `nmblookup` verwenden, um alle Rechner im lokalen Subnetz zu finden, die einen LMB-Posten anstreben. Suchen Sie dazu nach Namen mit dem Suffix `<1E>`:

```
# nmblookup HOGWARTS#1e
querying HOGWARTS on 192.168.0.255
192.168.0.2 HOGWARTS<1e>
192.168.0.100 HOGWARTS<1e>
```

Auch wer nach dem LMB-Wettbewerb das Rennen gemacht hat, lässt sich mit `nmblookup` herausfinden:

```
# nmblookup HOGWARTS#1d
querying HOGWARTS on 192.168.0.255
192.168.0.100 HOGWARTS<1d>
```

Der LMB läßt sich alternativ auch mit dem Namen `__MSBROWSER__` ansprechen:

```
# nmblookup HOGWARTS#__MSBROWSER__
querying HOGWARTS on 192.168.0.255
192.168.0.100 HOGWARTS<00>
```

Ähnliche Informationen bekommen Sie übrigens auch mit `smbclient`: Suchdienst und `smbclient`

```
# smbclient -NL 192.168.0.100
<<<<<
Anonymous login successful
Domain=[HOGWARTS] OS=[Unix] Server=[Samba 3.5.4]

        Server               Comment
        ---------            -------
        ALBUS                Samba 3.5.4
        MINERVA              Samba-Testserver für den Samba-Kurs

        Workgroup            Master
        ---------            -------
        HOGWARTS             ALBUS
```

Wenn der Suchdienst nicht funktioniert, kann das übrigens daran liegen, dass Sie kein Gastkonto definiert haben (mit »`guest account`«) bzw. das Gastkonto nicht auf einen gültigen Linux-Benutzer abgebildet wird. Beim Zugriff auf einen Rechner über den Suchdienst versucht ein Client, sich an die `IPC$`-Freigabe zu binden und so die Liste der Freigaben und Drucker für diesen Rechner zu bestimmen. Suchdienst-Probleme

Im Falle von Windows 2000 und späteren Versionen ist das möglicherweise kein Problem, da diese den `IPC$`-Zugriff mit dem Namen des gerade angemeldeten Benutzers versuchen, wenn sie als Gast nicht zu Potte kommen. Frühere Versionen, insbesondere Windows 95/98/ME, tun das nicht, so dass das Unterfangen fehl schlägt.

Übungen

6.3 [!1] Verwenden Sie `nmblookup`, um die Adressen der Rechner zu bestimmen, die in Ihrem Netz für den lokalen und den Domänen-Suchdienst zuständig sind. Handelt es sich dabei um Windows- oder um Samba-Server?

6.4 [3] Beobachten Sie die »Schönheitskonkurrenz« zur LMB-Bestimmung mit `ethereal` oder einem anderen Netzwerksniffer.

6.5 [!3] Konfigurieren Sie Ihren Rechner als *Local Master Browser* und prüfen Sie, ob er diesen Posten in Ihrem Netz zugesprochen bekommt (mit `nmblookup`). Testen Sie seine Funktion, etwa mit `smbtree` oder der »Netzwerkumgebung« auf einem Windows-Rechner. – *Wichtig:* Versuchen Sie nicht, auf

demselben physikalischen Subnetz zwei LMBs zu etablieren. Diese Übung funktioniert nur, wenn Sie Ihr physikalisches Subnetz für sich alleine haben, also zum Beispiel zu Hause oder in einer geeignet konfigurierten »virtuellen« Umgebung, seltener in einem Schulungsraum.

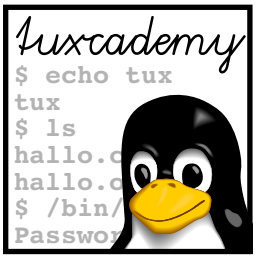
6.6 [!2] Warum würde ein Samba-Server mit Standardeinstellungen sein lokales Netz von einem domänenweiten Suchdienst abkoppeln, wenn es keinen NT-Domänencontroller gibt?

Kommandos in diesem Kapitel

nmblookup NetBIOS-over-TCP/IP-Client für Namensdienste `nmblookup`(1) 89
smbtree Ein textbasierter SMB-Umgebungs-Browser (à la Windows-„Netzwerkumgebung“) `smbtree`(1) 90

Zusammenfassung

- NetBIOS-Namen dienen zur Identifizierung von Rechnern. Heute wird zur tatsächlichen Implementierung eines Namensschemas oft DNS herangezogen.
- WINS basiert auf einem zentralen WINS-Server, der eine dynamische WINS-Datenbank vorhält.
- Jeder Samba-Rechner sollte auch einen NetBIOS-Namen haben.
- Samba kann als WINS-Server fungieren.
- Der WINS-Dienst kann mit dem Programm `nmblookup` getestet werden.
- Browsing dient dazu, die Ressourcen aller Rechner im ganzen Netzwerk bekanntzumachen.



7

Authentisierung und Zugriffsrechte

Inhalt

Lernziele

- Zugriffsrechte für Verzeichnisse und Dateien definieren können
- Benutzer lokal auf dem Samba-Server einrichten können
- Für die SMB-Authentisierung einen externen Passwortserver verwenden

Vorkenntnisse

- Erfahrung im Umgang mit Unix-Zugriffsrechten
- Kenntnisse über Windows-Zugriffsrechte und ACLs sind hilfreich

7.1 Überblick

Nicht jeder beliebige Benutzer sollte auf eine Netzwerkressource zugreifen dürfen. Zugriffsbeschränkungen für Netzwerkressourcen können einerseits auf Dateisystemebene und andererseits auf Ebene der Dienste selber stattfinden. So können Sie beispielsweise bei einem FTP-Server sämtlichen Benutzern im Netzwerk den Zugriff erlauben (engl. *Anonymous FTP*), was einem zugreifenden Benutzer allerdings nichts bringt, solange die UID des FTP-Server-Prozesses keine passenden Zugriffsrechte auf die entsprechende Datei besitzt.

Die Zugriffsbeschränkungen auf Dateisystemebene erfolgen auf einem Samba-Server über die bekannten Unix-Zugriffsrechte. Auf Dienstebene ist es in der Regel möglich, Zugriffe nur für bestimmte Rechner zu erlauben; auch benutzerspezifische Beschränkungen sind hier teilweise möglich.

Dienstebene

Wie sieht es unter Samba aus? Auf Dienstebene können Sie unter Samba verschiedene Zugriffseinstellungen durchführen. Sie können beispielsweise für jede einzelne Freigabe festlegen, ob darauf nur lesend oder auch schreibend zugegriffen werden darf. Weiterhin können Sie Einschränkungen auf der Basis von IP-Adressen vornehmen.

Authentisierung einzelner Benutzer

SMB ermöglicht außerdem eine Authentisierung einzelner Benutzer. Auf Dienstebene können Sie festlegen, welche Benutzer welche Freigaben benutzen dürfen. Unter Windows ebenfalls möglich ist die Definition von Dateiattributen und Zugriffskontrolllisten (ACLs), die Zugriffsrechte definieren. Wenn Sie auf der Linux-Seite ein Dateisystem verwenden, das ACLs erlaubt (etwa ein ext-Dateisystem oder XFS), können die meisten dieser Möglichkeiten auch von einem Samba-Server unterstützt werden.

Unix-Zugriffsrechte

Zuunterst und letztgültig legen aber die Zugriffsrechte des Unix-Dateisystems fest, welche Zugriffe erlaubt sind – und schon haben Sie diverse Probleme:

- Ein Windows-Client kennt keine Unix-Zugriffsrechte
- Ein Windows-Client möchte vielleicht Dateiattribute setzen, die unter Unix kein Äquivalent haben
- Ein Windows-Benutzer kann nur auf Dateien zugreifen, wenn er die UID eines Unix-Benutzers verwendet
- Ein Windows-Benutzer hat einen Windows-Benutzernamen, mit dem er sich normalerweise anmeldet, der aber nicht notwendigerweise automatisch auf der Unix-Seite bekannt ist.

Samba muss also Rechte und Benutzer zwischen den »Welten« übersetzen können.

Dieses und die folgenden Kapitel beschäftigen sich mit diesen Problemen und ihren Lösungen in der Praxis.

7.2 Zugriffsbeschränkungen auf Netzwerkebene

Samba kennt verschiedene Parameter, die Zugriffe auf einzelne Netzwerkbereiche oder einzelne Rechner einschränken. Diese Möglichkeiten sind mit der gebotenen Vorsicht zu genießen; insbesondere eine Zugriffsbeschränkung auf der Basis von IP-Adressen kann durch das Vortäuschen falscher Adressen (engl. *IP-Spoofing*) einfach unterlaufen werden.

Für die Zugriffssteuerung über IP-Adressen können Sie die folgenden Parameter verwenden:

hosts deny

hosts deny = ⟨*Rechner*⟩ ermöglicht den Ausschluss einzelner Rechner oder Netzwerke. ⟨*Rechner*⟩ kann eine IP-Adresse mit oder ohne Netzmaske oder ein DNS-Name sein:

```
hosts deny = 192.168.0.5                    schließt einen Rechner aus
hosts deny = 192.168.0.0/24                 schließt ein Subnetz aus
hosts deny = draco.example.com              schließt einen Rechner aus
```

Sie können jeweils Teile weglassen, diese werden dann als Netzwerkbereich bzw. DNS-Domäne interpretiert:

```
hosts deny = 192.168.0.                     schließt ein Subnetz aus
```

Ein solcher Netzwerkbereich sollte immer mit einem Punkt aufhören.

Sie dürfen mehrere Rechner oder Netze angeben, die dann durch Leerzeichen, Tabs oder Kommas getrennt sein müssen:

```
hosts deny = 192.168.0.5, 192.168.1.0/28, pc42.example.com
```

Es gibt auch ein Schlüsselwort `EXCEPT`, mit der Sie Ausnahmen zulassen können:

```
hosts deny = 192.168.0.0/24 EXCEPT 192.168.0.42
```

Nähere Informationen über die Syntax von Zugriffskontrolllisten können Sie der Manpage `hosts_access`(5) entnehmen (sofern diese auf Ihrem System installiert ist).

`hosts allow =` ⟨*Rechner*⟩ ist sozusagen das Gegenteil von `hosts deny`. Damit können Sie den Zugriff erlauben, wenn er sonst von `hosts deny` verboten worden wäre. Die Syntax für die Rechner oder Netze ist genau wie bei `hosts deny`.

Sowohl `hosts allow` als auch `hosts deny` können entweder im `[global]`-Abschnitt der `smb.conf`-Datei oder in der Konfiguration einer speziellen Freigabe auftauchen. Definitionen auf der globalen Ebene gelten für alle Freigaben, sofern die Freigaben keine eigenen Definitionen enthalten.

Wenn die `hosts allow`- und die `hosts deny`-Listen zueinander im Widerspruch stehen, hat die `hosts allow`-Liste Vorrang:

```
hosts deny = 192.168.0.0/24
hosts allow = 192.168.0.1
```

ist dasselbe wie

```
hosts deny = 192.168.0.0/24 EXCEPT 192.168.0.1
```

Die Parameter `hosts allow` und `hosts deny` können auch `allow hosts` und `deny hosts` geschrieben werden.

Die lokale Adresse `127.0.0.1` bekommt immer Zugriff, solange sie nicht ausdrücklich von einer »`hosts deny`«-Option ausgeschlossen wird.

Sie können übrigens jederzeit mit dem Programm `testparm` testen, ob ein bestimmter Rechner auf Ihren Server zugreifen darf:

```
# testparm /etc/samba/smb.conf fawkes.example.com 192.168.0.100
Load smb config files from /etc/samba/smb.conf
Processing section "[test]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Allow connection from fawkes.example.com (192.168.0.100) to albus
```

Dabei müssen Sie als Parameter zuerst die Konfigurationsdatei und dann DNS-Name und IP-Adresse des zu testenden Rechners angeben. Besser wäre natürlich ein Test vom betreffenden Rechner aus …

Die Parameter `hosts deny` und `hosts allow` beschränken zwar Zugriffe auf Netzwerkebene, setzen aber niemals eine eventuell erforderliche Benutzerauthentisierung außer Kraft – Benutzername und Kennwort werden also zusätzlich benötigt. Sollten Sie das nicht wollen – aber Sie wollen eigentlich immer –, *könnten* Sie den Parameter `hosts equiv` = ⟨*Datei*⟩ verwenden (auch wenn wir Ihnen dringend davon abraten). Für alle in der ⟨*Datei*⟩ aufgelisteten Rechner und Benutzer ist dann ein Zugriff ohne weitere Authentisierung möglich.

`interfaces` Hat Ihr Server mehrere Netzwerkschnittstellen, können Sie die Aktivität von Samba auf eine Teilmenge davon beschränken. Hierfür steht der Parameter `interfaces` zusammen mit `bind interfaces only = yes` zur Verfügung. Für `interfaces` können Sie über IP-Adresse oder Interface (z. B. `eth0`; ab Samba 2.0.5) das Gerät oder die Geräte festlegen, die für »ausgehendes« NBT verwendet werden. Standardwert ist dabei »alle aktiven Netzwerkschnittstellen, die Broadcast unterstützen, außer `127.0.0.1`«. Dabei reagiert Samba aber nach wie vor auf Anfragen, die auf anderen Netzwerkschnittstellen eingehen. Erst mit `bind interfaces only = yes` werden auch Zugriffe von außen auf die anderen Interfaces ignoriert:

```
interfaces = eth1
bind interfaces only = yes
```

Die tatsächliche Syntax ist etwas komplizierter; lesen Sie in `smb.conf`(5) nach.

Übungen

7.1 [2] Konfigurieren Sie Ihren Samba-Server so, dass alle Zugriffe vom Rechner eines anderen Schulungsteilnehmers abgewiesen werden. (Arbeiten Sie zum Testen mit dem anderen Schulungsteilnehmer zusammen, etwa indem Sie Ihre Rechner »über Kreuz« ausschließen.)

7.2 [2] Konfigurieren Sie Ihren Samba-Server so, dass alle Zugriffe von *allen* Rechnern im lokalen Netz bis auf den Rechner eines anderen Schulungsteilnehmers abgewiesen werden.

7.3 [!1] Was müssen Sie ändern, damit nur Zugriffe auf eine bestimmte Freigabe erlaubt oder abgewiesen werden, statt Zugriffen auf den kompletten Server?

7.4 [2] Wie können Sie einem bestimmten Rechner den Zugriff auf alle Freigaben Ihres Samba-Servers verweigern bis auf eine bestimmte?

7.5 [3] Probieren Sie aus, wie die Unix-Zugriffsrechte von Dateien auf einer Freigabe sich auf die Zugriffsmöglichkeiten von einem (Windows-)Client auswirken.

7.3 Einfache Zugriffsbeschränkungen für Freigaben

Haben Sie die Zugriffe auf der Netzwerkebene geregelt, können Sie sich den allgemeinen Regeln für die einzelnen Freigaben zuwenden. Hierfür stehen Ihnen verschiedene freigabespezifische Parameter zur Verfügung:

`available = yes` Hiermit können Sie eine Freigabe in der Datei `smb.conf` deaktivieren, was im Prinzip einem Auskommentieren des kompletten Konfigurationsblocks der Freigabe entspricht (aber natürlich viel bequemer ist).

browseable = no erzeugt eine versteckte Freigabe, die z. B. in der Windows-»Netzwerkumgebung« nicht mehr auftaucht. Dies ist nicht wirklich eine Maßnahme zur Zugriffskontrolle, da nach wie vor jeder versuchen kann, auf die Freigabe zuzugreifen, der ihren Namen kennt. (Ob das klappt oder nicht, hängt natürlich von den sonstigen Einstellungen der Freigabe ab.)

read only = no erlaubt allgemein Schreibzugriffe auf die Freigabe. Alternativ können auch `writeable = yes` bzw. `writable = yes` verwendet werden. Standardmäßig sind keine Schreibzugriffe möglich. Ausnahmen können jeweils mit `write list` gemacht werden.

guest ok = yes erlaubt den Zugriff auf die Freigabe ganz ohne vorherige Authentisierung.

guest only = yes erlaubt *ausschließlich* nicht authentisierten Benutzern den Zugriff.

guest account = ⟨*Unix-Benutzer*⟩ legt fest, unter welcher Unix-UID unauthentisierte Benutzer, die über `guest ok` oder `guest only` Zugriff bekommen haben, sich auf dem Dateisystem bewegen. Als Standard ist in der Regel `nobody` voreingestellt. (Dies ist kein freigabespezifischer, sondern ein globaler Parameter!)

Zugriffskontrolle für einzelne Dateien

Nicht nur dem Zugriff auf Freigaben, sondern auch dem Zugriff auf einzelne Dateien und Verzeichnisse können Sie einen Riegel vorschieben. Möchten Sie, dass bestimmte Dateien gar nicht von Samba freigegeben werden, müssen Sie diese in die (freigabespezifische) »`veto files`«-Liste aufnehmen. Die Einstellung

```
veto files = ./*/*~
```

bewirkt beispielsweise, dass alle Dateien mit einem Punkt am Anfang und Dateien mit einer Tilde am Ende (jeweils unter Windows kein gültiger Dateiname) nicht freigegeben werden. Beachten Sie, dass der Schrägstrich hier nicht als Pfadtrenner fungiert, sondern verschiedene Muster in der Liste voneinander trennt. In den Mustern sind die Sonderzeichen »*« und »?« erlaubt, die sich so benehmen wie in der Shell.

Löschen von Verzeichnissen

Dieser Parameter kann zu Problemen führen, wenn ein Benutzer von der Windows-Seite aus ein Verzeichnis löschen möchte, das auf der Samba-Seite nur Dateien enthält, die `veto files` unterliegen. Samba verbietet das, was für den Benutzer möglicherweise unerklärlich ist, da er die Dateien, die Samba vor ihm geheim hält, ja nicht sehen kann. Um Samba in diesem Fall das Löschen zu erlauben, müssen Sie den Parameter »`delete veto files`« auf `yes` setzen (sein Standardwert ist `no`).

`veto files` und Performance

Rechnen Sie, wenn Sie `veto files` verwenden, mit einem gewissen Performance-Verlust, da Samba für jede Datei und jedes Verzeichnis prüfen muss, ob die Veto-Liste greift.

»Verbotene« Verzeichnisse

Etwas Ähnliches wie »`veto files`« für Dateien bewirkt »`dont descend`« für Verzeichnisse. Verzeichnisse, die in der »`dont descend`«-Liste auftauchen, werden immer als leer dargestellt (`veto files` würde ihre Existenz komplett verleugnen).

Die Manpage `smb.conf`(5) weist darauf hin, dass Samba sich beim Format der »`dont descend`«-Einträge sehr kleinlich anstellen kann, und empfiehlt Experimentieren. Statt `/proc` müssen Sie zum Beispiel möglicherweise `./proc` angeben.

Übungen

7.6 [!2] Exportieren Sie eine schreibbare Freigabe und prüfen Sie Schreibzugriffe. Mit welcher Unix-UID werden die Zugriffe ausgeführt? Verändern Sie die Unix-Zugriffsrechte der Dateien auf der Freigabe so, dass nur noch Schreib- bzw. gar keine Zugriffe mehr möglich sind.

7.7 [2] Verstecken Sie alle Dateien auf der Freigabe, deren Namen auf ».txt« enden.

7.4 »Authentisierung« auf Freigabeebene

Im Einleitungskapitel haben Sie gelernt, wie Sie eine Freigabe erstellen können, auf die jeder auch ohne Benutzername und Kennwort zugreifen darf. Im letzten Abschnitt haben Sie gelernt, Freigabenzugriffe auf bestimmte Rechner im Netzwerk einzugrenzen und Freigabenzugriffe für alle einzuschränken. Eine Authentisierung war dafür nicht notwendig; alle Zugriffe fanden mit den Rechten des Unix-Benutzers `nobody` statt.

Anonyme Freigaben mit Kennwort

Dieser Abschnitt beschreibt, wie Sie solche anonyme Freigaben mit einem Kennwort versehen können, ohne dabei eine benutzerspezifische Authentisierung durchzuführen. Wie Sie schon in Abschnitt 5.5 erfahren haben, steht hierfür die SMB-Sicherheitsstufe »`share`« zur Verfügung. Auch in den bisherigen Beispielen war diese Sicherheitsstufe eingestellt, allerdings hatten wir noch kein Kennwort für die Freigabe vergeben.

`share`

Clients und Kennwörter

Normalerweise überträgt jeder Windows-Client ein Kennwort, nämlich das mit dem sich der Windows-Benutzer am Client angemeldet hat. Das entsprechende Kennwort wird im Fall der bisherigen Beispielskonfigurationen auch durchaus vom Samba-Server empfangen und zur Authentisierung herangezogen – das schlug bisher zwar mangels geeigneter Konfiguration immer fehl, aber dann griff »`guest ok`« und der Benutzer konnte auch ohne gültiges Kennwort zugreifen.

Freigaben vs. Kennwörter

Wenn Sie möchten, dass für eine Freigabe ein bestimmtes Kennwort gilt, müßten Sie einen SMB-Benutzer mit diesem Kennwort anlegen und Samba dazu zwingen, nur noch Zugriffe mit diesem Kennwort zu akzeptieren. Dazu müßten Sie, wie im nächsten Abschnitt beschrieben, einen Unix-Benutzer (etwa `freigabe`) sowie einen korrespondierenden SMB-Benutzer anlegen. Die Einträge `username = freigabe` und `only user = yes` würden dann Samba zwingen, nur noch das neue Kennwort zu akzeptieren.

Probleme mit modernen Clients

Die beschriebene Methode hat den Haken, dass moderne Windows-Clients standardmäßig von einer benutzerspezifischen Authentisierung ausgehen und entweder das Kennwort übertragen, mit dem sich der Benutzer auf dem Client angemeldet hat, oder nach einer anderen Benutzer/Kennwort-Kombination fragen. Aus Sicht des Komforts müssten Sie alle Benutzer als SMB-Benutzer anlegen und diese Benutzer in die `username`-Liste aufnehmen. Damit haben Sie im Prinzip schon eine benutzerspezifische Authentisierung, und die lässt sich auch einfacher erreichen.

Fazit: Die Verwendung der Sicherheitsstufe »`security = share`« ist nur in Zusammenhang mit »`guest ok`« bzw. »`guest ok`« wirklich sinnvoll. Eine freigabebasierte »Authentisierung« mit Kennwort funktioniert bei heutigen Windows-Systemen nicht wirklich. Abgesehen davon ist die beschriebene Methode ohnehin relativ sinnfrei. Sollen viele Benutzer mit demselben Kennwort auf eine Freigabe zugreifen, bleibt das Kennwort eh nicht lange geheim.

7.5 »Benutzerorientierte« Authentisierung

Wenn es darum geht, Zugriffe auf Netzwerkressourcen einzuschränken, ist eine benutzerorientierte Authentisierung erste Wahl. Ist ein Benutzer dem System individuell bekannt, kann gezielt festgelegt werden, was dieser Benutzer mit Verzeichnissen und Dateien tun darf. Während des Aufbaus einer SMB-Verbindung wird mittels der *SMB Negotiate Protocol*-SMBs das zu verwendende Authentisierungsverfahren ausgehandelt.

Zur Erinnerung; SMB kennt verschiedene Authentisierungsverfahren, die über die SMB-Sicherheitsstufen definiert sind. Hier noch einmal eine kurze Zusammenfassung:

Share level (siehe voriger Abschnitt) Zugriffsbeschränkungen sind rein freigabeorientiert – wie gesagt handelt es sich hier um keine benutzerorientierte Authentisierung.

User level Die Benutzer authentisieren sich für den Zugriff auf eine Freigabe gegenüber einer lokal auf dem Server mit der Freigabe liegenden Datenbank.

Server level Wie »User level«, aber die Authentisierungsinformationen liegen hier auf einem anderen Rechner als die Freigabe selbst. Wird nur in Zusammenhang mit Arbeitsgruppen (*workgroups*), ergo nur noch selten eingesetzt.

Domain level Wie »User level«, aber die Authentisierungsinformationen liegen hier auf dem »Domänen-Controller« der NT-Domäne – das Verfahren setzt also eine vorhandene Domänenstruktur voraus. Die Benutzer müssen sich nur einmal gegenüber der Domäne anmelden und sich anschließend nicht mehr für jede Freigabe im Netz einzeln authentisieren.

ADS level Entspricht im Wesentlichen dem »Domain level«, außer dass die Authentisierungsinformationen nicht auf einem NT-Domänen-Controller, sondern verteilt im *Active Directory* liegen.

Ein Samba-Server kann jede der Methoden zur Authentisierung einsetzen:

User level Die Benutzer werden lokal auf dem Server in einer Datei (`smbpasswd`) oder Datenbank abgelegt.

Server level Der Samba-Server kann einerseits einen Kennwortserver benutzen und andererseits selbst als Kennwortserver dienen.

Domain level Der Samba-Server kann einerseits einen PDC benutzen und andererseits selbst als PDC dienen, ab Samba 3 ohne Einschränkung.

ADS level Der Samba-Server kann mit entsprechendem Kennwort-Backend und LDAP ein *Active Directory* zur Authentisierung heranziehen. Auf Windows-Deutsch heißt das, er kann *Member Server* eines AD werden. Mehr aber auch nicht; für ein ADS auf Samba-Basis müssen wir Sie auf Samba 4 vertrösten.

In den folgenden Abschnitten wedern Sie lernen, wie man die verschiedenen Sicherheitsstufen mit Samba umsetzt. Dem Thema »Samba und ADS« wird später ein eigenes Kapitel gewidmet. Zuvor noch ein paar wichtige Grundlagen zum Thema »Benutzer und Kennwörter«:

SMB-Benutzerkonten

Das Wichtigste: Bei der Anmeldung für eine SMB-Freigabe werden SMB-Benutzerkonten verwendet. Diese liegen entsprechend der Sicherheitsstufe auf dem Rechner mit der Freigabe oder auf einem anderen Rechner. Ist ein Benutzer einmal angemeldet, kann er auf Dateien zugreifen. Auf einem Samba-Server muss dazu ein Unix-Prozess mit entsprechender Unix-UID gestartet werden. Damit ein SMB-Benutzerkonto unter Samba funktioniert, muss es immer einen korrespondierenden Unix-Benutzer geben!

Unix vs. SMB

Leider kann für die Authentisierung der Benutzer unter Samba nicht der übliche Authentisierungsmechanismus von Unix verwendet werden, da SMB-Kennwörter nicht auf Unix-Kennwörter abgebildet werden können. Für Zugriffe auf Dateien ist eine Unix-UID erforderlich, aber die Authentisierung erfolgt über SMB-Kennwörter.

In der Praxis legen Sie also für jeden Benutzer zuerst ein Unix-Konto an und danach einen korrespondierenden SMB-Benutzer. Das Unix-Konto sollte nur dann ein gültiges Kennwort zugewiesen bekommen, wenn der Benutzer sich auch anderweitig, etwa per `ssh`, auf dem Unix-Rechner anmelden können soll – was in der Regel unerwünscht ist.

7.6 Freistehende SMB-Server – die Sicherheitsstufe »*User Level*«

Soll Ihr Samba-Server eine lokale Benutzerdatenbank für die Anmeldung von Benutzern verwenden, müssen Sie im globalen Abschnitt der Datei `smb.conf` als Sicherheitsstufe »`security = user`« eintragen, z. B.

`security = user`

```
[global]
        workgroup = HOGWARTS
        security = user
[test]
        path = /testfreigabe
```

Ab Samba 2.0.5 könnten Sie sich das auch sparen, da der Eintrag seitdem Standard ist.

Benutzer anlegen

Jetzt können Sie lokal die Benutzer anlegen. Jeder SMB-Benutzer benötigt ein entsprechendes Unix-Konto, das zuerst angelegt werden sollte:

```
# useradd -m harry
```

Dann kann ein entsprechender Samba-Benutzer angelegt und gleichzeitig mit einem Kennwort versorgt werden. Das geschieht üblicherweise mit dem Programm `smbpasswd`:

`smbpasswd`

```
# smbpasswd -a harry
New SMB password: N1mbus2k
Retype new SMB password: N1mbus2k
```

Die Option `-a` (für *add*) fügt den angegebenen Benutzer in die spezielle Samba-Benutzer-Datenbank hinzu. Dabei handelt es sich in der Regel um die Datei `smbpasswd` im Samba-Konfigurationsverzeichnis, die bei den aktuellen Sambaversionen so aussieht:

`smbpasswd`

```
# cat /etc/samba/smbpasswd
nobody:65534:XXXX<◁◁◁◁XXXX:XXXX<◁◁◁◁XXXX:[DU         ]:LCT-00000000:
harry:1005:3DC6<◁◁◁◁04EE:243B<◁◁◁◁B051:[U          ]:LCT-41F8BC97:
```

Die Datei enthält einen Benutzer pro Zeile. Jedem Benutzer sind verschiedene Werte in Form von Feldern zugeordnet, die ihrerseits durch Doppelpunkte voneinander getrennt sind. Die ersten beiden Felder enthalten Name und UID des Benutzers, dabei *muss* es sich um die Unix-Werte handeln! Das dritte und das vierte Feld enthalten die Kennwort-Hash-Werte (DES für die älteren Windows-Versionen 95, 98 und ME; MD4 für Windows NT und seine Nachfahren).

Wer die Kennwort-Hash-Werte kennt, kann sich erfolgreich authentisieren. Der Dateiinhalt darf daher nur für `root` lesbar sein!

`passdb backend`

Als Ablage für die SMB-Benutzerdaten können verschiedene Backends ausgewählt werden. Zu diesem Zweck existiert der Parameter »`passdb backend`«. Sein Standardwert ist »`smbpasswd`«, Sie können aber beispielsweise auch einen LDAP-Server als Ablage verwenden:

```
passdb backend = ldapsam:ldap://ldap.example.com
```

Der Wert legt zuerst den Typ des Backends fest und spezifiziert (nach einem Doppelpunkt) das Backend dann genauer. Bei dem Wert »`smbpasswd`« könnten Sie zum Beispiel einen bestimmten Pfad angeben:

```
passdb backend = smbpasswd:/etc/samba/smbpasswd
```

Das `smbpasswd`-Backend ist für kleine Installationen mit höchstens ein paar Dutzend Benutzern sinnvoll; geht die Anzahl in die Hunderte, kann die `smbpasswd`-Datei nicht mehr schnell genug durchsucht werden. Ein anderes interessantes Backend für kleine bis mittlere Installationen ist `tdbsam`, das die Benutzerdaten in einer einfachen Datenbank unterbringt (`tdb` steht für *trivial database*). Dies skaliert bis zu einigen hundert Benutzern; für noch größere Benutzerzahlen ist die Verwendung von `ldapsam` in Verbindung mit einem LDAP-basierten Verzeichnisdienst (etwa OpenLDAP) dringend zu empfehlen.

`tdbsam`

Wie schon in Abschnitt 5.5 erwähnt, sollten SMB-Kennwörter (bzw. deren Hash-Werte) auf jeden Fall in verschlüsselter Form im Netzwerk übertragen werden. Das ist bei den aktuellen Samba-Versionen (ab Samba 3.0) standardmäßig der Fall. Sie können die Kennwortverschlüsselung jedoch mit dem Parameter

Kennwortverschlüsselung

```
encrypt passwords = yes
```

jederzeit explizit festlegen.

Das fünfte Feld enthält diverse Parameter für das Benutzerkonto; `D` bedeutet beispielsweise, dass das Konto deaktiviert ist. Im Beispiel hat der entsprechende Benutzer (`nobody`) ohnehin kein gültiges Kennwort (`XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX`). Das sechte Feld schließlich enthält das Datum der letzen Kennwortänderung (*Last Change Time*; hexadezimale Unix-Zeit).

Die Datei wird üblicherweise nicht direkt, sondern eben mit dem Kommando `smbpasswd` bearbeitet. Achtung: Das Programm verwendet, wenn es von nicht privilegierten Benutzern aufgerufen wird, das Loopback-Interface, um lokal auf den `smbd` zuzugreifen. Dieser sollte daher zumindest lokal erreichbar sein (Parameter `hosts deny`).

Nicht privilegierte Benutzer können mit dem Programm unter anderem ihr SMB-Kennwort lokal, und was vielleicht interessanter ist, auch auf entfernten SMB-Maschinen (also auch auf Windows-Rechnern) ändern:

Kennwort ändern

```
$ smbpasswd -r 192.168.0.1 -U harry
Old SMB password: N1mbus2k
New SMB password: F1reb0lt
Retype new SMB password: F1reb0lt
Password changed for user harry on 192.168.0.1.
```

`root` kann jederzeit die Kennwörter der lokalen SMB-Benutzer ändern:

```
$ smbpasswd harry
New SMB password: secret123
Retype new SMB password: secret123
```

Mit der Option `-d` (engl. *disable*) kann er Benutzer sperren und mit `-e` (engl. *enable*) wieder entsperren. Die Option `-x` schließlich ermöglicht es, Benutzer ganz zu löschen.

Jetzt wissen Sie, wie Sie SMB-Benutzerkonten anlegen und dass hinter jedem SMB-Benutzer ein Unix-Benutzer steckt. In bestehenden Umgebungen kann es aber vorkommen, dass ein bestehender SMB-Benutzer unter Unix einen ganz anderen Namen hat. Normalerweise müssen beide Namen gleich sein; netterweise haben Sie aber die Möglichkeit, zusätzlich Aliasnamen zu vergeben. Der entsprechende Parameter lautet »`username map`«. Hiermit können Sie den Pfad zu einer Datei angeben, in der die Aliasnamen stehen:

Benutzer-Aliasnamen
`username map`

```
username map = /etc/samba/users.map
```

Der Inhalt der Datei könnte etwa so aussehen:

```
smbtest = tux
root = Administrator admin
harry = "Harry Potter"
```

Im Beispiel kann sich ein Windows-Benutzer als `tux` anmelden, während er auf der Unix-Seite als `smbtest` geführt wird. Häufig wird die Datei auch benutzt, um Windows-Administratoren zu »`root`« oder Windows-Benutzer mit Leerzeichen im Namen zu gültigen Unix-Benutzern zu machen.

Eine letzte Frage zur Authentisierung: Was passiert eigentlich mit Anmeldeversuchen, die ein falsches Kennwort oder gar einen falschen Benutzernamen enthalten? Klar, sie werden abgewiesen. Wie bei einigen Windows-Systemen der Fall, kann man unter Samba solchen »Benutzern« aber auch einen Gastzugriff erlauben. Dafür gibt es den Parameter »`map to guest`«. Erlaubt sind folgende Werte:

`map to guest`

`map to guest = Never` Das ist der Standard: Falsche Angaben – kein Zugriff.

`map to guest = Bad User` Wenn ein gültiger Benutzername mit einem ungültigen Kennwort angegeben wird, wird der Zugriff abgewiesen – bei einem ungültigen Benutzernamen wird dagegen der Zugriff als »Gast« erlaubt. Dieser Ansatz ist etwas fragwürdig, da er Angreifern potentiell ermöglicht, gültige Benutzernamen zu raten.

`map to guest = Bad Password` Hier werden gültige Benutzernamen mit ungültigem Kennwort als Gastzugriff interpretiert. Ihre Benutzer werden sich über diese Konfiguration vermutlich ärgern, da sie bei einem Tippfehler im Kennwort ohne weiteren Kommentar als »Gast« angemeldet werden (mit völlig anderen Rechten, als ihnen sonst gebühren würden).

`map to guest = Bad Uid` Wenn Samba Domänen- oder ADS-Security verwendet, bedeutet diese Einstellung, dass authentisierte Benutzer, für die auf der Samba-Seite kein Benutzerkonto existiert (und für die Samba auch keins automatisch anlegen kann), als »Gast« angemeldet werden. (Diese Einstellung gibt es nur bei Samba 3; Samba 2 macht das automatisch so.)

Sie sollten allerdings mit dem Parameter `guest account` kontrollieren, welcher Unix-Benutzer hinter dem Gastbenutzer steht. Und: Benutzer, die sich explizit als Gast anmelden, sind von `map to guest` nicht betroffen. Für diese Benutzer sollte allerdings ein `guest ok = yes` existieren …

»`map to guest`« ist die einzige Möglichkeit, bei Sicherheitsebenen außer `share` einen »Gastzugriff« zu erlauben. Das liegt daran, dass auf diesen Ebenen die Authentisierung stattfindet, *bevor* der Server mitgeteilt bekommt, welche Ressource überhaupt angesprochen werden soll. Die Gastauthentisierung kann also nicht im Rahmen der eigentlichen Authentisierung stattfinden.

Übungen

7.8 [!3] Konfigurieren Sie Ihren Samba-Server so, dass er nur noch authentisierte Benutzer auf Freigaben zugreifen lässt. Legen Sie dazu einen entsprechenden SMB-Benutzer an und testen Sie Ihre Konfiguration. Verwenden Sie dazu, wenn möglich, einen Windows-Rechner. Bei den gängigen Windows-Systemen müssen Sie dabei in der »Netzwerkumgebung« unter »Extras« ein »Netzlaufwerk verbinden«. Sorgen Sie dafür, dass dabei die korrekten Benutzer-Informationen übermittelt werden.

7.9 [!3] Konfigurieren Sie Ihren Server so, dass kein Benutzer Kennwörter haben darf, die kürzer als sieben Zeichen sind.

7.10 [2] Sorgen Sie dafür, dass auf Ihrem Samba-Server ein Benutzer namens `Administrator` existiert. Von seinen Rechten her soll er dem Unix-Benutzer `root` entsprechen. Diskutieren Sie im Kurs, ob das eine sinnvolle Konfiguration ist.

7.11 [4] Nehmen wir einmal an, Sie hätten schon eine ganze Reihe Benutzer als Unix-Benutzer angelegt. Nun möchten Sie aus allen Nicht-Systembenutzern (UID größer als 499) auch SMB-Benutzer machen. Schreiben Sie ein Shellskript, das diese Aufgabe bewältigt und zusätzlich ein Standard-Kennwort an jeden Benutzer vergibt (z. B. »password«).

7.7 Zugriffsregeln und -rechte

Im letzten Abschnitt haben Sie gelernt, wie Sie unter Samba eine benutzerspezifische Authentisierung einrichten können. Viel bringt die beschriebene Konfiguration aber noch nicht, denn Authentisierung ist zwar gut und schön, bringt aber ohne Zugriffsregeln wenig. Selbstverständlich könnenn Sie die Unix-Rechte verwenden. Mit diesen sowie den Windows-Zugriffsrechten und ACLs werden wir uns später noch beschäftigen. Zuallererst mal ein paar allgemeinere Einstellungen in der Datei `smb.conf`.

Durch verschiedene freigabespezifische Parameter haben Sie die Möglichkeit, einzelnen Benutzer den Zugang zu bestimmten Freigaben zu verweigern bzw. zu erlauben. Hier eine Übersicht über die relevanten Parameter.

`valid users` Dieser Parameter legt fest, welche (Unix-)Benutzer überhaupt auf die Freigabe zugreifen dürfen. Wer nicht in dieser Liste steht, bekommt keinen Zugriff.

`invalid users` Die hier genannten Benutzer bekommen unter keinen Umständen Zugriff auf die Freigabe (egal was in den anderen Parametern steht).

`write list` Dieser Parameter gibt an, welche Benutzer auf die Freigabe schreiben dürfen – egal welchen Wert die `read only`- bzw. `writeable`-Einstellung für die Freigabe hat.

`read list` Dieser Parameter gibt an, welche Benutzer die Freigabe auf jeden Fall nur lesen dürfen – egal welchen Wert die `read only`- bzw. `writeable`-Einstellung für die Freigabe hat.

`admin users` Die hier genannten Benutzer bekommen vollen Zugriff (à la Benutzer `root`) auf die Freigabe. Achtung! Äußerst riskant, da sämtliche Unix-Zugriffsrechte außer Kraft gesetzt werden.

Syntax

Alle diese Parameter akzeptieren dieselbe Syntax für die Benutzernamen. Sie müssen immer Unix-Benutzernamen angeben, und Sie können mehrere Benutzernamen aufzählen, die dann durch Leerzeichen, Tabs oder Kommas voneinander getrennt werden müssen.

```
valid users = harry
valid users = harry, ron, hermione
```

Gruppen

Außerdem können Sie (außer bei »`admin users`«) auch die Namen von Unix-Gruppen angeben, wenn Sie ein »`@`« davorsetzen:

```
write list = @gryffindors
```

Eigentlich ist es mit den Gruppen etwas komplizierter: Das Zeichen »+« sorgt dafür, dass der Name als Unix-Gruppenname angesehen wird. »&« interpretiert den Namen als NIS-*netgroup*-Name (sofern Ihr System NIS verwendet). Die beiden Zeichen können gemeinsam verwendet werden, wobei

die Reihenfolge der Zeichen die Suchreihenfolge angibt. »@« ist äquivalent zu »&+«.

Häufig ist es sinnvoll, zuerst mit allgemeinen Zugriffs-Parametern einige oder sogar alle Benutzer auszusperren und dann nachträglich einige wenige zuzulassen. Typischerweise könnte eine solche Konfiguration wie folgt aussehen:

```
<<<<<
valid users = root
read list = nobody, gast
write list = harry
admin users = mcgonagall
<<<<<
```

Mit `valid users = root` ist `root` erst einmal der Einzige der potentiell Zugriff auf die entsprechende Freigabe hat. Dabei ist nicht einmal entscheidend, ob der Benutzer existiert, wichtig ist nur, dass die Liste nicht leer ist. Sonst hätte nämlich nach wie vor jeder Zugriff! Anschließend erlauben wir `nobody` und `gast` das Lesen und `harry` das Schreiben. `mcgonagall` gilt als Administrator.

Nach diesem Vorgeplänkel jetzt aber zu den eigentlichen Zugriffsrechten. Erst einmal: Jeder SMB-Benutzer greift mit einer Unix-UID auf Dateien zu, und selbstverständlich gelten hier die Unix-Rechte. Aber ganz so einfach ist es leider nicht. Durch die Interaktion der Windows- mit der Unix-Seite treten einige Probleme auf. Viele Aktionen benötigen unter Windows andere Rechte-Konstellationen als unter Unix.

Rechte ändern

Unter anderem sind es Windows-Benutzer gewohnt, dass sie die Rechte jeder Datei verändern dürfen, sofern sie Schreibrecht dafür haben. Wie Ihnen bekannt sein dürfte, ist das unter Unix anders; hier darf nur der Eigentümer einer Datei deren Rechte ändern. Für das entsprechende Windows-Verhalten reicht ein

```
dos filemode = yes
```

im entsprechenden Freigabeabschnitt der Datei `smb.conf`.

Dateien löschen

Normalerweise erlaubt Windows Ihnen nicht, Dateien zu löschen, die Sie nur lesen dürfen. Unter Unix dagegen sind die Rechte der Datei bekanntlich für das Löschen irrelevant (es zählen nur die Rechte auf dem Verzeichnis, in dem die Datei steht). Samba setzt normalerweise das Windows-Verhalten durch. Wenn Sie möchten, dass Windows-Benutzer auf Freigaben schreibgeschützte Dateien löschen können, verhilft ein

```
delete readonly = yes
```

dem Samba-Server zum gewohnten Unix-Verhalten.

Dieser Parameter ist nützlich in manchen Fällen, wo die Unix-Seite verhindert, dass Benutzer die Rechte für Dateien ändern, die ihnen nicht gehören, während die Windows-Seite verhindert, dass Benutzer schreibgeschützte Dateien löschen. Eine Anwendung, die das betrifft, ist das (etwas verstaubte) Quellcodemanagementsystem RCS.

neue Dateien

Außerdem legen Windows-Benutzer neue Dateien und Verzeichnisse an. Leider kennt der Windows-Client keine Unix-Rechte, kann also auch keine setzen. Daher muss sich der Samba-Server um die passenden Rechte der neuen Dateien kümmern. Sie können die Standardrechte mit den Parametern »`create mask`« (für neue Dateien) und »`directory mask`« (für neue Verzeichnisse) beeinflussen. Beide Parameter haben ein Argument, das eine Unix-übliche oktale Rechtespezifikation darstellt; die Standardwerte sind

```
create mask = 0744
directory mask = 0755
```

Die genaue Vorgehensweise ist wie folgt: Aus den Wünschen des Windows-Clients werden möglichst passende Unix-Rechte konstruiert. Anschließend werden diese Rechte mit dem Wert der »create mask« (bzw. »directory mask«) binär UND-verknüpft. Das heißt, alle Bits, die in der »create mask« *nicht* gesetzt sind, werden aus den vom Windows-Client gewünschten Dateirechten entfernt.

Verwechseln Sie diese Parameter nicht mit der Unix-umask. In der umask müssen Sie genau die Bits *setzen*, die eine neue Datei *nicht* haben soll. Ein umask-Wert ist also das binäre Komplement eines gleichbedeutenden Samba-»create mask«-Werts. (Außerdem gilt die umask für Dateien *und* Verzeichnisse.)

Das ist noch nicht die ganze Geschichte: Samba unterstützt außerdem zwei Parameter namens »force create mode« und »force directory mode«, die mit »create mode« und »directory mode« korrespondieren. Nachdem die Wunsch-Rechte einer neuen Datei mit der »create mask« binär UND-verknüpft wurden, werden sie mit dem »force create mode« binär ODER-verknüpft. Damit können Sie dafür sorgen, dass neue Dateien bestimmte Rechte immer gesetzt haben. Dasselbe gilt für neue Verzeichnisse mit der »force directory mask«.

Die eben beschriebenen Parameter betreffen nur neu angelegte Dateien, aber beeinflussen nicht die Dateirechte, die Sie im Windows-»Sicherheit«-Dialog manipulieren können. Hierfür gibt es die Parameter »security mask« und »force security mode«. Die auf der Windows-Seite eingestellten Rechte werden mit der »security mask« binär UND-verknüpft bzw. mit dem »force security mode« binär ODER-verknüpft. Normalerweise stehen »security mask« auf 0777 und »force security mode« auf 0, so dass den Benutzern keine Restriktionen auferlegt werden. Verwenden Sie diese Parameter mit Vorbedacht, da ein beträchtliches Verwirrungspotential existiert. – Die Parameter »directory security mask« und »force directory security mode« wirken entsprechend für Verzeichnisse.

Alternativ zu »create mask« und »directory mask« können Sie die Rechte für neu erstellte Dateien oder Verzeichnisse auch mit »inherit permissions« festlegen. Ist für eine Freigabe

```
inherit permissions = yes
```

gesetzt, spielen »create mask«, »directory mask« & Co. *keine* Rolle mehr; für neue Verzeichnisse werden alle Rechte vom Elternverzeichnis übernommen, für neue reguläre Dateien nur Schreib- und Leserecht. Das Set-UID-Bit wird nie übernommen.

Windows-Attribute

Ein weiteres Problem: Unter Windows gibt es die speziellen Attribute Schreibgeschützt, Versteckt, Archiv und System. Das erste läßt sich leicht auf die Unix-Dateirechte abbilden; die letzten drei haben unter Unix kein Äquivalent, aber der Samba-Server kann sie trotzdem speichern. Dafür werden die Unix-Ausführrechte herangezogen, die unter Windows sowieso keine Bedeutung haben:

map hidden Der Wert yes bewirkt, dass das Versteckt-Attribut auf das Unix-Ausführrecht für den »Rest der Welt« abgebildet wird.

map archive Hier bewirkt der Wert yes, dass das Archiv-Attribut auf das Unix-Ausführrecht für den Dateieigentümer abgebildet wird. Windows setzt dieses Attribut, wenn eine Datei seit dem letzten Backup geändert wurde. Der Standardwert für diese Einstellung ist yes, aber es gibt gute Gründe, sie auf no zu setzen – vor allem wenn Sie nicht wollen, dass jede Datei, die ein Windows-Client anfasst, auf der Unix-Seite ausführbar wird (was bei Quellcode-Dateien oder Dokumenten ärgerlich sein könnte).

map system Der Wert yes bewirkt, dass das Versteckt-Attribut auf das Unix-Ausführrecht für die Gruppe abgebildet wird (Standardwert ist no).

Achten Sie darauf, dass die Unix-Ausführbits, die zur Darstellung der Windows-Attribute verwendet werden, nach wie vor der Modifikation durch Parameter wie »create mask« und »force create mode« unterliegen. Die Einstellungen

```
map archive = yes
force create mode = 100
```

setzen aus der Sicht eines Windows-Clients also für alle neuen Dateien das Archiv-Attribut.

Übungen

7.12 [2] Setzen Sie die Unix-Zugriffsrechte einer Testdatei auf einer Freigabe so, dass Ihr Windows-Benutzer alles darf, nur lesen darf bzw. gar nichts darf. Testen Sie Ihre Einstellungen.

7.13 [!3] Wovon hängt es ab, ob ein Benutzer Dateien auf der Freigabe löschen, anlegen oder umbenennen darf? Diskutieren und testen Sie Ihre Hypothesen.

7.14 [!2] Erlauben Sie auf Ihrer Testfreigabe nur Schreibzugriffe des Benutzers harry (oder eines anderen sich anbietenden Benutzers). Schreibzugriffe aller anderen Benutzer sollen abgewiesen werden. Allerdings sollen alle Benutzer lesen dürfen.

7.15 [3] Setzen Sie für eine Datei das Windows-Archiv-Attribut. Was für eine Bedeutung hat es für Windows? Gibt es etwas Vergleichbares unter Linux/Unix?

7.8 Access Control Lists

Neben den herkömmlichen Zugriffsrechten kennen sowohl die meisten Unix-Systeme als auch die Windows-Systeme der NT-Familie sogenannte *Access Control Lists* (ACLs). ACLs ermöglichen es nicht nur, allgemeine Rechte zu setzen, sondern für eine Vielzahl von Benutzern individuell Rechte zu verteilen.

POSIX-ACLs

Eigentlich gibt es einen Standard für ACLs, die POSIX-ACLs. Leider entsprechen die Windows-ACLs nicht diesem Standard – genauer gesagt gehen sie darüber hinaus. Grundsätzlich ermöglicht Samba es einem Windows-Benutzer, ACL-Einträge zu setzen. Allerdings werden nur diejenigen ACL-Einträge auch gespeichert, die Unix unterstützt; werden auf der Unix-Seite keine ACLs unterstützt, können also nur Einträge gespeichert werden, die den Standard-Unix-Rechten entsprechen. Auf den meisten proprietären Unix-Systemen wie auch auf neueren Linux-Systeme stehen aber POSIX-ACLs zur Verfügung.

Selbst wenn ACLs vom Kernel grundsätzlich unterstützt werden, müssen Sie sie noch für das entsprechende Dateisystem aktivieren. Das können Sie am bequemsten mit der mount-Option acl in der Datei /etc/fstab erledigen:

```
/dev/hda2  /testfreigabe  ext3  acl 1 1
```

Im laufenden System können Sie die Änderungen danach mit einem

```
# mount -o remount /dev/hda2
```

wirksam werden lassen.

ACL-Bearbeitung

POSIX-ACLs können mit den Werkzeugen `getfacl` und `setfacl` ausgelesen und bearbeitet werden:

```
$ ls -l test
-rwxr--r--  1 test users 0 Jan 28 10:04 test
$ getfacl test
# file: test
# owner: test
# group: users
user::rwx
group::r--
other::r--

$ setfacl -m u:tux:rw test
$ getfacl test
# file: test
# owner: test
# group: users
user::rwx
user:tux:rw-
group::r--
mask::rw-
other::r--

$ ls -l test
-rwxrw-r--+ 1 test users 0 Jan 28 10:04 test
```

Beim ersten Auslesen der ACL wurden, zwar in ungewohnter Darstellung, lediglich die Standard-Unix-Rechte angezeigt. Ein Ändern bestehender ACLs erfolgt immer mit der `setfacl`-Option `-m` (engl. *modify*). Mit `u:tux:rw` geben Sie an, dass für einen Benutzer (*user*) die ACL geändert werden soll, nämlich für `tux`; dieser bekommt abweichend die Rechte zum Lesen und Schreiben. Automatisch wird dabei auch eine ACL-Maske gesetzt (*mask*), die die maximalen Rechte vorgibt, die ein nicht privilegierter Benutzer für die betreffende Datei haben kann. Es handelt sich im Prinzip um eine letztgültige Regel (*policy*), mit der sich ein Administrator gegen Konfigurationsfehler absichern kann.

ACL-Warnung

`ls` kann die ACLs zwar nicht anzeigen, deutet aber durch ein Plus-Zeichen hinter den Unix-Rechten an, dass es da noch mehr zu beachten gibt. Außerdem zeigt `ls` an Stelle der Gruppenrechte die ACL-Maske an!

Zugriff

Beim Zugriff auf eine Datei wird zuerst die ACL nach dem zugreifenden Benutzer durchsucht – bei einem Treffer werden die entsprechenden Rechte angewendet. Gibt es keinen ACL-Eintrag für den Benutzer, kommen die Rechte seiner Gruppe zum Zug. Sind auch für seine Gruppe keine Rechte gesetzt, gelten die Rechte für den Rest der Welt. Die Rechte des Dateieigentümers werden von den ACLs nicht beeinflusst. Weitere Informationen zu POSIX-ACLs vermittelt übrigens die Handbuchseite `acl`(5).

ACLs: POSIX vs. Windows

Sind die POSIX-ACLs auf der Unix-Seite aktiviert, können von der Windows-Seite für beliebige Benutzer Einträge in der ACL der Datei gemacht werden – und zwar bezüglich des Lesens, Schreibens und Ausführens der Datei. Zusätzliche Berechtigungen, wie Windows sie kennt, z. B. das Recht »Erweiterte Rechte lesen« zu können, versucht Samba sinnvoll umzusetzen. Falls das nicht möglich ist, tut das Windows-GUI zwar, als ob die Rechte gespeichert würden, was in Wirklichkeit aber nicht der Fall ist. Ein Fehlermeldung erfolgt nicht!

Zu guter Letzt: Mit dem Parameter `nt acl support = no` in der `smb.conf` können Sie verhindern, dass von Windows-Seite überhaupt Rechte auf der Freigabe verändert werden können.

Übungen

7.16 [3] Aktivieren Sie (wenn notwendig) die ACLs für das Dateisystem Ihrer Freigabe und probieren Sie, welche Windows-ACL-Einträge gespeichert werden.

7.9 Verwendung existierender Kennwort-Server

In Abschnitt 7.5 haben Sie gelernt, wie Sie auf einem Samba-Server lokal Benutzer anlegen und für die Authentisierung verwenden können. Diese Vorgehensweise entspricht den Standard-Einstellungen von Samba, namentlich der Windows-Sicherheitsstufe »User«. In Netzwerken ist es allerdings recht lästig, auf jedem Datei-Server separat Benutzer verwalten zu müssen. SMB unterstützt deshalb die Verwendung zentraler Kennwort-Server. Das Konzept kam insbesondere in Zusammenhang mit den Windows-Arbeitsgruppen (*workgroups*) zum Einsatz. In der Praxis wird heute in der Regel das Domänenkonzept eingesetzt. Dieser Abschnitt existiert darum eher der Vollständigkeit halber …

Arbeitsgruppen

Die Verwendung eines existierenden Kennwort-Servers zur Authentisierung der Sambabenutzer ist denkbar einfach: Zuerst sollten Sie in der Datei `smb.conf` im globalen Abschnitt die Windows-Sicherheitsstufe auf »server« setzen:

```
security = server
```

Das alleine kann selbstverständlich nicht ausreichen, da Samba außerdem noch wissen muss, welcher Rechner der Kennwort-Server ist. Das geschieht mit

```
password server = 192.168.0.1                    über die IP-Adresse
password server = pwdsrv.example.com             über den Namen
```

Namen werden dabei gemäß der »`name resolve order`« (S. 86) aufgelöst.

Aus der Sicht der Clients gibt es keinen Unterschied zwischen »`security = user`« und »`security = server`«; sie authentisieren sich in jedem Fall gegen den Samba-Server, dessen Freigabe sie einbinden wollen. Das bedeutet insbesondere, dass auf dem Samba-Server die benötigten Unix-Benutzerkonten zur Verfügung stehen müssen.

Bedenken Sie, dass unter »`security = server`« die Sicherheit Ihres Samba-Servers von der Sicherheit des Kennwort-Servers abhängt. Sie sollten keinen Kennwort-Server verwenden, dem Sie nicht zu 100% vertrauen.

Verkneifen Sie es sich, einen Samba-Server auf sich selbst als Kennwort-Server zeigen zu lassen, da dies eine Endlosschleife produziert und Ihr Samba-Server sich verklemmt.

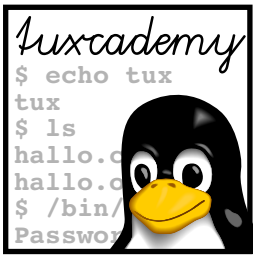
»`security = server`« entspricht, wenn man so will, einer Art legalisiertem *Man-in-the-middle*-Angriff. Aus diesem Grund gibt es oft Probleme, wenn der Kennwort-Server abstürzt; der Samba-Server bekäme in diesem Fall einen neuen Sitzungsschlüssel, zu dem die verschlüsselten Kennwörter der Clients dann nicht mehr passen. Aus diesem Grund gibt Samba in dieser Situation notgedrungen auf – ein ziemlich unakzeptables Verhalten. Unter anderem aus diesem Grund wird von der Verwendung von »`security = server`« dringend abgeraten.

Kommandos in diesem Kapitel

getfacl	Zeigt ACL-Informationen an	getfacl(1)	108
setfacl	Erlaubt die Manipulation von ACLs	setfacl(1)	108
smbpasswd	Ändert das Samba-Kennwort eines Benutzers	smbpasswd(1)	102

Zusammenfassung

- Zugriffsrechte auf Samba-Server ergeben sich aus den Zugriffsrechten des Windows- und des Linux/Unix-Systems.
- Samba-Benutzer müssen in der Regel auch als Unix-Benutzer angelegt werden.
- Es gibt diverse Möglichkeiten zur Steuerung der Abbildung von Windows- auf Unix-Dateizugriffsrechte.
- Die speziellen Windows-Dateiattribute `Versteckt`, `Archiv` und `System` können auf die unter Windows unbenutzten Ausführrechte einer Datei abgebildet werden.
- Windows-ACLs werden unterstützt, wenn auf dem Samba-Server POSIX-ACLs zur Verfügung stehen.



8

Drucken mit Samba

Inhalt

Lernziele

- Verstehen, wie Samba das Problem »Drucken« adressiert
- Von Linux-Clients aus auf SMB-Druckerfreigaben drucken können
- Drucker auf einem Samba-Server über Berkeley-LPD oder CUPS für Linux- und Windows-Clients freigeben
- Druckertreiber-Installation auf Windows-Clients mit Samba realisieren können

Vorkenntnisse

- Kenntnisse über die Konfiguration von Samba
- Kenntnisse über die Linux-Drucksysteme Berkeley-LPD und CUPS
- Kenntnisse über die Konfiguration von Druckern auf Windows-Systemen sind hilfreich

8.1 Einleitung

Grundprinzipien

Drucker sind wie Plattenplatz eine Ressource, die sinnvoll von mehreren Benutzern und Rechnern aus gemeinsam verwendet wird. Unter Linux stehen mit den Drucksystemen Berkeley-LPD und CUPS zwei verbreitete und leistungsfähige Lösungen zur Verfügung, die die gemeinsame Nutzung von Druckern ermöglichen, und viele Drucker können heute auch direkt an ein Ethernet angeschlossen werden und so mehreren Rechnern zur Verfügung stehen. Ebenso bietet Windows Möglichkeiten zur gemeinsamen Nutzung von Druckern über Freigaben.

In diesem Kapitel besprechen wir die Integration von Windows- und Linux-Systemen für die Zwecke der Druckernutzung. Zunächst behandeln wir, wie Unix- bzw. Linux-Rechner SMB-Druckerfreigaben ansprechen können; anschließend diskutieren wir die Möglichkeiten dafür, mit einem Samba-Server genau solche Freigaben anbieten und damit Drucker verfügbar machen zu können, die in einer Linux-Umgebung per LPD oder CUPS zur Verfügung stehen.

Drucken: Samba vs. Windows

Grundsätzlich kann ein Rechner mit Samba sämtliche Druckfunktionen eines Windows-Servers übernehmen. Dazu zählt nicht nur die Weiterleitung von Druckaufträgen an Drucker, sondern auch die automatische Installation von Druckertreibern auf Windows-Clients oder das Hinzufügen neuer Treiber für Clients von der Windows-Seite aus.

Samba enthält keine eigenen Druckfunktionen, sondern verläßt sich für die eigentliche Arbeit auf das auf dem Linux-Rechner installierte Drucksystem (LPD oder CUPS). Die Samba-Software kümmert sich lediglich um die Weiterleitung von Druckaufträgen an das lokale Drucksystem.

Dieses Kapitel setzt voraus, dass Sie sich mit den Linux-Drucksystemen Berkeley-LPD und CUPS auskennen, etwa auf dem für die LPIC-1-Zertifizierung erforderlichen Stand. Näheres hierzu finden Sie zum Beispiel in der Linup-Front-Schulungsunterlage *Systemadministration I – System und Benutzer*.

8.2 Drucken auf Windows- oder Samba-Freigaben

`smbclient`

Unix- und Linux-Rechner können SMB-Druckerfreigaben, egal ob sie von einem Windows- oder Samba-Server zur Verfügung gestellt werden, über das `smbclient`-Programm ansprechen. Ein Kommando wie

```
$ echo Hallo | smbclient //SERVER/DRUCKER -c print
```

sollte die Zeichenkette `Hallo` auf der Druckerfreigabe `DRUCKER` des Servers `SERVER` ausgeben, jedenfalls auf altmodischen Druckern, die keine aufwändige Vorverarbeitung der Druckdaten benötigen.

LPD-Warteschlange

Im »wirklichen Leben« würden Sie Ihre Benutzer das `smbclient`-Programm natürlich nicht direkt aufrufen lassen, sondern eine LPD-Warteschlange für den Drucker definieren, etwa so:

```
smblj|smb-laserjet:\
    :sd=/var/spool/lpd/smblj:\
    :lp=/dev/lpd/smblj:\
    :if=/usr/bin/smbspool smb://SERVER/DRUCKER 1 X X 1 X:\
    :sh:\
    :mx#0:
```

`smbspool`

Hier verwenden wir das Programm `smbspool`, das Bestandteil des Samba-Pakets ist. Die etwas eigenartigen Aufrufargumente hinter dem Druckernamen (`smb://SERVER/DRUCKER`) sind Platzhalter, die das `smbspool`-Programm verlangt, aber nicht auswertet.

Der Lebenszweck von `smbspool` ist eigentlich, als Backend für CUPS zu fungieren, wenn CUPS Druckaufträge auf SMB-Drucker weiterleiten soll. Hierzu gleich mehr …

Beachten Sie, dass die so installierten Warteschlangen davon ausgehen, dass Sie ihnen Daten in einem Format schicken, das der angeschlossene Drucker direkt verarbeiten kann. Das bedeutet in der Regel, dass Sie PostScript- oder Grafikdaten nicht direkt an die Warteschlange schicken, sondern vorverarbeiten müssen. Das ist explizit über Programme wie GhostScript möglich; die bequemere Möglichkeit involviert automatische Systeme wie `apsfilter` oder `magicfilter`, die versuchen, das Format der Druckdaten zu erkennen und eine selbsttätige Umwandlung vorzunehmen. Mit CUPS gibt es solche Probleme natürlich nicht.

Datenformate

CUPS unterstützt das Drucken auf SMB-Freigaben direkt über das `smb`-Backend (alias `smbspool`). Prüfen Sie das Verzeichnis, in dem Ihre CUPS-Installation ihre Backends ablegt (typischerweise etwas wie `/usr/lib/cups/backend`), ob es dort einen Eintrag für `smb` gibt; wenn nein, legen Sie mit

CUPS

```
# ln -s $(which smbspool) /usr/lib/cups/backend
```

einen an.

Um aus CUPS einen SMB-Drucker anzusprechen, müssen Sie bei der Installation des Druckers nur einen URI wie

CUPS und SMB-Drucker

```
smb://HOGWARTS/SCROLLS/printer
```

angeben, um die Druckerfreigabe `printer` auf dem Server `SCROLLS` in der Domäne `HOGWARTS` anzusprechen. Die Angabe der Domäne ist nur nötig, wenn Ihr Clientsystem sich in einer anderen Domäne befindet.

CUPS wandelt alle Druckdaten wie üblich in PostScript (oder sein internes Rasterformat) um und erzeugt daraus druckerspezifische Daten, die dann ans Backend übergeben werden. Wenn Sie auf Ihrem Rechner den Drucker mit dem richtigen Hersteller und Modell installiert haben, sollte es darum unerheblich sein, ob er lokal angeschlossen oder über eine SMB-Freigabe erreichbar ist.

PostScript

Im Zusammenhang mit SMB ist es meist sinnvoll, einen CUPS-basierten Druckerserver so zu installieren, als böte er ausschließlich »echte« PostScript-Drucker an. Auf Windows-Clients genügt es dann, einen PostScript-Druckertreiber zu installieren, und der Druckerserver übernimmt die Aufgabe, die von den Clients geschickten PostScript-Daten ins jeweils passende druckerspezifische Format umzuwandeln. Dieses Thema wird in Abschnitt 8.5 aufgegriffen.

Formatumwandlung auf dem Server

Übungen

8.1 [!2] Installieren Sie auf Ihrem System eine Warteschlange, deren Aufträge über SMB an einen Drucker weitergeleitet werden. (Ihr Trainer wird Ihnen einen Server- und einen Druckernamen sagen, oder Sie verschieben diese Aufgabe bis nach dem nächsten Abschnitt.)

8.2 [*3] CUPS unterstützt außer SMB auch die Weiterleitung von Aufträgen über das *Internet Printing Protocol* (IPP, [RFC2910, RFC2911]). Recherchieren Sie, wie Sie unter Windows einen IPP-Server zur Verfügung stellen können, und versuchen Sie, eine CUPS-Warteschlange zu installieren, die Aufträge an einen solchen weiterleitet.

8.3 Samba als Druckserver

Ablauf eines Druckvorgangs

Natürlich kann ein Samba-Server auch Druckfreigaben für Windows- und Linux-Clients anbieten. Auch hier betätigt Samba sich nur als Instanz in der Mitte, die zwischen SMB und dem lokalen Drucksystem vermittelt. Ein SMB-Client verbindet sich mit der Druckfreigabe, Samba authentisiert den Benutzer, und dann schickt der Client eine Datei mit den zu druckenden Daten an den Server, der sie in eine temporäre Datei schreibt. Der SMB-Client schließt die Verbindung, und Samba ruft ein passendes Druckkommando auf, um die temporäre Datei ans lokale Drucksystem zu übergeben. Je nach dem lokal installierten Drucksystem kann es danach erforderlich sein, die temporäre Datei explizit zu löschen.

Samba-Druckserver mit LPD

Eine einfache Konfiguration für einen Druckserver, die auf dem Samba-Server das LPD-Drucksystem voraussetzt, könnte so aussehen:

```
[global]
    printing = bsd
    load printers = yes
[printers]
    path = /var/spool/samba
    printable = yes
    guest ok = yes
    writeable = no
```

(Alle anderen Parameter, die mit dem Drucken zu tun haben, bekommen von Samba Standardwerte zugeordnet.)

Drucksystem wählen

Mit dem »printing«-Parameter wählen Sie das Drucksystem, das auf Ihrem Rechner installiert ist. Samba braucht diese Information, um die korrekten Druckkommandos für das lokale Abschicken der über SMB eingereichten Druckaufträge zu wählen. »load printers« sorgt dafür, dass alle lokalen Drucker in der Netzwerkumgebung sichtbar gemacht werden.

»printing« ist eigentlich ein freigabespezifischer Parameter, Sie könnten sich also für jede Druckerfreigabe ein anderes Drucksystem wünschen. Da man in der Regel aber auf einem Rechner nur eins installiert hat, werden Sie diesen Parameter üblicherweise im [global]-Abschnitt vorfinden.

Freigabe »printers«

Die Freigabe »printers« dient in Analogie zu »homes« dazu, bequemen Zugriff auf alle auf dem betreffenden Rechner verfügbaren Drucker zu ermöglichen. Sie enthebt Sie der Pflicht, für jeden einzelnen Drucker eine eigene Freigabe zu definieren. Statt dessen können Sie von einem SMB-Client aus die verschiedenen Drucker unter den Namen ansprechen, die sie im Linux-Drucksystem haben. Entsprechend gelten die in printers angegebenen Werte als Vorgaben für die einzelnen automatisch erstellten Druckerfreigaben.

Vorgehensweise

Die genaue Vorgehensweise ist laut smb.conf(5) wie folgt: Wenn ein Client sich mit einer Freigabe verbinden möchte, werden die explizit angegebenen Freigaben durchsucht, ob eine dabei ist, die so heißt wie die Freigabe, die der Client wünscht. Wird keine entsprechende Freigabe gefunden, aber ein homes-Abschnitt existiert in smb.conf, dann wird der Name der gewünschten Freigabe als Benutzername interpretiert und nach einem entsprechenden Heimatverzeichnis gesucht; wird auch kein solches gefunden und ein printers-Abschnitt existiert, dann wird der Freigabename als Druckername angesehen. Existiert ein passender Drucker, wird der printers-Abschnitt »geklont«, um eine Freigabe für den betreffenden Drucker zu konfigurieren. Dabei wird der Druckername als Freigabename verwendet; wenn die Freigabe keinen Gastzugriff erlaubt und der Client keinen Benutzernamen angegeben hat, dient der Druckername auch als Benutzername für die Authentisierung.

Wenn es darum geht, welche Drucker auf dem Server existieren, orientiert Samba sich an der Datei `/etc/printcap`. Wenn Sie ein Drucksystem benutzen, das keine solche Datei verwendet, oder nur eine Teilmenge Ihrer Drucker verfügbar machen wollen, können Sie eine »Pseudo-`printcap`«-Datei anlegen, die Zeilen der Form

`/etc/printcap`

```
lp|printer|laserjet
```

enthält. Jede Zeile steht dabei für einen Drucker und die durch vertikale Balken getrennten Namen als Synonyme für diesen. Verwenden Sie den Parameter »`printcap name`« in der `smb.conf`-Datei, um Samba dazu zu bringen, diese Datei anstelle von `/etc/printcap` zu lesen. – Die eigentlichen Druckerkonfigurationsangaben, die in einer »echten« `printcap`-Datei stehen, sind Samba egal.

Wenn Sie CUPS verwenden, brauchen Sie sich wegen Samba nicht um eine `printcap`-Datei (echt oder unecht) zu kümmern. Wenn Samba für CUPS konfiguriert ist (Abschnitt 8.4), holt es sich die Informationen über vorhandene Drucker direkt von CUPS.

CUPS vs. `printcap`

Druckerfreigaben werden durch den Parameter »`printable = yes`« als solche gekennzeichnet. Clients dürfen bei solchen Freigaben Druckdateien in das mit »`path`« angegebene Verzeichnis stellen (egal was ansonsten die Zugriffsrechte auf die Freigabe sind). Das »`writeable = no`« im Beispiel oben bezieht sich also darauf, direkt Dateien in das Verzeichnis schreiben zu dürfen (eine potentielle Sicherheitslücke).

Parameter »`printable`«

Die `printers`-Freigabe *muss* als »`printable`« gekennzeichnet sein, ansonsten verschmäht Samba die Konfigurationsdatei.

Bei Druckerfreigaben gibt »`path`« das Verzeichnis an, wo Samba eingehende Druckaufträge zwischenlagert, bevor sie an das lokale Drucksystem weitergereicht werden. Widerstehen Sie der Versuchung, für dieses Verzeichnis das lokale Spool-Verzeichnis der Druckerwarteschlange zu missbrauchen – das führt unter Garantie ins Chaos.

`path` für Druckerfreigaben

Im Beispiel erlauben wir mit »`guest ok = yes`« allen Benutzern im Netz Zugriff auf unsere Drucker. Es hält Sie aber niemand davon ab, die in Kapitel 7 diskutierten Mechanismen zu verwenden, um detailliertere Zugriffsrechte für die Freigabe zu vergeben.

Zugriffsrechte

Neben der globalen Freigabe aller Drucker über `printers` können Sie auch gezielt einzelne Drucker (bzw. Druckerwarteschlangen) für Samba freigeben. Definieren Sie ganz normale Freigaben und setzen Sie »`printable`« auf `yes`. Hier ist ein Beispiel:

Spezifische Freigaben

```
[quilljet]
    comment = QuillJet Printer
    printable = yes
    path = /var/spool/samba
    writeable = no
    browseable = yes
    printer admin = ron
    hosts allow = 192.168.22
    guest ok = no
```

Hier definieren wir eine Druckerfreigabe namens `quilljet`, auf die nur Clients aus dem Netz `192.168.22.0/24` zugreifen dürfen, und auch dann nur mit Benutzerauthentisierung (der Gastzugriff ist gesperrt).

Druckernamen

Wenn Sie nichts anderes sagen, bezieht eine Druckerfreigabe sich immer auf einen Drucker, der genauso heißt wie sie selbst. Sie können das übersteuern, indem Sie mit dem Parameter »`printer name`« einen anderen Druckernamen festlegen. Steht »`printer name`« im `global`-Abschnitt, dann gilt er für alle Druckerfreigaben, die keinen expliziten eigenen Druckernamen angeben.

Mit dem Parameter »`printer admin`« können Sie einen oder mehrere Benutzer benennen, die die Windows-Drucksteuerungsfunktionen verwenden dürfen, um beliebige Dinge mit dem Drucker zu tun (typischerweise von einem Windows-NT/2000/XP-Arbeitsplatz aus). Dieser Parameter ist allerdings verpönt und wird in einer künftigen Samba-Version entfernt werden, zugunsten einer Windows-artigen Privilegzuweisung mit dem `net`-Kommando. Die Details stehen zum Beispiel in [TV05, Kapitel 14]; suchen Sie nach »`SePrintOperatorPrivilege`«.

eigene Kommandos

Mit dem Parameter »`printing`« wählen Sie das Drucksystem aus, dessen Kommandos zum Einreichen von Druckaufträgen verwendet werden sollen. Allerdings hält Sie niemand davon ab, statt dessen eigene Kommandos zu definieren, etwa zur Fehlersuche oder für spezielle Aufgaben. Eine Definition wie

```
print command = echo Drucke %s >>/tmp/print.log; lpr -r -P %p %s
```

würde zum Beispiel den Namen der zu druckenden Datei in `/tmp/print.log` protokollieren, bevor das eigentliche Druckkommando aufgerufen wird. Im »`print command`« können Sie verschiedene »Makros« angeben, die Samba ersetzt, bevor es das Kommando an eine Shell zur Ausführung übergibt. Die Details finden Sie in `smb.conf`(5).

Entsprechend gibt es die Parameter »`lpq command`« zum Erheben des Druckerstatus, »`lppause command`« zum Unterbrechen und »`lpresume command`« zum Wiederaufnehmen einer Druckausgabe, »`queuepause command`« und »`queueresume command`« zum Sperren und Freigeben einer Druckerwarteschlange sowie »`lprm command`«, um einen Druckauftrag zu stornieren. Auch hier ergeben die Standardwerte sich aus dem Wert von »`printing`«.

Achten Sie darauf, alternative Kommandos zur Druckersteuerung *nach* dem Parameter »`printing`« zu vereinbaren. »`printing`« setzt nämlich alle Druckkommandos auf die jeweils passenden Standardwerte zurück – Ihre mühsam gemachten Einstellungen werden wieder vergessen.

Übungen

8.3 [!3] Verwenden Sie Berkeley-LPD, um eine lokale Warteschlange für einen Drucker einzurichten (der Drucker muss nicht lokal sein). Geben Sie den Zugriff auf diese Warteschlange anschließend über Samba frei.

8.4 [4] Installieren Sie ein Paket zur automatischen Formatumwandlung (etwa `apsfilter` oder `magicfilter`) und konfigurieren Sie Ihre Warteschlange so, dass sie dieses Paket verwendet (die einfachste Möglichkeit dazu ist unter Umständen Löschen und Neuanlegen). Testen Sie das Drucken von Daten in verschiedenen Formaten (ASCII-Text, PostScript, JPEG, …) von einem Windows-Client oder über Ihre Warteschlange aus Übung 8.1.

8.4 Samba und CUPS

CUPS, das *Common Unix Printing System*, verhält sich zu Berkeley-LPD wie ein Mercedes zu einem Eselskarren. Es ist leistungsfähig, flexibel und kann mit modernen Druckern perfekt umgehen. Aus diesem Grund basieren die Druckersubsysteme fast aller namhaften Linux-Distributionen inzwischen auf CUPS, oder CUPS wird zumindest als Option mitgeliefert.

Das ist Grund genug, dem Zusammenspiel zwischen Samba und CUPS einen eigenen Abschnitt zu widmen. Grundsätzlich kann Samba CUPS ansprechen wie ein Berkeley-LPD- oder System-V-Drucksystem; die aktuellen CUPS-Versionen in den Distributionen sind jedoch in der Regel so übersetzt, dass sie direkt auf CUPS zugreifen können und die externen Kommandos gar nicht benötigen. Dadurch wird eine Ebene der Integration erreicht, die diverse Anwendungen beträchtlich vereinfacht. CUPS ansprechen

Sie können herausfinden, ob Ihr Samba-Server direkt mit CUPS reden kann, indem Sie etwas wie

```
# ldd $(which smbd) | grep cups
        libcups.so.2 => /usr/lib/libcups.so.2 (0x40113000)
```

eingeben. Erscheint eine Ausgabe wie im Beispiel gezeigt, ist das der Fall.

Um die Direktverbindung zwischen Samba und CUPS zu aktivieren, sollten Sie im globalen Abschnitt der `smb.conf`-Datei die Parameter Direktverbindung

```
printing = cups
printcap file = cups
```

setzen.

Bloß wegen Samba müssen Sie keine `/etc/printcap`-Datei mehr erzeugen, da es sich die Informationen über installierte Drucker direkt von CUPS holen kann. Allerdings bestehen andere Programme darauf, zumindest die vorhandenen Drucker aus der `/etc/printcap`-Datei lesen zu können. CUPS ist in der Lage, eine »Mickymaus-Version« von `/etc/printcap` anzulegen; Informationen darüber finden Sie in der CUPS-Dokumentation.

In dem Moment, wo Sie »`printing = cups`« setzen, werden »`print command`« & Co. völlig links liegen gelassen. Wenn Sie Ihre eigenen Kommandos definieren wollen, sollten Sie CUPS mit »`printing = sysv`« betreiben.

Grundsätzlich kommt ein CUPS-basierter Druckerserver mit allen möglichen Druckdatenformaten zurecht. Windows-Clients verwenden aber in der Regel lokal installierte Druckertreiber, die auch aus Anwendungen heraus gestartet werden. Auf der Warteschlange landen dann die bereits für den Drucker aufbereiteten Daten – oft »Binärdaten«, mit denen CUPS gar nicht viel anfangen kann. Tatsächlich werden diese Daten von einem CUPS-Server oft sogar als undruckbar abgelehnt; CUPS weigert sich aus Sicherheitsgründen, beliebige Binärdaten an einen Drucker zu schicken, wenn Sie das nicht ausdrücklich erlauben. CUPS und »Binärdaten«

Es gibt natürlich viele Möglichkeiten, Drucker über *denial-of-service*-Angriffe lahmzulegen oder zumindest Mengen von Toner, Tinte oder Papier zu verschwenden. Solange das über die offiziellen CUPS-Kanäle geht, haben Sie dank der recht umfangreichen Protokollierung von CUPS gewissen Chancen, den oder die Schuldigen zu identifizieren. Bei Binärdaten, die CUPS nicht mehr vorfiltert, sieht die Welt allerdings anders aus.

Um das Drucken beliebiger Binärdaten freizuschalten, müssen Sie in den Dateien `/etc/cups/mime.types` und `/etc/cups/mime.convs` die anfänglichen Kommentarzeichen von den Zeilen entfernen, die mit Binärdaten drucken

```
#application/octet-stream …
```

anfangen. Anschließend können Sie mit CUPS eine binäre (*raw*) Druckerwarteschlange anlegen – etwa unter dem Namen `rawqueue` – und diese in `smb.conf` mit

```
[printers]
    printable = yes
    ◁◁◁◁◁
    use client driver = yes
    ◁◁◁◁◁
```

zugänglich machen. Auf dem Windows-Client installieren Sie dann zunächst einen »lokalen« Drucker (etwa mit Ausgabe nach `LPT1:`). Danach können Sie dessen Konfiguration ändern und unter »Details« einen »lokalen Port« anlegen, der auf die oben erzeugte binäre Druckerwarteschlange verweist, etwa `\\ALBUS\rawqueue` im Beispiel.

Nachteile des Ansatzes

Diese Methode – CUPS druckt binäre Druckdaten von manuell clientseitig installierten Druckertreibern – ist nicht nur unbequem zu handhaben, sondern bringt Sie auch um diverse Vorteile, die Sie ansonsten von CUPS praktisch geschenkt bekämen, wie Statistiken über die Druckausgabe pro Drucker, Benutzer oder Abteilung oder die Kontingentierung von Druckaufträgen nach Seitenanzahl oder Datenvolumen. Außerdem gibt es erfahrungsgemäß immer wieder Systemabstürze durch fehlerhafte Druckertreiber, die à la Windows NT im »Kernelmodus« ausgeführt werden. Sie tun daher gut daran, auf die Druckdaten-Aufbereitung durch zweifelhafte Windows-Druckertreiber so weit wie möglich zu verzichten und statt dessen CUPS die Arbeit machen zu lassen. Verwenden Sie auf den Windows-Clients lediglich einen PostScript-Druckertreiber, um CUPS mit PostScript zu füttern, das dann gegebenenfalls in druckerspezifische Daten umgewandelt wird. Solche Treiber gibt es von Adobe, Microsoft oder von der Herstellerfirma von CUPS, Easy Software Products – der letztere Treiber steht für Windows NT/200*x*/XP zur Verfügung und kann von `http://www.cups.org/software.html` heruntergeladen werden. Für die älteren Windows-Versionen ist der Adobe-Treiber empfehlenswert.

Gemäß der Samba-Dokumentation ist der Adobe-PostScript-Treiber überraschend schwer zu besorgen, da er nicht auf den Adobe-Webseiten in bequemer Form zum Herunterladen angeboten wird, sondern nur in Gestalt von selbstextrahierenden Archiven (und selbst so nicht leicht zu finden ist). Es wird empfohlen, die native Installationsprozedur auf einem Client durchzuführen, dort einen lokalen PostScript-Drucker anzulegen und diesen Drucker freizugeben. Anschließend können Sie sich per `smbclient` den Druckertreiber aus der `[print$]`-Freigabe auf diesem Rechner holen.

Übungen

8.5 [!2] Installieren Sie auf Ihrem Rechner eine CUPS-Druckerwarteschlange (der Drucker muss nicht lokal sein; wenn Sie gar keinen Drucker haben, dann installieren Sie eine Warteschlange, die in eine Datei druckt). Geben Sie diese Warteschlange anschließend über Ihren Samba-Server frei. Testen Sie sie von einem Windows-Client oder wie in Abschnitt 8.2 beschrieben von Linux.

8.5 Automatische Installation von Druckertreibern auf Windows-Clients

Während es zweifellos möglich ist, auf den Windows-Clients die erforderlichen Druckertreiber manuell zu installieren, besteht die anerkannte Methode heutzutage darin, die Treiber nur auf dem Druckserver zur Verfügung zu stellen, von wo die Clients sie sich dann halbautomatisch holen können. (Das gilt jedenfalls für Clients auf der Basis von Windows NT und seinen Abkömmlingen, während Sie für Windows 95, 98 oder ME auf die altertümliche Methode angewiesen sind.)

Für das Zurverfügungstellen der Treiber auf dem Druckserver gibt es im wesentlichen drei Methoden. Auf einem Rechner mit Windows NT, 200*x* oder XP Professional können Sie den *Add Printer Wizard* verwenden, auf einem Samba-Rechner die Kommandozeilenprogramme `smbclient` und `rpcclient` und auf einem Samba-Rechner mit CUPS das Programm `cupsaddsmb`. In jedem Fall benötigen Sie aber eine Dateifreigabe namens `print$`, die etwa wie folgt definiert sein sollte:

```
[global]
    ◁◁◁◁◁
    printer admin = @ntadmin                    Mitglieder der Gruppe ntadmin
    ◁◁◁◁◁
[print$]
    comment = Freigabe zum Herunterladen von Druckertreibern
    path = /etc/samba/drivers
    browseable = yes
    guest ok = yes
    read only = yes
    write list = root, @ntadmin
```

(Natürlich sollte das in `path` benannte Verzeichnis existieren.) Die Freigabe wird allgemein schreibgeschützt und kann nur von Drucker-Administratoren mit Treibern beschickt werden, vor allem aus Sicherheitsgründen – wer mag schon trojanische Druckertreiber? Um wirklich Treiber hinzufügen zu können, so dass Samba sie als solche kennt, reicht einfaches Schreibrecht übrigens nicht aus; die betreffenden Benutzer müssen entweder in »`printer admin`« enthalten sein oder auf der Linux-Seite die UID 0 haben.

Der Gastzugriff auf `print$` wird nur dann gebraucht, wenn Sie nicht sicher sein können, dass alle Windows-Benutzer vom Samba-Server authentisiert werden können. Insbesondere ist er unnötig, wenn alle Benutzer sich sowieso gegen eine Domäne authentisieren.

Innerhalb des Verzeichnisses `/etc/samba/drivers` sollten Sie Unterverzeichnisse für die verschiedenen Rechnerarchitekturen einrichten, die Sie mit Druckertreibern zu unterstützen gedenken. Von den in der Samba-Dokumentation zitierten fünf Architekturen für Windows sind heute nur noch NT-auf-x86 (Unterverzeichnis `W32X86`) und Windows 95/98/ME (Unterverzeichnis `WIN40`) übrig geblieben.

Nun zur tatsächlichen Treiberinstallation, auf drei verschiedene Arten:

Mit dem Add Printer Wizard Bevor Sie mit den Druckern auf dem Samba-Server etwas anfangen können, müssen Sie ihnen Treiber zuweisen. Eine Möglichkeit dazu ist der *Add Printer Wizard* von Windows NT/200*x*/XP. Auf dem Windows-Rechner öffnen Sie dazu den Samba-Rechner in der Netzwerkumgebung klicken mit der rechten Maustaste auf das Icon des gewünschten Druckers und wählen »Eigenschaften …«. Sie erhalten eine Fehlermeldung, dass kein Druckertreiber installiert ist, und werden gefragt, ob Sie den Treiber jetzt installieren wollen. *Antworten Sie* nicht *mit »Ja«!* Sagen Sie statt dessen »Nein« – Sie erhalten das Fenster mit den Druckereigenschaften und können anschließend den Treiber installieren[1].

Um den Treiber zu installieren, können Sie entweder einen aus der Liste der bereits installierten Treiber auswählen (diese Liste ist jetzt noch leer), oder den schon erwähnten *Add Printer Wizard* starten, indem Sie auf »Neuer Treiber« klicken. Ab hier geht es dann weiter wie in Windows üblich.

Stellen Sie sicher, dass Sie als ein Benutzer angemeldet sind, der tatsächlich Druckeradministrator-Privilegien genießt.

[1] Fragen Sie nicht. Dies ist offenbar ein klassisches Beispiel für die sprichwörtliche Intuitivität von Windows, die es unnötig macht, fest angestellte und ausgebildete Vollzeit-Administratoren zu beschäftigen, und dadurch die TCO im Vergleich zu komplizierten Betriebssystemen wie Linux gering hält.

Mit `rpcclient` Sie können auch von der Linux-Seite aus Druckertreiber installieren. Leider ist das etwas mühseliger. Zuerst müssen Sie die Dateien identifizieren, die zum Druckertreiber gehören. Den Distributions-CDs, die zum Drucker gehören, sieht man leider nicht ohne weiteres an, woraus der Treiber besteht, da die Dateien gerne in proprietären Formaten gepackt sind und bei der Installation mitunter auch noch umbenannt werden. Manchmal ist es am leichtesten, den Treiber auf einem Windows-Rechner zu installieren und zu schauen, was am Ende in `print$` steht und welche Dateien davon tatsächlich gebraucht werden. Eine Möglichkeit dafür bietet das Kommando `rpcclient`:

```
# rpcclient -U'ron%sc4bber5' \
>   'getdriver "…" 3' SEVERUS
◁◁◁◁
```

(In diesem Beispiel fand die Testinstallation auf dem Rechner `SEVERUS` statt.)

Als nächstes können Sie die Druckertreiber-Dateien auf den Server kopieren. Am bequemsten verwenden Sie dafür `smbclient`:

```
# smbclient //SEVERUS/print\$ -U'ron%sc4bber5' \
>   -c 'cd W32X86/2;mget …'
◁◁◁◁
```

Nach diesem Schritt stehen die Dateien im aktuellen Verzeichnis und können wiederum mit `smbclient` in die `print$`-Freigabe des Druckservers kopiert werden. Übernehmen Sie dafür exakt die Verzeichnisangaben vom ursprünglichen Rechner:

```
# smbclient //ALBUS/print\$ -U'ron%sc4bber5' \
>   -c 'cd W32X86; put …
◁◁◁◁
```

Zu guter Letzt müssen wir die Treiberdateien noch beim Samba-Druckserver als solche registrieren. Auch das geht wieder mit `rpcclient`:

```
# rpcclient -U'ron%sc4bber5' -c 'adddriver "Windows NT 86" \
>   … ALBUS
```

Danach sollte der Samba-Server diese Dateien eigentlich erkennen. Prüfen Sie, ob die Treiberdateien ins Unterverzeichnis (`2` oder `3`) des Architektur-Verzeichnisses kopiert worden sind (»`rpcclient adddriver`« macht das) und ob das Änderungsdatum der TDB-Dateien, die sich mit dem Drucken beschäftigen, und/oder ihre Größe sich geändert haben. Mit einem Kommando wie

```
# rpcclient -U'ron%sc4bber5' -c enumdrivers ALBUS
```

können Sie eine Liste der beim Samba-Server angemeldeten Druckertreiber abrufen; der neue Treiber sollte darin vorkommen.

Sie könne `rpcclient` auch verwenden, um einem Drucker einen Treiber zuzuordnen. Das Kommando dafür entspricht

```
# rpcclient -U'ron%sc4bber5' -c 'setdriver … …' ALBUS
  ◁◁◁◁
```

Dazu muss der Drucker bei Samba bereits bekannt sein.

Mit `cupsaddsmb` Die dritte Methode zur Treiberinstallation auf dem Druckserver setzt voraus, dass auf dem Druckserver CUPS installiert ist, und verwendet das Programm `cupsaddsmb`. Es ist spezialisiert dafür, die verschiedenen PostScript-Druckertreiber von Microsoft, Adobe und Easy Software Products zu bearbeiten. `cupsaddsmb` automatisiert im Wesentlichen die eben gezeigte Vorgehensweise mit `rpcclient` und `smbclient` auf der Basis eines ausgepackten Treiberarchivs in `/usr/share/cups/drivers`. Nähere Details finden Sie in [TV05, Kap. 21].

Um einen Druckertreiber auf einem Windows-Client zu installieren, gehen Sie wie folgt vor: Stellen Sie sicher, dass Sie auf dem Windows-Client als Benutzer mit Druckadministrations-Privilegien angemeldet sind. Anschließend begeben Sie sich in der Netzwerkumgebung zum Eintrag des Samba-Servers, öffnen dort den »Drucker und Faxe«-Ordner und klicken mit der rechten Maustaste auf das Icon des gewünschten Druckers. Wählen Sie hier »Verbinden« (oder gegebenenfalls »Installieren«) aus. Damit sollte im *lokalen* Ordner »Drucker« ein neuer Drucker auftauchen.

Bevor Sie tatsächlich Ausgabe an diesen neuen Drucker schicken können, müssen Sie sicherstellen, dass der Treiber für den Drucker einen »Gerätemodus« und einen vollständigen Satz Treiberdaten erzeugt hat. Andernfalls kann es zu suboptimaler oder gar völlig unlesbarer Ausgabe kommen.

Gerätemodus

Bei diesen Daten handelt es sich im Wesentlichen um Voreinstellungen für alle Eigenschaften einer Druckerwarteschlange, die vernünftige Werte liefern und garantieren sollen, dass man einen Drucker sofort verwenden kann.

Unbequemerweise sind Windows-Druckertreiber Windows-Code, der auf einem Linux-Rechner nicht ausgeführt werden kann. Um das erste Mal einen Gerätemodus zu setzen, müssen Sie also den Treiber auf dem Samba-Server von einem Client aus »anschubsen«, etwa indem Sie die Seitenorientierung (hoch oder quer) ändern. Dadurch wird auf dem Client genug vom Treiber ausgeführt, um einen Gerätemodus zu erzeugen und an den Server zurück zu schicken.

Wählen Sie dazu aus dem Kontextmenü des Druckers den Eintrag »Eigenschaften« (sollte nach dem oben angegebenen Verbindevorgang erschienen sein). Gehen Sie auf *Advanced* und klicken Sie auf *Printing Defaults*. Stellen Sie dort die Seitenorientierung auf »quer« und wieder zurück; sorgen Sie jedes Mal dafür, dass die Änderungen zwischendurch akzeptiert wurden. Sie können außerdem Voreinstellungen für den Druckertreiber treffen, die dann an alle späteren Clients als Standard weitergegeben werden.

Es ist sehr wichtig, dass dieser erste Installationsversuch funktioniert. Sorgen Sie dafür, dass Sie ihn als ein Benutzer unternehmen, der Druckerverwaltungs-Privilegien hat.

Wenn Sie den Treiber auf weiteren Clients installieren, ist der letzte Schritt nicht mehr erforderlich.

Diese Vorgänge sind in epischer Breite in [TV05, Kap. 20] beschrieben.

Übungen

8.6 [3] Jetzt geht es ans Eingemachte: Besorgen Sie sich, wie oben beschrieben, einen PostScript-Treiber für Windows und arrangieren Sie, dass Windows-Clients diesen von Ihrem Samba-Server abrufen können.

Kommandos in diesem Kapitel

cupsaddsmb Exportiert Drucker an Samba für Windows-Clients
cupsaddsmb(8) 122

rpcclient Werkzeug zum Ausführen von MS-RPC-Funktionen
rpcclient(1) 120

smbspool Schickt eine Druckdatei an einen SMB-Drucker smbspool(8) 114

Zusammenfassung

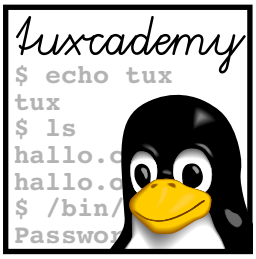
- Samba enthält keine eigenen Druckfunktionen, sondern vermittelt nur zwischen SMB-Clients und dem Drucksystem des Servers.
- Unix- und Linux-Rechner können SMB-Druckerfreigaben, egal ob sie von einem Windows- oder Samba-Server zur Verfügung gestellt werden, über das `smbclient`-Programm oder über CUPS ansprechen.
- Samba erlaubt die Auswahl verschiedener Unix-seitiger Drucksysteme.
- Es ist möglich, Samba über explizite Kommandos an beliebige Drucksysteme anzupassen.
- Über die Freigabe `printers` werden alle auf dem Samba-Server zugänglichen Druckerwarteschlangen als Freigaben zugänglich gemacht.
- Samba enthält spezielle Unterstützung für das verbreitete und leistungsfähige Drucksystem CUPS.
- Samba unterstützt die halbautomatische Installation von Druckertreibern auf Windows-Clients über den Server.

Literaturverzeichnis

RFC2910 R. Herriot, S. Butler, P. Moore, et al. »Internet Printing Protocol/1.1: Encoding and Transport«, September 2000.
http://www.ietf.org/rfc/rfc2910.txt

RFC2911 T. Hastings, R. Herriot, R. deBry, et al. »Internet Printing Protocol/1.1: Model and Semantics«, September 2000.
http://www.ietf.org/rfc/rfc2911.txt

TV05 John H. Terpstra, Jelmer R. Vernooij. *Samba 3 – das offizielle Handbuch*. München: Addison-Wesley, 2005. ISBN 3-82732152-2.



9

Das Network File System (NFS)

Inhalt

Lernziele

- Die Komponenten von NFS verstehen
- NFS-Server und -Clients einrichten können
- Fehler bei der Konfiguration finden und beheben können

Vorkenntnisse

- TCP/IP-Kenntnisse
- Kenntnisse über das Starten von Systemdiensten

9.1 Einleitung

Netzwerk-Dateisystem

NFS ist ein einfach konfigurierbares, simples und verbreitetes Netzwerk-Dateisystem für Unix-Systeme (mit Client-Implementierungen für andere Systeme). Es wurde in den 1980er Jahren von Sun Microsystems entwickelt, ist im Vergleich zu SMB/Samba extrem primitiv und enthält keine besonderen Sicherheitsvorkehrungen. NFS-Clients stehen auch für andere (Nicht-Unix-)Betriebssysteme zur Verfügung.

Grundidee

Die Grundidee hinter NFS ist, dass NFS-Server gewisse Verzeichnisse *exportieren* (analog den Freigaben von Samba). Clients können diese Verzeichnisse dann einhängen (»mounten«) und auf die darin enthaltenen Unterverzeichnisse und Dateien so zugreifen, als lägen sie lokal vor. Dabei wird versucht, die Unix-Dateisystemsemantik so weit wie möglich zu erhalten.

Sun-RPC

zustandslose Implementierung

NFS beruht auf dem Sun-RPC-Mechanismus und (normalerweise) UDP. Eines der wichtigeren Entwurfsziele von NFS war eine soweit möglich zustandslose Implementierung des Servers; als NFS neu war, waren Rechner und Netze noch nicht so stabil wie heute, und der Ausfall und Neustart eines Servers durfte den Betrieb nicht nennenswert aufhalten. Gewisse Aspekte der Unix-Dateisystemsemantik leiden unter dieser Anforderung der Zustandslosigkeit, so dass ein über NFS eingehängtes Dateisystem sich nicht unter allen Bedingungen so wie ein Dateisystem auf der lokalen Platte verhält.

Die verbreitetste Geschmacksrichtung von NFS ist die Version 3 (kurz »NFSv3«). Es gibt auch eine Version 4, die gegenüber NFSv3 wesentliche Vorteile hat; allerdings existieren nicht so viele Implementierungen davon. Der Rest dieses Kapitels behandelt NFSv3, das wir der Einfachheit halber nur »NFS« nennen.

9.2 Sun RPC

9.2.1 Grundlagen

NFS basiert, wie erwähnt, auf dem RPC-Dienst von Sun [RFC1057], also einer Methode für »entfernte Prozeduraufrufe« (*remote procedure calls*). Kurz gesagt stellt ein RPC-Server eine Schnittstelle zur Verfügung, die – ähnlich einer Programmbibliothek – aus Prozedurnamen mit Parametern und definierten Rückgabewerten besteht. Der RPC-Dienst macht es dann möglich, dass Clients auf anderen Rechnern diese Prozeduren über das Netz mit passenden Parametern aufrufen und die Rückgabewerte geliefert bekommen.

Dies im Gegensatz zu »herkömmlichen« Anwendungsprotokollen, wo in der Regel eine Verbindung geöffnet und (meist) textuelle Nachrichten hin und her geschickt werden.

Natürlich ist die Idee entfernter Prozeduraufrufe mit allen möglichen praktischen Problemen gespickt, etwa dass verschiedene Rechnerarchitekturen unterschiedliche Formate verwenden, um bestimmte Datentypen darzustellen. Es ist die Aufgabe einer Infrastruktur wie Sun RPC, diese Unterschiede auszugleichen. Sun RPC bedient sich dazu eines definierten Datenformats namens XDR (engl. *External Data Representation*, [RFC1014]).

XDR

Sie können in Sun RPC die drei oberen Schichten des ISO/OSI-Referenzmodells wiederfinden: RPC selbst gehört zur »Sitzungsschicht«, XDR zur »Darstellungsschicht« und Dienste wie NFS zur »Anwendungsschicht«.

Sun RPC unterscheidet verschiedene »Dienste«, also Programme, die RPC-Aufrufe anbieten, anhand einer »Programmnummer«, die lose den Ports von TCP und UDP entspricht. Programmnummern sind vier Byte lang, so dass grundsätzlich um die 4 Milliarden verschiedene RPC-Dienste möglich wären; die Popula-

Nummern	Bedeutung
`0x00000000–0x1fffffff`	Offiziell vergebene Programmnummern; entsprechen den *well-known ports* bei TCP und UDP. Diese Programmnummern sollen auf jedem Rechner das gleiche Programm bezeichnen. Nummern aus diesem Bereich müssen bei Sun Microsystems (heuer wahrscheinlich Oracle) registriert werden.
`0x20000000–0x3fffffff`	Nummern für den lokalen/privaten Gebrauch.
`0x40000000–0x5fffffff`	Dynamisch von Anwendungen erzeugte Programmnummern (denken Sie da nicht mal drüber nach).
`0x60000000–0xffffffff`	Reserviert für zukünftigen Gebrauch (*fat chance*).

Tabelle 9.1: Sun-RPC-Programmnummern

rität von Sun RPC bleibt im wirklichen Leben allerdings weit dahinter zurück[1]. Tabelle 9.1 erklärt, wie die Programmnummern aufgeteilt sind.

In Analogie zu `/etc/services` gibt es eine Datei `/etc/rpc`, die RPC-Programmnummern textuelle Namen zuordnet. Der NFS-Dienst hat zum Beispiel den Namen `nfs` und die Nummer 100003.

Übungen

9.1 [!1] Vergleichen Sie die Dateien `/etc/services` und `/etc/rpc`. Welche Einträge in `/etc/rpc` sagen Ihnen etwas?

9.2.2 Der Portmapper

Sun RPC macht keine Annahmen über den Transportmechanismus für Prozeduraufrufe und Rückgabewerte – aber natürlich kann das Thema nicht völlig ignoriert werden. Um RPC-Programmnummern mit Details wie Transportprotokollen und TCP/IP-Portnummern zusammenzubringen, gibt es den **Portmapper**, der eine Art Verzeichnisdienst für RPC-Dienste darstellt. Um einen RPC-Dienst auf einem entfernten Rechner anzusprechen, kontaktiert ein RPC-Client zunächst den Portmapper auf diesem Rechner. Dieser kann ihm dann mitteilen, über welchen (typischerweise UDP-)Port der eigentliche RPC-Dienst zu erreichen ist.

Portmapper

Der Portmapper ist selber ein RPC-Dienst. Bevor Sie hier ein Henne-Ei-Problem wittern, lassen Sie sich gesagt sein, dass dem Portmapper fest der TCP- bzw. UDP-Port 111 zugeordnet ist. Das ausführbare Programm heißt `portmap`; als RPC-Dienst hat es die Nummer `100000/portmapper` und der Dienstname des Ports 111 in `/etc/services` ist `sunrpc`.

Bevor ein RPC-basierter Dienst von Clients angesprochen werden kann, muss das betreffende Programm sich beim Portmapper registrieren. Dazu teilt es dem Portmapper seine RPC-Programmnummer und seine(n) TCP- und/oder UDP-Port(s) mit. Wenn das Programm sich beendet, muss es sich beim Portmapper auch wieder abmelden.

Registrierung

Ein Programm, das RPC-Dienste anbieten möchte, läßt sich einfach beim Start beliebige Portnummern vom Betriebssystem zuteilen. Es ist die Aufgabe des Portmappers, interessierten Clients diese Portnummern anhand der RPC-Programmnummer mitzuteilen. Die Idee dahinter – man hätte ja auch gleich wie bei anderen TCP/IP-Diensten feste Portnummern registrieren können – ist eine Flexibilisierung, die zunächst wohl feste Portnummern sparen sollte, sich langfristig allerdings als weniger klug herausgestellt hat.

Außer RPC-Programmnummer und Transportdienst-Ports merkt der Portmapper sich auch noch eine Versionsnummer für die vom RPC-Dienst an-

Versionsnummer

[1] Wenn wir gehässig wären, würden wir sagen, dass auch ein Byte gereicht hätte.

gebotene Aufrufschnittstelle. Damit kann dasselbe RPC-Programm gleichzeitig mehrere Versionen der Aufrufschnittstelle unterstützen, indem es alle diese Versionen beim Portmapper registriert. Ein RPC-Client gibt für die Suche nach einem Dienst nicht nur die RPC-Programmnummer, sondern auch die Versionsnummer der Aufrufschnittstelle an, so dass der Portmapper die richtige »Verbindung« herstellen kann.

Der Portmapper merkt sich die bei ihm registrierten Dienste nicht dauerhaft. Wenn Sie den Portmapper neu starten, müssen Sie also auch alle RPC-Dienste neu starten, damit sie sich neu registrieren können.

Übungen

9.2 [2] Warum ist die flexible Zuordnung von TCP- bzw. UDP-Ports zu RPC-Diensten nicht wirklich eine gute Idee?

9.2.3 Diagnosewerkzeuge für Sun RPC

Übliche Diagnosewerkzeuge

Für eine grundlegende Fehlersuche bei Sun-RPC-Diensten können Sie zunächst die üblichen Diagnosewerkzeuge einsetzen. Mit den Programmen `ps`, `netstat` oder `tcpdump` sowie bei LSB-konformen Distributionen durch die Init-Skripte (beispielsweise bei SUSE/Novell »`rcportmap status`«) lässt sich einiges in Erfahrung bringen.

`rpcinfo`

Speziell für die Schnittstelle zum Portmapper gibt es aber noch ein weiteres Werkzeug: Das Programm **`rpcinfo`** befragt einen lokalen oder entfernten Portmapper nach den den bei ihm registrierten RPC-Programmen.

```
# rpcinfo -p localhost
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp  32806  nlockmgr
    100021    3   udp  32806  nlockmgr
    100021    4   udp  32806  nlockmgr
    100005    1   udp  32807  mountd
    100005    1   tcp  33149  mountd
    100005    2   udp  32807  mountd
    100005    2   tcp  33149  mountd
    100005    3   udp  32807  mountd
    100005    3   tcp  33149  mountd
```

Die Option `-p` müssen Sie dabei immer angeben; wenn es um den `localhost` geht, kann der Rechnername auch entfallen. `rpcinfo` zeigt die Namen und Programmnummern aller registrierten Programme an, so wie diese sich in der Datei `/etc/rpc` finden; hinzu kommen noch die für diese Programmnummer unterstützten Versionen sowie der jeweilige (aktuelle) Port.

Registrierung löschen

Mit `rpcinfo` können Sie sich nicht nur die registrierten Programme auflisten lassen, sondern auch eine (lokale) Registrierung eines Programms löschen:

```
# rpcinfo -d mountd 1
```

hebt die Registrierung von Programm `100005/mountd` in der Version 1 auf. Die Version muss immer angegeben werden; den Dienst können Sie als RPC-Programmnummer oder als Namen in `/etc/rpc` angeben.

Übungen

9.3 [!2] Überprüfen Sie, ob der Portmapper läuft, und starten Sie ihn gegebenenfalls. Vergleichen Sie die Ausgabe von `rpcinfo` bei Befragung des lokalen Rechners mit der bei Befragung eines entfernten Rechners.

9.4 [2] Starten Sie den NFS-Server durch eines der folgenden Kommandos:

```
# /etc/init.d/nfs start                          # RedHat
# /etc/init.d/nfsserver start                    # SUSE
# /etc/init.d/nfs-kernel-server start            # Debian
```

Kontrollieren Sie das Ergebnis mit `ps` usw. und mit `rpcinfo`. (Sollte der NFS-Server nicht starten, so fügen Sie die Zeile

```
/tmp *(rw,sync)
```

an die Datei `/etc/exports` an und probieren es noch einmal.) Ordnen Sie die Ausgabe von `rpcinfo` einzelnen Prozessen zu.

9.5 [2] Starten Sie zuerst den Portmapper und dann den NFS-Server wie in Übung 9.4 beschrieben. Stoppen Sie nun den Portmapper und starten ihn nach einiger Zeit wieder. Kontrollieren Sie dabei des Ergebnis jeweils mit `rpcinfo` und `ps`. Gibt es einen Unterschied zum ursprünglichen Zustand? Wie können Sie diesen gegebenenfalls wieder herstellen?

9.6 [2] Starten Sie zuerst den Portmapper und dann den NFS-Server wie in Übung 9.4 beschrieben. Entfernen Sie die Registrierungen von `mountd` beim Portmapper durch »`rpcinfo -d`«. Kontrollieren Sie dabei des Ergebnis jeweils mit `rpcinfo` und `ps`.

9.2.4 Zugriffskontrolle

TCP-Wrapper

Die durch den Internet-Daemon `inetd` bekannte Zugriffskontrolle über die TCP-Wrapper ist auch für Remote Procedure Calls verfügbar. Sofern die entsprechenden RPC-Programme mit Unterstützung dafür kompiliert wurden, beachten sie die entsprechenden Einträge in den Dateien `/etc/hosts.allow` und `/etc/hosts.deny`. Der TCP-Wrapper muss dabei nicht als eigenständiges Programm aufgerufen werden.

Wenn Sie bei der Benutzung von RPCs Fehlermeldungen wie

```
# rpcinfo -p 192.168.0.42
No remote programs registered.
```

bekommen, so empfielt sich die probeweise Deaktivierung der TCP-Wrapper durch den Eintrag

```
ALL: ALL
```

am Anfang der Datei `/etc/host.allow`. Sollte RPC nach dieser Deaktivierung wie gewünscht funktionieren, so ist offensichtlich ein Eintrag zum Freischalten erforderlich. (Das dazu zu verwendende Schlüsselwort entnehmen Sie der Handbuchseite des RPC-Programms; es kann vom Programmnamen, so wie ihn z. B. `ps` zeigt, abweichen.)

Eintrag zum Freischalten

Der Portmapper selbst wird zum Beispiel durch die Einträge

```
# /etc/hosts.allow
portmap: 192.168.0.0/255.255.255.0
```

und

```
# /etc/hosts.deny
portmap: ALL
```

Portmapper macht keine IP-Namensauflösung

nur für das Netz `192.168.0/24` freigeschaltet. Zugriffskontrollen für das Loopback-Interface oder jenseits von IP-Adressen sind nicht möglich. Der Portmapper ignoriert die Auflösung von Rechnernamen u. ä., um die Gefahr von Endlosschleifen zu umgehen: die Namensauflösung könnte auf NIS beruhen, also auf RPC, was zwingend die Benutzung des Portmappers erfordert.

Um die Zugangskontrolle zu aktivieren, reicht es nicht, nur den Portmapper freizuschalten: auch die einzelnen RPC-Programme können für sich die TCP-Wrapper-Bibliotheken benutzen und müssen daher freigegeben werden. Umgekehrt reicht es nicht, den Zugang nur zum Portmapper zu beschränken, denn der Portmapper stellt nur eine Vermittlungsinstanz dar, die nicht zwingend benötigt wird. Ein Angreifer könnte die betreffenden Ports durch einen Port-Scan herausbekommen oder einfach »raten« (letzteres ist erfolgversprechender, als es sich anhört: die zentrale Komponente von NFS beispielsweise benutzt fast immer den festen Port `2049/udp`).

Übungen

9.7 [2] Starten Sie zuerst den Portmapper und dann den NFS-Server wie in Übung 9.4 beschrieben. Sperren Sie den Zugriff auf den Rechner durch den Eintrag

```
ALL: ALL
```

in `/etc/hosts.deny`. Geben Sie den Zugriff nur auf den Portmapper durch einen geeigneten Eintrag in `/etc/hosts.allow` wieder frei.

Kontrollieren Sie dabei des Ergebnis jeweils lokal und von einem entfernten Rechner mit `rpcinfo`.

9.3 Komponenten von NFS

NFS besteht aus mehreren Komponenten, die als unabhängige RPC-Server implementiert sind. Unter Linux unterscheidet man »User-Mode NFS«, bei dem alle Komponenten als Serverprogramme vorliegen, und »Kernel-Mode NFS«, das Teile des NFS-Servers in den Kernel verlegt. Heute ist letzteres allgemein gebräuchlich. Die Unterschiede sind subtil. Wir betrachten in diesem Kapitel nur Kernel-Mode NFS.

NFS-Server
Mount-Daemon
NFS-Daemon

Auf einem NFS-Server werden außer dem Portmapper der NFS-Mount-Daemon und der NFS-Daemon im eigentlichen Sinne benötigt. Der Mount-Daemon kümmert sich um Anfragen von Clients, die exportierte Verzeichnisse einhängen möchten. Der NFS-Daemon bearbeitet Anfragen nach Datei- und Verzeichnisinhalten.

NFS-Client

Auf einem NFS-Client werden für den reinen Dateizugriff keine Daemon-Programme benötigt. Eine Version von `mount`, die NFS unterstützt, genügt. Für gewisse Sonderfunktionen trifft das jedoch nicht zu.

Lokale Dateisysteme unterstützen das Konzept von Dateisperren (*file locking*). Dies beißt sich fundamental mit der Zustandslosigkeit von NFS und wird darum über einen besonderen Server zur Verfügung gestellt, den Lock-Daemon oder (`lockd`). Dieser Server muss auf dem Server *und* dem Client laufen; in diesem Fall braucht auch der Client den Portmapper. Für Dateisperren ebenfalls auf dem Server und dem Client nötig ist der Status-Daemon (`statd`).

Lock-Daemon
Status-Daemon

Remote-Quota-Daemon

Schließlich können Sie auf dem Server den Remote-Quota-Daemon laufen las-

sen, wenn auf NFS-Dateisystemen Plattenplatzkontingentierung verwendet wird. Dieses Programm erlaubt es, dass Benutzer auf Clients das `quota`-Kommando aufrufen, um ihre Quota zu überprüfen.

Übungen

9.8 [3] Halten Sie es für eine gute Idee, den NFS-Server in den Linux-Kern zu verlegen? Diskutieren Sie die Vor- und Nachteile.

9.4 NFS: Konfiguration und Betrieb

9.4.1 NFS-Dateisysteme verwenden

Einhängen Wie unter Linux (naja Unix) üblich müssen Dateisysteme eingehängt (»gemountet«) werden, damit die darauf abgelegten Dateien zugänglich sind. Ein Client muss – mit einem entsprechenden `mount`-Kommando – den NFS-»Mount-Daemon« auf dem Server ansprechen. Dieser überprüft, ob (und wie) der Client das entsprechende Verzeichnis einhängen darf und übergibt ihm ein Dateisystem-»Handle«, das dieser für die Kommunikation mit dem eigentlichen NFS-Server benutzt. Der Client meldet sich beim Mount-Daemon wieder ab, wenn er das Verzeichnis aushängt.

Der Mount-Daemon führt zwar eine Liste aller eingehängten Verzeichnisse und aktiven Clients, allerdings dient die Liste nur zur Dekoration. Wer ein Dateisystem-Handle hat, kann das – jedenfalls aus der Sicht des NFS-Servers – auch benutzen, selbst wenn er vorher `umount` gesagt hat.

Dateizugriff Für die eigentlichen Dateioperationen ist der »NFS-Daemon« zuständig. Da er stark beansprucht wird, liegt er üblicherweise in mehreren Instanzen (Prozesse oder Threads) vor. Die Anzahl der Instanzen kann beim Starten des NFS-Daemons angegeben werden (wie genau, hängt von der jeweiligen Distribution ab).

Als große Ausnahme registriert sich der NFS-Daemon beim Portmapper mit den festen Portnummern `2049/udp` und `2049/tcp` (TCP-Unterstützung ist nicht immer implementiert). Andere Ports sind aber durch explizite Konfiguration möglich.

NFS-Daemon: zustandslos

Der NFS-Daemon ist als *zustandsloser* Server konzipiert. Der Grund hierfür liegt vor allem in einer einfacheren Handhabung von Server-Ausfällen: Für den Client ist ein abgestürzter und neu gestarteter Server von einem sehr langsamen Server nicht zu unterscheiden. (Dies ist auch der Grund für die feste Portwahl; so kann der Client davon ausgehen, dass sich der Port auch nach einem Absturz nicht ändert, und muss daher nicht vor jeder Dateioperation vorsorglich den Portmapper konsultieren.) Damit trotz der Zustandslosigkeit des Servers (schreibende) Dateizugriffe nicht zum Glücksspiel werden, wiederholt der Client seine Aufrufe, bis er vom Server eine Bestätigung erhält. Umgekehrt führt der Server identische Schreiboperationen, die kurz hintereinander eingehen, nur einmal aus. Damit führt zum Beispiel das Löschen einer bereits gelöschten Datei nur dann zu einer Fehlermeldung (»Datei existiert nicht«), wenn zwischen dem ersten und zweiten Löschversuch Zeit verstrichen ist, aber nicht, wenn der ungeduldige Client die Löschaufforderung mehrmals hintereinander abschickt.

Auch mit UDP als Transportprotokoll sind Dateioperationen bei NFS »zuverlässig«. Der Umstieg auf TCP bringt daher keinen Zuverlässigkeitsgewinn, sondern beschleunigt lediglich bei Langstreckenverbindungen den Dateizugriff, da keine Pakete »auf Verdacht« mehrmals gesendet werden. Für Kurzstreckenverbindungen hingegen ist UDP schneller.

Tabelle 9.2: Optionen für das Exportieren von Verzeichnissen per NFS (Auswahl)

Option	Bedeutung
ro	Nur Lesezugriff
rw	Lese- und Schreibzugriff
root_squash	Zugriffe von root erfolgen mit den Rechten des anonymen Benutzers nobody. Dies ist die Voreinstellung.
all_squash	Zugriffe von jedem Benutzer erfolgen mit den Rechten des anonymen Benutzers nobody.
anonuid	Setzt den anonymen Benutzer für root_squash und all_squash auf einen anderen Wert, z. B. anonuid=4711. Erwartet wird die numerische UID; voreingestellt ist −2, also nobody.
anongid	Wie anonuid, allerdings für die Gruppe des anonymen Benutzers; voreingestellt ist −2.
async	Der Server darf das NFS-Protokoll verletzen und Anfragen bestätigen, bevor die betreffenden Daten auf die Platte geschrieben sind. Dies war früher die Voreinstellung, ist es heute aber nicht mehr (Sie müssen sich das ausdrücklich wünschen). Um Sie auf diesen Umstand hinzuweisen, gibt der NFS-Server eine Warnung an, wenn Sie weder async noch sync (das Gegenteil) angeben.

9.4.2 Konfiguration des Servers

/etc/exports
Exportieren

Die Konfiguration des Servers ist sehr einfach: Sie müssen lediglich in der Datei /etc/exports vermerken, welche Verzeichnisse für wen und wie freizugegeben sind. Im NFS-Sprachgebrauch heißt ein solches Freigeben **Exportieren**. Die Einträge in der /etc/exports könnten etwa so aussehen:

```
/usr                    *(ro)
/nfs-exports/home       192.168.42.0/24(rw)
/nfs-exports/ag14       *(ro) 192.168.14.0/24(rw,all_squash)
```

Mit der ersten Zeile wird das Verzeichnis /usr für beliebige Clients (*) für Lesezugriffe (ro) freigegeben; die zweite erlaubt Clients aus dem angegebenen Subnetz Lese- und Schreibzugriffe (rw); die dritte Lesezugriff für alle und Schreibzugriff für das Subnetz, wobei durch all_squash für alle Zugriffe die UID auf die von *nobody* gesetzt wird (Idee: Eine Arbeitsgruppe teilt sich ein Verzeichnis).

Eine Auswahl der gängigsten Optionen, die Sie beim Exportieren von Verzeichnissen angeben können, finden Sie in Tabelle 9.2. Die komplette Liste steht in exports(5).

Init-Skripte

Die benötigten Daemons werden üblicherweise über Init-Skripte gestartet, die den verschiedenen Distributionen beiliegen. Für deren Namen gibt es keine Standards; schauen Sie sich auf Ihrem System um.

Sie können durch das Kommando exportfs Verzeichnisse *ad hoc* freigeben bzw. Freigaben zurücknehmen, ohne den NFS neu zu starten. Somit verhält sich exportfs zu /etc/exports wie mount zu /etc/fstab.

Übungen

9.9 [!2] Konfigurieren Sie Ihren Rechner als NFS-Server so, dass er das Verzeichnis /tmp für alle anderen Rechner im lokalen Netz schreibbar freigibt. Zugriffe des Benutzer root sollen auf die Benutzerkennung nobody abgebildet werden. Prüfen Sie die Konfiguration mit dem Kommando »showmount -e localhost«.

9.10 [2] Ändern Sie die Konfiguration aus der vorigen Aufgabe so, dass die Zugriffe *aller* Benutzer auf nobody abgebildet werden. Vergewissern Sie sich,

Tabelle 9.3: NFS-Optionen für `mount` (Auswahl)

Option	Bedeutung
`rsize`	Anzahl der auf einmal vom NFS-Server gelesenen Bytes. Durch etwas wie `rsize=8192` wird der Durchsatz deutlich erhöht.
`wsize`	Wie `rsize`, allerdings für Schreibzugriffe.
`tcp`	Als Transportprotokoll wird TCP statt UDP (der Voreinstellung) gewählt. Nur wenige NFS-Server unterstützen TCP.
`nolock`	Deaktiviert *file locking*, d. h. der Lock-Daemon wird nicht gestartet.
`hard`	Schreib- und Leseversuche werden beliebig lange wiederholt. Dies ist die Voreinstellung.
`soft`	Schreib- und Leseversuche werden nach einigen Fehlschlägen (Timeouts) abgebrochen.
`intr`	Erlaubt das Abbrechen von Dateioperationen durch das Senden von Signalen. Nützlich, falls die Option `hard` aktiv ist.

dass dies für Lese- und Schreibzugriffe das korrekte Resultat liefert. (Arbeiten Sie ggf. erst den Rest des Kapitels durch.)

9.4.3 Konfiguration des Clients

Auf der Client-Seite werden NFS-exportierte Verzeichnisse einfach eingehängt, entweder direkt mit `mount` oder über einen Eintrag in der Datei `/etc/fstab`. Einzige Vorbedingung ist ein NFS-fähiges `mount`-Programm.

`mount`
`/etc/fstab`

Das sieht dann in der Datei `/etc/fstab` z. B. so aus:

```
172.16.0.200:/nfs-exports/usr /usr nfs defaults,ro 0 0
```

oder *ad hoc* als Kommando

```
# mount -t nfs -o ro 172.16.0.200:/nfs-exports/usr /usr
```

Dabei kann die Angabe des Typs mit »`-t nfs`« entfallen, da dies bereits durch den Doppelpunkt zwischen Rechnername und Verzeichnis-Pfad verraten wird. Eine Reihe von Optionen für das Einhängen, die mit `-o` angegeben werden können, steht in Tabelle 9.3.

Die komplette Liste von Optionen für `mount` steht in `mount`(8), was die dateisystemunabhängigen Optionen anbetrifft; die NFS-spezifischen finden Sie in `nfs`(5).

Übungen

9.11 [!2] Geben Sie ein `mount`-Kommando an, mit dem Sie das `/tmp`-Verzeichnis eines anderen Rechners »hart« einhängen können.

9.12 [!1] Geben Sie eine Zeile für die Datei `/etc/fstab` an, mit dem Sie das `/tmp`-Verzeichnis eines anderen Rechners »hart« einhängen können.

9.13 [3] Hängen Sie das `/tmp`-Verzeichnis eines anderen Rechners ein und stoppen Sie auf diesem den NFS-Dienst. Was passiert, wenn Sie trotzdem versuchen, von Ihrem Rechner auf das Dateisystem zuzugreifen? Was passiert, wenn Sie den NFS-Dienst wieder starten?

9.5 Diagnose-Werkzeuge

Die allgemeinen Diagnosemittel für Prozesse und RPC-Programme (Software wie `ps`, `netstat` oder `rpcinfo`) haben natürlich auch für NFS Relevanz. Es gibt aber noch einige spezifische Werkzeuge. Damit eine (Schreib-)Operation unter NFS Erfolg hat, müssen drei Bedingungen erfüllt sein.

Server-Dateisystem muss Zugriff gestatten

Erstens muss das zugrundeliegende Dateisystem auf dem Server den Zugriff gestatten. Schlägt der Zugriff trotz anscheinend richtig gesetzter Datei-Rechte fehl, so kann dass an unterschiedlichen UIDs auf Client und Server liegen. Das kommt bewusst gewollt bei `root` zum Tragen, da er im Normalfall als Benutzer `nobody` auf dem Server agiert (Export-Option `root_squash`). Würde dies nicht so behandelt werden, so könnte ein Benutzer, der auf einem NFS-Client `root`-Rechte hat, aber auf dem NFS-Server nur die eines gewöhnlichen Benutzers, sich leicht auch auf letzterem die Administratorprivilegien erschleichen, etwa indem er in einem Verzeichnis auf dem NFS-Server, das für ihn über NFS schreibbar ist, als `root` auf dem Client ein geeignets SUID-`root`-Programm ablegt und es dann unter seiner normalen Benutzerkennung auf dem NFS-Server aufruft.

Exportoptionen müssen stimmen

Zweitens muss das Verzeichnis mit den richtigen Optionen exportiert worden sein. Welche Verzeichnisse gerade exportiert werden, kann auf dem Server mit dem Kommando »`exportfs -v`« überprüft werden. Dies kann auch vom Client aus abgefragt werden, indem Sie »`showmount -e`« gefolgt vom Namen des NFS-Servers aufrufen.

Verzeichnis muss richtig eingehängt sein

Drittens muss das Verzeichnis richtig eingehängt worden sein. Auf dem Client können Sie dies natürlich mit `mount` kontrollieren, oder Sie fragen beim Server mit »`showmount -a`« gefolgt vom Servernamen an, wer gerade bei ihm Dateisysteme eingehängt hat. (Da der Server zustandslos ist, muss das, was der Server zurückliefert, nicht notwendigerweise stimmen.)

RPC-Probleme

Weiterhin kann ein Nichtfunktionieren von NFS auf Problemen mit dem Portmapper beruhen. Eine mögliche Hilfe bietet hier das Programm `rpcinfo`.

Geschwindigkeit

Funktioniert NFS zwar im Prinzip, aber nicht gerade mit berauschender Geschwindigkeit, so kann das an einem anderweitig verursachten Netzwerk-Engpass liegen, der sich mit Programmen wie `iptraf` aufspüren lässt. NFS-interne Statistiken hingegen liefert das Kommando `nfsstat`. Auch Paketsniffer wie `wireshark` können hier gute Dienste leisten.

Uhrzeit

Haben NFS-Server und NFS-Client unterschiedliche Vorstellungen davon, was die aktuelle Uhrzeit ist, so kann es zu Problemen kommen. Verwenden Sie unbedingt etwas wie NTP, um die Zeit zwischen Server und Client(s) zu synchronisieren (z. B.mit `ntpdate` oder `xntpd`).

Übungen

9.14 [!1] Verwenden Sie die Programme `rpcinfo` und `showmount`, um die aktiven RPC-Dienste und die Namen der exportieren Verzeichnisse auf Ihrem NFS-Server anzuzeigen.

9.6 NFS und Sicherheit

Keine Verschlüsselung

Spielen die Vertraulichkeit oder Integrität der Daten eine Rolle, so ist NFS nur bedingt einsetzbar. Dateizugriffe werden nämlich unverschlüsselt übertragen, so dass ein Angreifer nur hinreichend lange den Netzwerk-Verkehr belauschen muss, um sensible Daten aufzuschnappen. Außerdem könnte ein Angreifer die Dateizugriffe verändern, denn es gibt keinen Schutz gegen Manipulation der Daten-Pakete.

Zugriffskontrolle über numerische UID

Zugriff auf Dateien wird vom NFS-Server anhand der numerischen UID gewährt. Eine weitergehende Identitätsprüfung findet nicht statt. Damit es im Normalbetrieb nicht zu Unfällen kommt, müssen Sie dafür Sorge tragen, dass die Zuordnungen von Login-Namen zu UIDs netzwerkweit übereinstimmen.

Tabelle 9.4: Namen der NFS-Komponenten für Zugriffskontrolle

Programm	RPC-Programmname	TCP-Wrapper
`portmap`	portmapper	portmap
`rpc.mountd`	mountd	mountd
`rpc.statd`	status	statd
`rpc.rquotad`	rquotad	rquotad

Im einfachsten Fall kann das heißen, die Kennwortdatei nur auf einem einzigen Rechner zu ändern und manuell, etwa mit `rdist`, auf die anderen zu kopieren. Praktischer ist die Verwendung eines automatischen Verteilungsmechanismus wie NIS; am flexibelsten, wenn auch aufwändigsten der Einsatz eines richtigen Verzeichnisdiensts wie LDAP.

Angriffsmöglichkeiten

Ein Angreifer kann nun ein Paket mit einer falschen UID an den NFS-Server schicken, um Zugriff auf Dateien zu haben, die eigentlich für ihn tabu sind. Zwar akzeptiert der NFS-Server, um genau dieses Vorgehen zu erschweren, nur Pakete, die von einem privilegierten Port ausgegangen sind – jedenfalls wenn das Verzeichnis mit der Option `secure` exportiert wurde, was der Standardfall ist. Aber das Konzept privilegierter Ports ist zum einen nur innerhalb von Unix etabliert und in heterogenen Umgebungen somit irrelevant. Zum anderen heißt das nur, dass der Angreifer auf irgendeinem Rechner `root`-Rechte erlangen muss. Er kann dann einfach mit `su` die Identität eines beliebigen Benutzers annehmen und dessen Dateien auf dem Server lesen oder gar schreiben.

Weil es relativ leicht ist, ein falsche UID vorzuspiegeln, behandelt der NFS-Server normalerweise Anfragen mit der UID 0 (`root`) so, als kämen sie von `nobody`. Genau dies ist die Aufgabe der Option `root_squash`.

Das Schreiben von Dateien unter einer falschen UID kann leicht zu weiteren Angriffen benutzt werden. Werden beispielsweise die Heimatverzeichnisse über NFS eingebunden, so kann ein Angreifer die Datei `~/.ssh/authorized_keys` im Heimatverzeichnis eines Benutzers ergänzen und kann sich dann ohne Kennwort-Abfrage als dieser Benutzer über SSH einloggen.

Zugriff für bestimmte Clients

Neben der Zugriffskontrolle auf Benutzerebene (UID) kann der Zugriff auch für einzelne Clients eingeschränkt werden (über die IP-Adresse). Dies kann individuell für jedes exportierte Verzeichnis geschehen – durch Konfiguration der Datei `/etc/exports` – oder global für alle Zugriffe. Letzteres geschieht durch den TCP-Wrapper, genauer gesagt durch die entsprechenden Bibliotheken. Einige Komponenten von NFS können einen eigenen Eintrag in `/etc/hosts.allow` bzw. `/etc/hosts.deny` haben, den sie auswertet. Die Schwierigkeit liegt allerdings darin, dass RPC-Programmname, Programmname und der Eintrag für den TCP-Wrapper voneinander abweichen können (Tabelle 9.4).

TCP-Wrapper

Portmapper und TCP-Wrapper

Sie können beispielsweise versuchen, den Zugriff auf den Portmapper (Voraussetzung für die Nutzung irgendwelcher NFS-Dienste) auf Rechner im lokalen Netz `192.168.0.0/24` zu beschränken, indem Sie die folgenden Einträge machen:

```
# In Datei /etc/hosts.deny
portmap: ALL

# In Datei /etc/hosts.allow
portmap: 192.168.0.0/255.255.255.0
```

Rechner- oder Domainnamen sind hier nicht erlaubt und auch nicht sinnvoll; sie könnten nämlich RPC-Aktivitäten auslösen und damit wieder den Portmapper betreffen, wodurch sich eine Endlosschleife ergäbe.

Alternativ – wenn Zugriff auf den Portmapper für mehr Rechner benötigt wird, als Verzeichnisse über NFS einbinden dürfen –, können Sie auch den `mountd` über TCP-Wrapper-Regeln absichern.

Die beschriebenen Methoden bringen einen gewissen Zugriffschutz, sind aber leicht zu unterlaufen. Sensible Daten haben im Datenverkehr per NFS also nichts zu suchen; hierfür bieten sich andere Methoden wie AFS oder SSH an.

Auch auf der Clientseite sollten Sie gewisse Sicherheitserwägungen nicht außer Acht lassen. Wer den NFS-Server kontrolliert, aber auf Ihrem NFS-Client nur Benutzer-, keine Systemprivilegien genießt, könnte zum Beispiel versuchen, sich diese zu erschleichen, indem er geeignete SUID-Programme oder Gerätedateien (Stichwort `/dev/sda`) in einem Verzeichnis ablegt, das Ihr Client importiert. Aus diesem Grund sollten Sie NFS-Dateisysteme sicherheitshalber mit den `mount`-Optionen `nodev` und `nosuid` einbinden, damit diese besonderen Datei-Eigenschaften nicht beachtet werden.

Übungen

9.15 [2] Wieso könnten sich während der Bearbeitung von TCP-Wrapper-Regeln für den Portmapper beim Interpretieren von Rechner- oder Domainnamen neue Portmapper-Aktivitäten ergeben?

9.16 [!2] Warum sind Gerätedateien in NFS-exportierten Verzeichnissen ein Problem, wenn der Client kein `nodev` angibt?

9.17 [3] Was sagen Sie zu dem Vorschlag, NFS-Zugriffe über die TCP-Portweiterleitung der Secure Shell zu »tunneln«?

Kommandos in diesem Kapitel

`exportfs` Verwaltet die Liste der exportierten NFS-Dateisysteme `exportfs`(8) 132

`iptraf` Misst den Netzwerkverkehr `iptraf`(1) 134

`mount` Hängt ein Dateisystem in den Dateibaum ein `mount`(8), `mount`(2) 130, 133

`nfsstat` Liefert NFS-Statistikinformationen `nfsstat`(8) 134

`portmap` Liefert TCP/IP-Portnummern für RPC-Dienste `portmap`(8) 127

`quota` Gibt den Kontingentierungs-Status eines Benutzers aus `quota`(1) 130

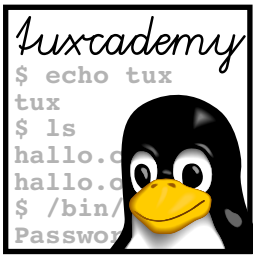
Zusammenfassung

- Eine eingeschränkte Zugriffskontrolle auf den Portmapper und RPC-Programme ist über den TCP-Wrapper möglich.
- Unter Linux haben Sie die Wahl zwischen einer in Form von Daemon-Programmen und einer (teils) im Kernel realisierten Version von NFS.
- NFS ist ein zustandsloser Dienst, d. h., er toleriert den Ausfall und das Wiedererscheinen von Servern. Zustandsbehaftete Operationen wie *file locking* müssen über Zusatzsoftware realisiert werden, die aber Teil gängiger NFS-Pakete ist.
- NFS ist ein zustandsloser Dienst, d. h., er toleriert den Ausfall und das Wiedererscheinen von Servern. Zustandsbehaftete Operationen wie *file locking* müssen über Zusatzsoftware realisiert werden, die aber Teil gängiger NFS-Pakete ist.
- Die Dateisysteme, die ein Server exportieren soll, werden in der Datei `/etc/exports` bezeichnet.
- Clients können Dateisysteme temporär über `mount` oder dauerhaft über einen Eintrag in der Datei `/etc/fstab` einhängen.
- `exportfs` und `showmount` sind nützliche Werkzeuge zur Fehlerdiagnose bei NFS.
- NFS hat keine sehr ausgeprägten Sicherheitseigenschaften. Für die Verteilung sensibler Daten im Netz sollten Alternativen wie AFS oder SSH in Betracht gezogen werden.

Literaturverzeichnis

RFC1014 Inc. Sun Microsystems. »XDR: External Data Representation Standard«, Juni 1987. http://www.ietf.org/rfc/rfc1014.txt

RFC1057 Inc. Sun Microsystems. »RPC: Remote Procedure Call Protocol Specification – Version 2«, Juni 1988. http://www.ietf.org/rfc/rfc1057.txt



10

NFS: Fortgeschrittene Themen

Inhalt

Lernziele

- Den NFS-Automounter kennenlernen
- NFS optimieren können
- Neue Entwicklungen wie NFS-über-TCP oder NFSv4 kennen

Vorkenntnisse

- Grundkenntnisse über NFS und NIS, etwa auf LPIC-1-Niveau (siehe *Linux-Netzwerkadministration I* oder Kapitel 9)

10.1 Der Automounter

10.1.1 Einleitung

In einer Umgebung mit vielen exportierten NFS-Verzeichnissen kann es sehr unübersichtlich und zeitraubend sein, alle NFS-Dateisysteme einzubinden. Abhilfe schafft hier ein »Automounter«, mit dem Sie das tatsächliche Einbinden von Dateisystemen auf den Zeitpunkt verschieben können, wo tatsächlich ein Zugriff auf das betreffende Verzeichnis erfolgt.

Die übliche Vorgehensweise für Linux bedient sich eines Kernelmoduls namens `autofs` (seit Kernel 2.4 `autofs4`, das rückwärtskompatibel ist) sowie eines Daemon-Programms, `automount`.

Der Automounter war ursprünglich eine Idee von Sun Microsystems, und der Linux-Automounter ist in seiner Gedankenwelt und Konfigurationssyntax an den von Sun angelehnt. Außer dem kernelbasierten Automounter für Linux gibt es noch ein (inkompatibles) Programm namens `amd`, das für diverse Unix-Versionen zur Verfügung steht und keine Kernelunterstützung benötigt. Es agiert im Wesentlichen als NFS-Server. `amd` wird in dieser Schulungsunterlage nicht weiter besprochen.

Der Automounter wird üblicherweise mit NFS in Verbindung gebracht, obwohl er genausogut SMB-Dateisysteme, CD-ROMs usw. verwalten kann.

Vorgehensweise

Der Automounter wird für ein oder mehrere Verzeichnisse für zuständig erklärt, zum Beispiel `/home`. Greift anschließend jemand auf eine Datei oder ein Verzeichnis unter `/home` zu, schaut der Automounter in seiner Konfiguration nach, wo er dieses Verzeichnis herbekommen kann, und macht es zugänglich (üblicherweise durch einen `mount`-Aufruf), bevor der eigentliche Dateizugriff fortgesetzt wird. Wird ein so eingebundenes Dateisystem eine gewisse (konfigurierbare) Zeitspanne lang nicht mehr benutzt, hängt der Automounter es selbsttätig wieder aus.

10.1.2 Konfiguration für NFS

Maps

Der Automounter verwendet Konfigurationsdateien, die sogenannten »Maps«. Maps haben einen Eintrag pro Zeile im Format

⟨*Schlüssel*⟩ [-⟨*Optionen*⟩] ⟨*Ortsangabe*⟩

Dabei ist der ⟨*Schlüssel*⟩ konzeptuell ein »Unterverzeichnis« in einem vom Automounter kontrollierten Verzeichnis. Die ⟨*Ortsangabe*⟩ legt fest, wo der Automounter das tatsächliche Dateisystem zum Einbinden finden kann, und die ⟨*Optionen*⟩ werden zum Einbinden benutzt. (Ein konkretes Beispiel kommt gleich.)

Derselbe Automounter kann sich um mehrere Verzeichnisse gleichzeitig kümmern. Beim Start liest das Init-Skript `autofs` eine Datei namens `/etc/auto.master`, die Zeilen der Form

```
/home   /etc/auto.home
/vol    /etc/auto.vol
```

enthält. Dabei steht immer links ein lokales Verzeichnis (das existieren muss) und rechts der Dateiname einer »Map«, die die Details für dieses lokale Verzeichnis liefert.

Der Name »Map« rührt daher, dass der Automounter diese Konfigurationsdateien auch über NIS beziehen kann. In diesem Fall schreiben Sie etwas wie

```
/home    yp:auto.home
```

Sie müssen natürlich arrangieren, dass Ihr NIS-Server eine entsprechende Map anbietet.

`autofs` kann sogar die `auto.master`-Map über NFS abrufen. Dazu muss in der »echten« `/etc/auto.master`-Datei nur ein Eintrag der Form »`+auto.master`« existieren. Alternativ können Sie auch in `/etc/nsswitch.conf` einen Eintrag »`automount: nis`« machen, dann muss lokal überhaupt keine Datei `/etc/auto.master` vorhanden sein.

Beispiel

In einer Map wie `/etc/auto.home` könnten dann Zeilen stehen wie

```
harry    albus.example.com:/home/gryffindor/harry
ron      albus.example.com:/home/gryffindor/ron
draco    albus.example.com:/home/slytherin/draco
```

Wenn der Benutzer `harry` sich anmeldet und das System eine Login-Shell mit dem aktuellen Verzeichnis `/home/harry` startet, sorgt der Automounter dafür, dass das Verzeichnis `/home/gryffindor/harry` vom NFS-Server `albus.example.com` als `/home/harry` zugänglich gemacht wird.

Der Rechner `albus.example.com` muss `/home/gryffindor/harry` nicht als einzelnes Verzeichnis exportieren (das Verzeichnis braucht keinen eigenen Eintrag in `/etc/exports` auf `albus`) – es genügt, wenn dort zum Beispiel `/home` exportiert wird.

`mount`-Optionen

Es ist (wie oben angedeutet) möglich, `mount`-Optionen anzugeben:

```
harry    -rw,rsize=8192,wsize=8192 albus:/home/gryffindor/harry
```

Diese werden an `mount` übergeben, mit zwei Ausnahmen: `fstype=` dient dazu, einen vom Standard `nfs` abweichenden Dateisystemtyp anzugeben, und `strict` sorgt dafür, dass Fehler beim Einbinden von Dateisystemen als fatal angesehen werden. (Die Anwendung dafür ist relativ abgehoben und in `autofs`(5) nachzulesen.)

Optionen in `auto.master`

Sie dürfen `mount`-Optionen auch in Zeilen von `auto.master` hinter den Map-Namen schreiben. Diese Optionen und die in den Zeilen der benannten Maps addieren sich dann auf.

Mehr Map-Tricks

Es gibt noch einige weitere Vereinfachungsmöglichkeiten in Maps: Zum Beispiel können Sie mit dem Schlüssel »`*`« eine Zeile angeben, die auf *alle* Schlüssel passt. Wenn Sie dann noch wissen, dass ein »`&`« in der Ortsangabe durch den Wert des aktuellen Schlüssels ersetzt wird, dann können Sie mit einem Map-Eintrag wie

```
# Datei /etc/auto.homes für Mountpoint /homes
*    &:/home
```

die (exportierten) `/home`-Verzeichnisse aller Rechner im lokalen Netz jeweils als `/home/`⟨*Rechnername*⟩ ansprechen.

Variable

Außerdem können Sie innerhalb von Maps auf eine Reihe von vordefinierten »Variablen« zurückgreifen, die gemäß den Eigenschaften des lokalen Systems gesetzt sind (Tabelle 10.1). Dies ist bequem für »Thin-Clients« ohne Platte, die (fast) alle ihre Dateisysteme über NFS beziehen (!), mit Maps der Form

```
usr   -ro,hard  albus:/var/nfs/usr-$HOST
opt   -ro,hard  albus:/var/nfs/opt-$ARCH
```

Tabelle 10.1: Vordefinierte Variable für `autofs`-Maps

Variable	Bedeutung	Äquivalent
`ARCH`	Architektur	`uname -m`
`CPU`	Prozessortyp	
`HOST`	Rechnername	`uname -n`
`OSNAME`	Betriebssystem	`uname -s`
`OSREL`	Betriebssystem-»Release«	`uname -r`
`OSVERS`	Betriebssystem-»Version«	`uname -v`

(*Achtung*: `OSREL` und `OSVERS` sind wahrscheinlich nicht, was Sie denken.)

Weitere Variable können Sie beim Aufruf von `automount` definieren – die Details sind systemabhängig, aber in der Regel liest das Init-Skript eine Konfigurationsdatei wie `/etc/default/autofs` (Debian GNU/Linux) oder `/etc/sysconfig/autofs` (Red Hat).

SMB-Dateisysteme

Grundsätzlich spricht nichts dagegen, auch SMB-Dateisysteme mit dem Automounter einzubinden. Sie müssen dazu nur als Option »`-fstype=smbfs`« angeben sowie statt der NFS-Serversyntax »`albus.example.com:/home`« die Samba-Freigabesyntax »`//ALBUS/HOME`« verwenden.

lokale Geräte

Dasselbe gilt für lokale Geräte, etwa Laufwerke für Wechselmedien. Mit einem Map-Eintrag wie

```
cdrom   -fstype=iso9660,ro   :/dev/hdc
```

können Sie zum Beispiel ein CD-ROM-Laufwerk unter einem Namen wie `/vol/cdrom` zugänglich machen. In der `auto.master`-Map schreiben Sie dann etwas wie

```
/vol    /etc/auto.vol   --timeout=10
```

Die »`--timeout=10`«-Option setzt die Inaktivitätsfrist auf 10 Sekunden, also wird das CD-ROM-Laufwerk 10 Sekunden nach dem letzten Zugriff automatisch ausgehängt. Welche Zeit Sie hier genau einsetzen sollten, ist debattierbar; wenn Sie innerhalb der Frist den Auswurfknopf drücken, passiert nichts. Bei Diskettenlaufwerken sollten Sie möglicherweise eine noch kürzere Zeit wählen, um ein versehentliches Auswerfen des Mediums im eingehängten Zustand zu vermeiden.

10.1.3 Direkte und Programm-Maps

Man spricht bei der bisher diskutierten Sorte von Konfiguration auch von einer »indirekten Map«. Der Automounter unterstützt auch noch einige andere Arten von Maps, namentlich »direkte Maps« und »Programm-Maps«.

Direkte Maps

In einer direkten Map ist der Schlüssel kein »Unterverzeichnis«, sondern ein kompletter Pfadname, etwa so:

```
# Datei /etc/auto.direct
/nfs/spells/charms      flitwick:/usr/local/charms
/nfs/recipes/potions    snape:/data/potions
```

Der wesentliche Unterschied ist, dass hierbei existierende Verzeichnisse wie `/nfs/spells/transfiguration` oder `/nfs/recipes/powders` sichtbar bleiben. In der `auto.master`-Map muss eine direkte Map als

```
/-     /etc/auto.direct
```

```
#!/bin/bash
PATH=/bin:/sbin:/usr/bin:/usr/sbin
showmount --no-headers -e $1 | sort | \
    awk -v key="$1" -v opts="-fstype=nfs,hard,nodev,nosuid" -- '
    BEGIN { ORS=""; first=1 }
          { if (first) { print opts; first=0 }
            print " \\\n\t" $1, key ":" $1 }
    END   { if (!first) print "\n"; else exit 1 }'
```

Bild 10.1: Shellskript für `/net`-Map

eingebunden werden.

Programm-Map

Eine Programm-Map ist eine Datei, die in `auto.master` als Map benannt worden ist, aber ausführbar ist (typischerweise ein Shellskript). Eine solche Map wird mit dem ⟨*Schlüssel*⟩ als einziges Argument aufgerufen und liefert den tatsächlichen Map-Eintrag für diesen Schlüssel als Standardausgabe. Ist die Ausgabe leer, gilt das als Fehler. Hierbei macht man gerne Gebrauch von der Möglichkeit, in einem einzigen Map-Eintrag mehrere einbindbare Dateisysteme zu benennen, etwa so:

```
albus  -fstype=nfs,hard,nodev,nosuid \
    /home albus:/home \
    /opt  albus:/opt
```

Dieser Map-Eintrag bindet die Dateisysteme …`/albus/home` und …`/albus/opt` ein, wobei »…« für den Mountpoint aus `auto.master` steht.

`/net`-Map

Diese Eigenschaft kann zum Beispiel benutzt werden, um eine Map zur Verfügung zu stellen, die unter einem Mountpoint wie `/net` sämtliche NFS-Server im lokalen Netz mit ihren exportierten Dateisystemen zur Verfügung stellt, und das ohne viel Schreib- oder Wartungsaufwand. Bild 10.1 zeigt ein Shellskript[1], das hierfür als Programm-Map dienen kann. Es bestimmt anhand eines Rechnernamens als Schlüssel mit `showmount` die von diesem Rechner exportierten Dateisysteme und gibt diese, verbrämt mit einem Satz Standardoptionen, als Map-»Wert« zurück.

Übungen

10.1 [!2] Sorgen Sie dafür, dass ein von einem anderen Rechner (etwa von einem anderen Kursteilnehmer) exportiertes NFS-Dateisystem über den Automounter auf Ihrem Rechner eingebunden wird. Verwenden Sie dazu ein Verzeichnis `/auto` als Mountpoint. Was passiert, wenn Sie »`ls /auto`« sagen, bevor Sie auf eine importierte Datei zugegriffen haben? Vergewissern Sie sich auch, dass das Dateisystem nach Ablauf der Inaktivitätsfrist wieder ausgehängt wird. (Für Extrapunkte: Probieren Sie es auch mit einer SMB-Freigabe.)

10.2 [3] Angenommen, in Ihrem Netz haben Sie diverse Linux-Rechner auf Intel-PC- und auf Sun-SPARC-Basis. Sie möchten Ihre lokal installierten Programme – den Inhalt von `/usr/local/bin` – auf einem einzigen Rechner verwalten (der Übersichtlichkeit halber), aber jedem Rechner passend zu seiner Architektur die korrekten Programme zur Verfügung stellen. Wie würden Sie das erreichen?

[1] Das Original ist Bestandteil der `autofs`-Distribution; wir haben es für unsere Zwecke im Interesse der Klarheit etwas abgespeckt.

10.2 NFS-Tuning

NFS ist ein grundlegender Dienst für Workstation-Netze, und Ineffizienzen machen sich dort schnell bemerkbar. Umgekehrt können Verbesserungen am Laufzeitverhalten von NFS einen großen Unterschied für die Arbeit mit dem System machen, vor allem wenn wichtige Daten wie die Heimatverzeichnisse über NFS angesprochen werden. Heutzutage sind schnelle »geswitchte« Ethernetverbindungen mit 100 MBit/s weit verbreitet und die Netzkapazität und physikalische Auslegung nicht mehr so kritisch wie in der Frühzeit von NFS, aber trotzdem gibt es ein paar Punkte, auf die Sie Ihr Augenmerk richten sollten.

Analyse

Grundsätzlich sollten Sie niemals blindlings Änderungen am System vornehmen, um die »Effizienz« zu erhöhen. Vorbedingung ist immer eine Analyse des Ist-Zustands, um sich zu vergewissern, dass tatsächlich ein Problem besteht und dass es tatsächlich an der Stelle ist, wo Sie es vermuten. Im Fall von NFS bedeutet das, dass Sie Messungen des Datendurchsatzes bei Dateien und Verzeichnissen machen sollten, die Sie über NFS ansprechen, etwa mit Programmen wie Bonnie oder IOzone. (Details über diese Programme finden Sie etwa in der Linup-Front-Schulungsunterlage *Linux-Storage*.)

Einfache Tests

Einige einfache Tests können Sie auch schon mit »Bordmitteln« wie `dd` vornehmen. Beispielsweise können Sie einen Moment abwarten, in dem Ihr Netz nicht anderweitig heftig benutzt wird, und ein Kommando geben wie

```
# time dd if=/dev/zero of=/mnt/test bs=16k count=32768
```

um die Zeit zu messen, die für das Anlegen einer 512 MiB großen Datei voller Nullbytes benötigt wird. (Die Datei in diesem Beispiel sollte mindestens zweimal so groß wie der RAM-Speicher des Rechners sein, um störende Effekte durch den Plattencache zu vermeiden.) Analog können Sie das Lesen über NFS testen:

```
# time dd if=/mnt/test of=/dev/null bs=16k
```

Versuchen Sie beides mehrmals hintereinander und hängen Sie das NFS-Dateisystem auf dem Client zwischendurch aus und wieder ein, damit der Plattencache die Ergebnisse nicht verfälscht. (Wenn Sie ganz sicher gehen wollen, dann machen Sie das auch auf dem Server.)

Blockgröße

Der gängigste Parameter, mit dem Sie herumspielen können, um NFS schneller zu machen, ist die Blockgröße, mit der Daten vom Server gelesen bzw. dorthin geschrieben werden. Diese kann getrennt zum Lesen (Mount-Option `rsize`) und Schreiben (`wsize`) gesetzt werden und ist standardmäßig auf 1024 Bytes voreingestellt, was zu niedrig ist. Ein gängiger Wert für Linux, der auch von den üblichen HOWTO-Dokumenten (etwa [NFS-HOWTO]) empfohlen wird, ist 8192. Im Detail hängt der beste Wert (auch) von der Serverplattform ab.

8192 war die traditionelle Obergrenze im NFS-Code der Linux-Kernels bis Version 2.4. Im Zusammenhang mit den Erweiterungen für NFS über TCP wurden jedoch Änderungen eingeschleppt, die die Obergrenze auf 32K gesetzt haben. Theoretisches Maximum in der (semi-)aktuellen NFS-Version 3 sind 64K.

Wenn Sie es genau wissen wollen und Kernel-Quellcode zur Hand haben: Die maximale NFS-Blockgröße wird durch den Parameter `NFSSVC_MAXBLKSIZE` in der Datei `include/linux/nfsd/const.h` festgelegt.

Sie sollten es vermeiden, bei `rsize` und `wsize` eine Blockgröße anzugeben, die größer ist als Ihr `NFSSVC_MAXBLKSIZE`. Tun Sie es trotzdem, können interessante und fremdartige Effekte auftreten.

Wenn Sie `rsize` oder `wsize` auf einen Wert setzen, der größer ist als die MTU Ihres Netzes (bei Ethernet typischerweise 1500), werden die UDP-Datagramme von

Fragmentierung

NFS für die Übertragung vom Sender »fragmentiert« und auf der Empfängerseite wieder zusammengesetzt. Dies braucht auf beiden Seiten eine gewisse Rechenleistung und kann zu Problemen führen, wenn einzelne Fragmente verloren gehen, da dann das komplette Datagramm neu geschickt werden muss. Beobachten Sie in der Datei `/proc/net/snmp` den Eintrag `Ip: ReasmFails`, etwa mit

```
$ grep ^Ip: /proc/net/snmp | cut -d' ' -f17
```

Wenn dieser Wert bei der Übertragung großer Dateien merklich ansteigt, dann liegt etwas im Argen.

Der Kernel reserviert sich einen gewissen endlichen Speicher für unzusammengesetzte Fragmente, dessen Größe in `/proc/sys/net/ipv4/ipfrag_high_thresh` gegeben ist. Ist dieser Speicher voll, werden weitere Fragmente einfach weggeworfen, bis der en Wert erreicht, in `/proc/sys/net/ipv4/ipfrag_low_thresh` enthaltene Wert unterschritten wird.

Synchroner oder asynchroner Export?

Ebenfalls Einfluss auf die NFS-Leistung (jedenfalls aus der Sicht der Clients) hat die serverseitige Einstellung, ob Verzeichnisse »synchron« oder »asynchron« exportiert werden (über die Optionen `sync` bzw. `async` bei `exportfs` oder in `/etc/exports`). »Synchron« heißt dabei, dass der NFS-Server bei Schreiboperationen erst dann Erfolg an den Client meldet, wenn die Operation tatsächlich auf der Platte umgesetzt wurde; »asynchron« dagegen bedeutet, dass der Server die Operation ans lokale Dateisystem übergibt, ohne auf deren Erledigung zu warten, und dem Client vorlügt, alles wäre bereits vom Tisch. Aus der Sicht eines Clients sind Zugriffe auf ein »asynchron« eingebundenes Dateisystem merklich schneller, aber falls der Server aus irgendwelchen Gründen abstürzt, bevor er die Operationen tatsächlich auf der Platte amtlich machen konnte, können Daten verloren gehen. Aus Kompatibilitätsgründen ist die Voreinstellung heutzutage `sync` (früher war es `async`), aber wenn Sie gerne gefährlich leben oder ein Journaling-Dateisystem wie `ext3` verwenden, wo Sie Metadaten- und Datenzugriffe durchs Journal laufen lassen (»`data=journal`«), können Sie auch `async` verwenden.

Im wirklichen Leben ist die Situation etwas komplizierter: Das synchrone Verhalten gilt wie oben beschrieben für die NFS-Protokollversion 2; beim heute gängigen NFSv3 wartet der Server auch bei synchronem Export nicht auf die tatsächliche Erledigung der Operation, sondern signalisiert dem Client, welche Daten auf der Platte gelandet sind und welche dieser noch weiter cachen muss. Clients können Dateien in einem besonderen »synchronen« Modus öffnen, in dem die Zugriffe für die betreffenden Dateien dann »wirklich« synchron ausgeführt werden. Clients haben auch die Möglichkeit, explizit vom Server die Erledigung aller ausstehenden Operationen zu fordern, und in diesem Fall wartet der Server bei synchronem Export tatsächlich auf eine positive Rückmeldung von der Platte. Bei asynchronem Export lügt der Server auch hier.

Einige Firmen bieten »NFS-Beschleuniger« in Hardware an, die Schreibzugriffe in nichtflüchtigem RAM puffern, das aussieht wie ein Teil des Plattensubsystems. Damit entfällt die Wartezeit bei synchronen Zugriffen im Wesentlichen. – Moderne Journaling-Dateisysteme wie `ext3` erlauben es auch, das Journal auf einer anderen Platte unterzubringen als das dazugehörige Dateisystem. Diese »Platte« kann natürlich ein beliebiges Blockgerät sein, also zum Beispiel auch Flash-Speicher.

Netzwerkkarten und Treiber

Als UDP-basierter Dienst ist NFS davon abhängig, dass die Netzinfrastruktur große IP-Datagramme korrekt übertragen kann. Manche Netzwerkkarten und Treiber haben damit Probleme. Testen können Sie das mit `ping`, etwa wie

```
# ping -f -s 16384 nfsserver.example.com
```

(die Option `-f` sorgt dafür, dass `ping` Pakete so schnell schickt, wie es kann, während `-s` die Paketgröße auf 16 KiB setzt – einen für NFS realistischeren Wert als die üblichen 64 Bytes). Wenn bei diesem Test Pakete verloren gehen oder lange brauchen, haben Sie ein Problem.

`nfsstat` Für detaillierte Analysen des NFS-Protokolls im besonderen gibt es das Programm `nfsstat`. `nfsstat` mit der Option »`-o net`« erlaubt unter anderem eine genaue Beobachtung der wiederholten Übertragung von Datagrammen. Wenn viele Datagramme mehrmals übertragen werden müssen – sei es aufgrund von Netzwerk-Engpässen, überlasteten Servern usw. – ist das tödlich für die NFS-Leistung.

Sollte es laut `nfsstat` tatsächlich öfters zu Wiederholungen kommen, können Sie versuchen, über die `mount`-Optionen `retrans` und `timeo` die Situation zu verbessern.

Option `timeo` `timeo` erlaubt es, die Zeit (in Zehntelsekunden) anzugeben, nach der eine Operation wiederholt wird. Wenn Sie dieses Intervall erhöhen, werden zwar einzelne Operationen später neu gestartet, aber unter Umständen steigern Sie so den Gesamtdurchsatz, da in einem beschäftigten Netz die Antworten vom Server eher eine Chance haben, beim Client anzukommen, bevor dieser sie abschreibt.

Eine Operation wird zuerst nach `timeo` Zehntelsekunden wiederholt. Danach wartet der NFS-Client die doppelte Zeitspanne ab, bevor er es wieder versucht. Die Zeitspanne wird sukzessive weiter verdoppelt, bis entweder eine Antwort ankommt, das Maximum von 60 Sekunden erreicht wird oder die im Parameter `retrans` angegebene Anzahl von Wiederholungen erreicht wurde (normalerweise 3). Tritt einer der letzteren Fälle ein, spricht man von einem *major timeout*, den das System durch die Fehlermeldung "*NFS server … not responding*" signalisiert.

Tritt ein *major timeout* auf einem mit der Option `hard` (der Standardfall) eingebundenen Dateisystem auf, dann versucht der NFS-Client es von neuem beginnend mit dem doppelten Wert von `timeo`. Im Falle von `soft` wird dem Programm, das die Schreib- oder Leseoperation abgesetzt hat, ein Fehler signalisiert.

Anzahl der NFS-Daemon-Threads Einen möglichen Einfluss auf die Effizienz des NFS-Diensts hat auch die Anzahl der NFS-Daemon-Threads im System. Diese wird traditionell auf 8 gesetzt. Auf einem sehr gefragten Server kann es sinnvoll sein, diesen Wert zu erhöhen (eingestellt wird er typischerweise über das Init-Skript für den NFS-Server beziehungsweise eine Datei, die dieses liest). Sie sollten mindestens so viele `nfsd`-Threads verwenden, wie Ihr Rechner Prozessoren hat; eine bessere Faustregel sind wahrscheinlich 4–8 Threads pro Prozessor. Über die Auslastung der Threads gibt

`/proc/net/rpc/nfsd` die Datei `/proc/net/rpc/nfsd` Aufschluss.

Übungen

10.3 [!3] Messen Sie (wie beschrieben) die Übertragungszeit einer großen Datei bei verschiedenen NFS-Blockgrößen (etwa 1, 8 und 32 KiB). Stellen Sie einen Unterschied fest?

10.4 [3] Vergleichen Sie (mit einer geeigneten festen Blockgröße) die Geschwindigkeit von NFS im »synchronen« und »asynchronen« Betrieb. Berücksichtigen Sie dabei auch die Eigenschaften des unterliegenden Dateisystems, etwa indem Sie den synchronen Betrieb auf einem `ext3`-Dateisystem mit oder ohne Journaling ausprobieren.

10.3 NFS über TCP

Neuerdings können Sie NFS nicht nur über UDP, sondern auch über TCP transportieren (Ihr Client und Ihr Server müssen mitspielen – für die gängige Linux-

Implementierung kein Problem). NFS über TCP hat Vorteile, wenn die Netzwerkverbindung zwischen Client und Server schlecht ist, da nur einzelne Pakete neu übertragen werden müssen, statt komplette RPC-Transaktionen zu wiederholen. Außerdem geht TCP eleganter mit dem Fall um, dass zwischen Sender und Empfänger verschiedene Wegstrecken unterschiedlicher Kapazität liegen.

Beide Situationen sollten in einem lokalen Netz eigentlich nicht eintreten, so dass NFS über TCP sich vor allem für den Weitverkehr zu empfehlen scheint. Wegen der mangelhaften Sicherheitsvorkehrungen in NFS sollten Sie allerdings dringend davon Abstand nehmen, einen ungeschützten NFS-Server dem Internet auszusetzen. Eine geeignete VPN-Umgebung ist hier zwingend nötig.

Eingeschaltet wird NFS über TCP mit der `mount`-Option »`-o tcp`«.

10.4 NFSv4

NFS ist, wie gezeigt, ein einfaches Protokoll, das durchaus seine Macken hat. Aus der Not heraus, mit SMB/CIFS und anderen Systemen wie AFS und seinen Nachfolgern konkurrieren zu müssen, wurde eine gründliche Revision von NFS in Angriff genommen – nicht mehr von Sun Microsystems, sondern der IETF. Das Ergebnis ist ein System – NFS Version 4 oder kurz »NFSv4« –, das mit herkömmlichem NFS vor allem den Namen gemeinsam hat.

NFSv4 bietet im Gegensatz zu früheren Versionen die folgenden Eigenschaften:

- »Traditioneller« Dateizugriff (wie bisher)
- Integrierte Unterstützung für Dateisperren (*file locking*) und das Mount-Protokoll (war vorher separat)
- »Kombinierte« Operationen im Protokoll und Caching im Client (für Effizienz)
- Starke Sicherheit (etwa mit Kerberos)
- Sinnvolle Funktion im Internet
- Internationalisierung

NFSv4 ist standardisiert in [RFC3530].

Für Linux existieren sowohl Client- als auch Server-Implementierungen von NFSv4. Diese sind im Linux-Kernel 2.6 enthalten und weitgehend funktionsfähig. Eine ausführlichere Diskussion würde den Rahmen dieser Schulungsunterlage sprengen.

Kommandos in diesem Kapitel

`automount` Hängt entfernte oder lokale Dateisysteme bei Bedarf ein
`autofs`(5), `automount`(8), `autofs`(8) 140

`nfsstat` Liefert NFS-Statistikinformationen `nfsstat`(8) 146

Zusammenfassung

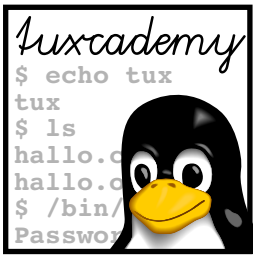
- Der Automounter (`autofs`) gestattet das automatische Einbinden von NFS- oder SMB-Dateisystemen (oder lokalen Geräten) bei Bedarf.
- Die Leistung einer NFS-Installation hängt von verschiedenen Faktoren ab, etwa der (konfigurierbaren) Blockgröße für die Übertragung, anderen Konfigurationsparametern des Systems sowie der Qualität der Netzwerkumgebung.
- NFS kann über TCP betrieben werden, was prinzipiell bessere Leistung im Weitverkehr ermöglicht.
- NFSv4 ist ein neues Protokoll, das mit den wesentlichen Mängeln früherer Versionen aufräumt, was Sicherheit und Effizienz angeht.

Literaturverzeichnis

NFS-HOWTO Tavis Barr, Nicolai Langfeldt, Seth Vidal, et al. »Linux NFS-HOWTO«, August 2002. http://nfs.sourceforge.net/nfs-howto/

RFC3530 S. Shepler, B. Callaghan, D. Robinson, et al. »Network File System (NFS) version 4 Protocol«, April 2003. http://www.ietf.org/rfc/rfc3530.txt

SEL01 Hal Stern, Mike Eisler, Ricardo Labiaga. *Managing NFS and NIS*. Sebastopol, CA: O'Reilly & Associates, 2001, 2. Auflage. ISBN 1-56592-510-6. http://www.oreilly.com/catalog/nfs2/



A

Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

1.3 Bevor der DHCP-Server einem Client eine dynamische Adresse anbietet, versucht er sich zu vergewissern, dass diese Adresse tatsächlich noch nicht anderweitig in Benutzung ist. Dazu verschickt er einige ARP- und ICMP-`ping`-Anfragen, auf die er aber nicht mit einer positiven Antwort rechnet. (Sollte trotzdem eine kommen, dann bricht er den Dialog mit dem Client ab.)

1.6 Der Client sollte, wenn er Parameter für 60 Sekunden bekommt, nach 30 Sekunden anfangen, sich um eine Verlängerung zu bemühen.

2.1 Beispielsweise könnten Sie Zugriff auf das CD-ROM-Laufwerk oder die Soundkarte eines Rechners aus Sicherheitsgründen auf den Benutzer beschränken wollen, der direkt vor dem betreffenden Rechner sitzt. Wenn Sie dafür sorgen, dass der Benutzer nur (zum Beispiel) in die Gruppe `audio` aufgenommen wird, wenn er sich über einen grafischen Displaymanager anmeldet, können Sie das erreichen.

2.2 Bei der Konfiguration über `/etc/pam.d` kann jedes Programm in seinem Distributionspaket eine eigene Datei mit seiner PAM-Konfiguration mitbringen. Das macht es viel einfacher, Pakete zu installieren und (vor allem) wieder zu entfernen, als wenn man Zeilen an die systemweite `/etc/pam.conf`-Datei anhängen und dort wieder entfernen müßte.

2.3 Dies ist eine Aufgabe für das PAM-Modul `pam_limits`. Fügen Sie in die Datei `/etc/security/limits.conf` eine Zeile ein wie

```
hugo - nproc 10
```

und arrangieren Sie, dass das Modul `pam_limits` als Teil der Anmeldeprozedur aufgerufen wird.

2.4 Sie können existierende Kennwörter bekanntlich nicht entschlüsseln und neu verschlüsseln. Sie müssen also zuerst dafür sorgen, dass *neu eingegebene* Kennwörter passend verschlüsselt werden (etwa indem Sie das gewünschte Verfahren als Parameter von `pam_unix` angeben) und anschließend alle Benutzer dazu zwingen, ihr Kennwort zu ändern, zum Beispiel über die Parameter zur Kennwort-Alterung in `/etc/shadow`.

2.5 Versuchen Sie etwas wie

```
auth required pam_listfile.so  onerr=fail item=user \
                               sense=allow \
                               file=/etc/sshd/sshusers
```

Wenn Sie feststellen, dass dies verblüffend dem zweiten Beispiel aus `pam_listfile`(8) ähnelt: Gut für Sie! – Sollten Sie etwas hier auf einem Rechner verwenden wollen, den Sie nur über die `ssh` erreichen können, dann müssen Sie übrigens aufpassen, dass Sie sich nicht selber aussperren: Sie sollten sicherstellen, dass entweder `root` (falls Sie direktes Anmelden als `root` erlauben) oder ein Benutzer, von dem aus Sie über `su` oder `sudo root` werden können, in der Datei steht.

2.6 Es könnte sein, dass Sie nicht `pam_unix` verwenden, sondern ein ganz anderes Anmeldeverfahren. In diesem Fall müssen Sie das Ganze nicht nochmal implementieren, sondern können auf `pam_pwhistory` zurückgreifen. *Wichtig:* Denken Sie an die im Haupttext gemachte Einschränkung bezüglich LDAP und so.

2.7 Führen Sie eine Grenze ein (zum Beispiel 10 Fehlversuche) und verwenden Sie das Programm `pam_tally`, um die Anzahl von Fehlversuchen für den betreffenden Benutzer auf einen irrwitzig hohen Wert zu setzen.

2.8 Schreiben Sie in die Datei `/etc/pam.d/su` etwas wie

```
# root darf alles
auth  sufficient  pam_rootok.so
# Mitglieder von wheel dürfen auch alles
auth  sufficient  pam_wheel.so  trust
# Der Rest der Welt darf nichts
auth  required    pam_deny.so
```

(Wenn Sie solche Sachen testen, sollten Sie eine Shell in der Hinterhand haben, wo Sie schon als `root` angemeldet sind.)

4.1 Hierzu können wir leider nichts sagen …

4.2 Auch hierzu können wir leider keine Musterlösung abdrucken. Versuchen Sie mal Kommandos wie »`dpkg -s samba`« (auf Debian-artigen Systemen) oder »`rpm -qi samba`« (bei Systemen wie Red Hat, Fedora oder SUSE).

4.4 Mehr über den Aufbau von Init-Skripten erfahren Sie zum Beispiel aus den Linup-Front-Schulungsunterlagen *Systemadministration II – Kernel und Shell* oder *Linux-Systemanpassungen*. Viele Distributionen greifen auf einen Satz vorgekochter Funktionen zurück, die das Erstellen von Init-Skripten im allgemeinen erleichtern sollen. Schauen Sie nach, ob das bei Ihrer Distribution auch so ist. Enthält Ihre Distribution spezielle Vorkehrungen zum automatischen Anlegen der Runlevel-spezifischen Links?

4.5 Samba setzt voraus, dass das Netz initialisiert ist (sonst kann es keine Dienste anbieten). Eventuell muss auch der Syslog-Dienst zur Verfügung stehen sowie andere Dienste, die der Samba-Server benötigt (etwa DNS und LDAP). Dienste, die direkt von Samba abhängen, gibt es eigentlich nicht; je nachdem, wie in Ihrem System die Benutzerverzeichnisse abgespeichert werden, kann es natürlich nötig sein, dass Samba auf Rechner *A* laufen muss, bevor Sie sich auf Rechner *B* erfolgreich anmelden können (etwa wenn Ihr Heimatverzeichnis auf *A* liegt und *B* es zum Beispiel über PAM beim Anmelden einbinden möchte).

4.6 Versuchen Sie ein Kommando wie

```
$ grep -v '^#' /etc/samba/smb.conf | grep -v '^$'
```

In Abschnitt 4.9 werden Sie außerdem das `testparm`-Kommando kennenlernen, das (unter anderem) etwas sehr Ähnliches tut.

6.2 Angenommen, Ihre Arbeitsgruppe heißt `HOGWARTS`. Ein mögliches Kommando wäre dann

```
# nmblookup HOGWARTS#03
```

(alle Mitglieder der Arbeitsgruppe `HOGWARTS` registrieren den Namen `HOGWARTS#03`).

6.3 Suchen Sie nach Ihrem Arbeitsgruppennamen gefolgt von `<1D>` bzw. `<1B>`, also zum Beispiel

```
$ nmblookup HOGWARTS#1d                    bestimmt LMB
$ nmblookup HOGWARTS#1b                    bestimmt DMB
```

Um die Plattform herauszufinden, lassen Sie sich die Samba-Server aufzählen (siehe Abschnitt 6.5) und vergleichen Sie diese Liste mit den gefundenen Suchdienst-Rechnern.

6.6 Ein Samba-Server nimmt standardmäßig an der Schönheitskonkurrenz um den LMB-Posten teil und gewinnt diese auch durch seinen Standard-OS-Level von 20, sofern kein NT-Domänencontroller ihn übertrumpft. Damit ist der Samba-Server der LMB, aber er weiß ohne besondere Konfiguration nicht, welcher Rechner als DMB fungiert, kann diesen also nicht mit aktuellen Informationen aus dem lokalen Subnetz versorgen.

7.1 Verwenden Sie zum Beispiel »`hosts deny = 192.168.0.5`«, wenn der betreffende Rechner die IP-Adresse `192.168.0.5` hat.

7.2 Angenommen, Ihr lokales Netz verwendet die Adressen `10.11.12.0/24`. Dann wäre etwas wie

```
hosts deny = 10.11.12.0/24 EXCEPT 10.11.12.13
```

zielführend. Im Text stehen noch andere Möglichkeiten.

7.3 Die Parameter »`hosts allow`« und »`hosts deny`« sind eigentlich freigabespezifische Parameter. Wenn sie im `global`-Abschnitt auftauchen, definieren sie im Grunde Standardwerte für alle Freigaben, was einer globalen Einstellung entspricht.

7.4 Siehe die Lösung zur vorigen Aufgabe – ein »`hosts deny`« im `global`-Abschnitt definiert einen Standardwert für alle Freigaben, wo keine spezielleren Einstellungen gemacht werden. Etwas wie

```
[global]
   ◁◁◁◁
   hosts deny = 192.168.0.5
   ◁◁◁◁
[test]
   ◁◁◁◁
   hosts deny =
   ◁◁◁◁
```

sollte also dem Rechner mit der IP-Adresse `192.168.0.5` den Zugriff auf alle Freigaben außer `test` verbieten.

7.5 Die Unix-Zugriffsrechte haben, wie im Text beschrieben, Vorrang gegenüber den Einstellungen im Samba-Server. Das liegt daran, dass Samba auf die Dateien mit den Rechten zugreift, die dem Benutzer zustehen, als der der Client sich mit der Freigabe verbunden hat (bisher vor allem `nobody`). Eine für den zugreifenden Benutzer auf der Unix-Ebene nicht lesbare Datei auf einer eigentlich als »lesbar« gekennzeichneten Freigabe wird also von Samba nicht als lesbar zugänglich gemacht. – Kommen Sie auf diese Aufgabe zurück, wenn Sie die Konfigurationsparameter in Abschnitt 7.3 zur Kenntnis genommen haben und experimentieren Sie noch etwas weiter.

7.7 Die Lösung ist ein Parameter der Form »`veto files = *.txt`«.

7.9 Versuchen Sie es mal mit dem Parameter »`min password length`«

7.11

```
for i in $(grep ':x:[0-9][0-9][0-9].:' /etc/passwd | cut -f 1 -d ':')
do
    if [ $i -gt 499 ]
    then
        echo -e "password\npassword" | smbpasswd -as $i
    fi
done
```

9.2 Die flexible Zuordnung macht es fast unmöglich, Regeln für einen Paketfilter aufzustellen, der NFS-Daten entweder gezielt durchläßt oder gezielt sperrt. Natürlich sollten Sie niemals einen NFS-Server direkt am Internet betreiben (auch nicht mit einem Firewall oder Paketfilter davor), aber Sie könnten ja auf die Idee kommen, im internen Netz einen Paketfilter zwischen einem NFS-Server und (manchen seiner) Clients zu schalten.

9.5 RPC-Programme registrieren sich nur einmal beim Portmapper. Wird dieser beendet und neu gestartet, so müssen sie sich bei ihm neu anmelden: alle RPC-Programme müssen ebenfalls neu gestartet werden.

9.7 Für lokale Zugriffe werden die Einträge in `/etc/hosts.allow` und `/etc/hosts.deny` ignoriert. Der entfernte Zugriff wird durch

```
portmap: 192.168.0.42
```

o. ä. ermöglicht; die Angabe eines Rechnernamens statt der IP-Adresse reicht nicht.

9.8 Zu den Vorteilen des Kernel-NFS-Daemons gehört zuerst die höhere Effizienz (bei Dateisystemen eine willkommene Eigenschaft). Nachteilig sind die höhere Komplexität der Lösung sowie die Tatsache, dass ein Defekt im Kernel-NFS-Code potentiell das komplette System zum Absturz bringen kann, was bei Code auf Benutzerebene eigentlich nicht passieren sollte. Zur Ehrenrettung des Linux-Kernel-Mode-NFS lässt sich sagen, dass es sich dabei um ein ziemlich solides Produkt handelt; es ist also nicht damit zu rechnen, dass gravierende Fehler auftreten.

9.11 Versuchen Sie etwas wie

```
#  mount -t nfs -o hard albus:/tmp /mnt
```

(»`-o hard`« ist tatsächlich die Voreinstellung, und »`-t nfs`« folgt aus dem Namen des einzuhängenden Verzeichnisses; die beiden können also genausogut weggelassen werden.)

9.12

```
albus:/tmp  /mnt  nfs  hard 0 0
```

(Betrachten Sie z. B. die Linup-Front-Schulungsunterlagen *Systemadministration I* und *Netzwerkadministration I* für die Details. Siehe auch `nfs`(5).)

9.13 Wenn der NFS-Server nicht erreichbar ist, wird beim üblichen »harten« Einhängen der Client so lange aufgehalten, bis der NFS-Server wiederkommt (notfalls bis zum Sankt-Nimmerleins-Tag). Im Falle des »weichen« Einhängens (Option `soft`) gibt es nach einer Weile eine Zeitüberschreitung und die Lese- oder Schreiboperation wird mit einer Fehlermeldung abgebrochen.

9.15 Es könnte sein, dass in Ihrem System `/etc/hosts` über NIS zugänglich gemacht wird. In diesem Fall involviert das Auflösen eines Rechnernamens NIS und mithin den Portmapper.

9.16 Gerätedateien werden immer im Kontext des Rechners interpretiert, der die Datei anschaut (nicht notwendigerweise der, wo die Datei tatsächlich gespeichert ist). Dabei ist der tatsächliche Dateiname irrelevant; für Schindluder interessant sind nur die Gerätenummern, der Eigentümer, die Gruppe und die Zugriffsrechte. Das »Gerät« mit den Gerätenummern (1, 2) und dem konventionellen Namen `/dev/kmem` erlaubt beispielsweise das Lesen und auch Schreiben des Kernelspeichers – für einen Angreifer ein gefundenes Fressen, da er, geeignete Zugriffsrechte und etwas Wissen über die Interna des Linux-Kernels vorausgesetzt – die Prozesstabelle suchen und die effektive UID seines Shellprozesses auf 0, also `root`, setzen kann (nur mal so als Beispiel). Das funktioniert aber nicht nur mit `/dev/kmem`, sondern mit jeder zeichenorientierten Gerätedatei, die die Gerätenummern (1, 2) hat, egal ob sie in `/dev` steht und nur für `root` schreibbar ist, oder ob sie `/home/draco/malfoy-manor.jpg` heißt mit Modus 0600 und Eigentümer `draco`.

9.17 Grundsätzlich ist das eine nette Idee, die nur an ein paar kleinen praktischen Problemen leidet. Zunächst verwendet NFS standardmäßig UDP und nicht TCP, und die SSH kann nur TCP-Verbindungen weiterleiten. NFS über TCP ist möglich, aber nicht sonderlich verbreitet. Bei der Installation müssen Sie die NFS-Daemons auf feste TCP-Portnummern einschwören, und das ganze hat etwas weiterreichende Sicherheitsimplikationen, die Sie am besten [NFS-HOWTO] entnehmen. Ferner kann das Tunneln von TCP-Verbindungen über TCP zu »interessanten« Effekten führen, was das Timing der Übertragung angeht, was im Falle von NFS Paketnachforderungen und möglicherweise Ineffizienzen auslösen kann. – Alles in allem ist ein richtiges VPN, etwa auf der Basis von OpenVPN, eher zu empfehlen. Mehr Details finden Sie zum Beispiel in der Linup-Front-Schulungsunterlage *Linux-Sicherheit*.

10.1 Sie brauchen eine `/etc/auto.master`-Datei mit einem Inhalt wie

```
/auto     /etc/auto.auto
```

und die tatsächliche Map mit etwas wie

```
otherhost   otherhost:/tmp
```

(oder was auch immer der andere Rechner exportiert). Wenn Sie auf `/auto/otherhost` zugreifen, sollten Sie (ohne explizites `mount`) den Inhalt von `/tmp` auf dem Rechner `otherhost` sehen. Ein »`ls /auto`« ganz am Anfang sollte nichts anzeigen.

10.2 Im einfachsten Fall installieren Sie auf den Intel-Rechnern eine Datei `auto.direct` mit dem Inhalt

```
/usr/local/bin  -ro,hard  server:/usr/local/bin-intel
```

und auf den SPARC-Rechnern eine analoge Datei `auto.direct` mit etwas wie

```
/usr/local/bin  -ro,hard  server:/usr/local/bin-sparc
```

Diese binden Sie jeweils als direkte Map ein.

Bequemer ist es, die vordefinierte Variable `$ARCH` zu verwenden, da Sie nur eine Map mit dem Inhalt

```
/usr/local/bin  -ro,hard  server:/usr/local/bin-$ARCH
```

verwenden müssen (vergewissern Sie sich, was »`uname -m`« bei den betreffenden Plattformen liefert).

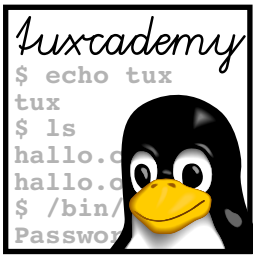
Wenn Sie ohne die direkte Map auskommen wollen, können Sie tricksen: Installieren Sie statt dessen in `auto.master`

```
/arch   auto.arch  -ro,hard
```

mit der Map `auto.arch` wie folgt:

```
bin    server:/usr/local/bin-$ARCH
```

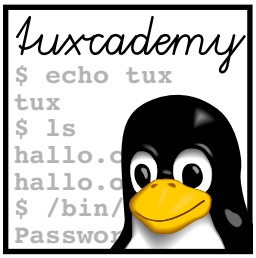
Dann brauchen Sie nur noch ein symbolisches Link von `/usr/local/bin` nach `/arch/bin`. (Derselbe Trick funktioniert auch für `lib`.)



B

Kommando-Index

Dieser Anhang fasst alle im Text erklärten Kommandos zusammen und verweist auf deren Dokumentation sowie die Stellen im Text, wo die Kommandos eingeführt werden.



Index

Dieser Index verweist auf die wichtigsten Stichwörter in der Schulungsunterlage. Besonders wichtige Stellen für die einzelnen Stichwörter sind durch **fette** Seitenzahlen gekennzeichnet. Sortiert wird nur nach den Buchstaben im Indexeintrag; „`~/.bashrc`“ wird also unter „B“ eingeordnet.