Kursskript

Volker Lendecke
Service Network GmbH
Göttingen
http://www.SerNet.DE/
http://samba.SerNet.DE/

21. Dezember 2001

Dieses Dokument ist eine Mitschrift des Sambakurses der Service Network GmbH in Göttingen. Es gibt einen guten Überblick über den Kurs und kann gleichzeitig als generelle Einführung in NetBIOS und Samba dienen.

Inhaltsverzeichnis

1 Einführung 4

2 Eine einfache Konfiguration 5

3 NetBIOS 7

4 Bestandteile von Samba 11

5 NetBIOS-Konfiguration mit Samba 12

6 Namensauflösung per Broadcast 13

7 Netzwerkumgebung 15

8 NetBIOS über Subnetzgrenzen 18

9 Windows-Namensauflösung im Detail 21

10 Browsing über Subnetzgrenzen 23

11 Virtuelle Sambaserver 26

12 Browsing im WAN – schneller 29

13 Einfache Freigaben 33

14 SMB-Sitzungen 34

15 Rechte an Freigaben 37

16 Zugriffsrechte im Dateisystem 39

17 Projektverzeichnisse, zum zweiten 43

18 ACLs 45

19 oplocks 49

20 Paßwörter 50

21 Druckfreigaben 57

22 Windows NT Domänen 58

23 Samba als Primary Domain Controller 59

24 Profile, Logon Scripts und Policies 62

25 Samba als Domänenmitglied 63

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

26 winbind **66**

27 smbcontrol **69**

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

1 Einführung

Samba – Was ist das?

Kurz gesagt läßt Samba jeden Unixrechner in der Netzwerkumgebung von Windows erscheinen. Das heißt, man kann von Windows aus auf einen Unixrechner genau wie auf einen anderen Windowsrechner zugreifen. Der Clientrechner merkt gar nicht, daß er es nicht mit einem echten Windowsserver zu tun hat. Im Detail bedeutet das, daß sehr einfach Dateifreigaben erstellt werden können. Jeder Benutzer kann transparent Dateien auf seinem Heimatverzeichnis unter Unix und in anderen freigegebenen Verzeichnissen ablegen. Weiterhin kann man Drucker, die unter Unix ansprechbar sind, als Netzwerkdrucker in Windows ansprechen. Darüber hinaus bietet Samba viele Dienste, die sonst nur von Windows NT geleistet werden. Dazu gehören:

WINS-Server Mit Samba kann sehr einfach ein WINS-Server eingerichtet werden. Dieser stellt Namensdienste für NetBIOS-Netze zur Verfügung, damit sich Windows-Maschinen über Subnetzgrenzen hinweg erreichen können

Computersuchdienst Samba als sehr stabiler Server kann alle Aufgaben des Computersuchdienstes übernehmen. Die in Windowsumgebungen oft nicht sehr vorhersagbare Netzwerkumgebung kann so etwas stabiler gemacht werden.

Logon Server Für Windows-95/98 ist Samba Logon-Server, kann also die Domänenanmeldung für diese Systeme übernehmen.

PDC Die Funktionalität des echten Primary Domain Controller ist nicht vollständig implementiert. Für viele Anwendungszwecke, insbesondere Authentifizierung von NT-Workstationbenutzern, reicht Samba jedoch völlig aus.

Diagnosewerkzeuge Samba bietet eine Reihe von kleinen, aber sehr effektiven Werkzeugen, die die oft mühselige Suche nach Fehlern im Netz vereinfachen können.

Samba bietet gegenüber anderen Implementationen des SMB-Protokolls einige Vorteile. Teilweise sind diese Vorteile von Unix geerbt, teilweise sind sie in der Architektur von Samba begründet.

Entfernte Administration Der größte Vorteil von Samba in größeren Umgebungen ist die Möglichkeit, die gesamte Administration von der Kommandozeile aus durchzuführen. Damit bekommt man gegenüber grafischen Oberflächen sehr viel bessere Möglichkeiten, von entfernten Standorten aus zu administrieren. Werkzeuge wie PC Anywhere sind hier deutlich weniger flexibel.

Zusätzlich bietet Samba die Möglichkeit der grafischen Administration über einen Webbrowser. Auch hier ist es unerheblich, wo sich Administrator und Server befinden.

Zentrale Konfiguration Die gesamte Konfiguration von Samba befindet sich in einer einzigen Datei und ist nicht über viele Dialogfelder verteilt. Das erleichtert die Administration erheblich. So läßt sich eine funktionierende Konfiguration sehr einfach sichern und wieder einspielen.

Stabilität Samba erbt von Unix eine hohe Stabilität. Unixrechner sind dafür ausgelegt, über Monate hinweg durchzulaufen und leisten dies auch. Samba als weiterer Prozeß profitiert von dieser hohen Verfügbarkeit. Die modulare Struktur von Unix läßt es darüber hinaus zu, daß der Serverdienst Samba unabhängig von allen anderen Systemprozessen eigenständig neu gestartet werden kann, sofern hier ein Problem vorliegen sollte.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Samba hat eine Architektur, die die Stabilität weiter fördert. Für jede Clientverbindung wird ein eigener Prozeß gestartet. Verursacht also ein Client ein Problem auf Serverseite, wird möglicherweise der für diesen Client zuständige Prozeß abstürzen. Die anderen Prozesse und damit Clients werden nicht gestört.

Skalierbarkeit Samba kann von dem vielzitierten kleinen 386er unter Linux bis hin zu den größten heute verfügbaren Maschinen jede Hardware optimal ausnutzen. Die Architektur von Samba ermöglicht es, daß auch Multiprozessormaschinen ausgelastet werden. Multiprozessormaschinen können alle Prozessoren dann beschäftigen, wenn es viele unabhängige Prozesse im System gibt. Samba erstellt für jeden Client einen Prozeß, der auf einem eigenen Prozessor ablaufen kann.

Flexibilität Samba bietet eine riesige Anzahl von Konfigurationsoptionen, die zunächst einmal überwältigend wirkt. Im Laufe des Kurses wird sich herausstellen, daß für das Funktionieren von Samba nur sehr wenige Optionen wirklich notwendig sind. Die meisten Optionen werden nur für Spezialfälle benötigt, oder sind aus Kompatibilitätsgründen zu sehr exotischen Clients vorhanden.

Soll Samba an spezielle Situationen angepaßt werden, ist es durch ein sehr flexibles Schema von Makroersetzungen möglich, die Konfigurationsdatei weitgehend dynamisch zu verändern. Damit sind erheblich mehr Konfigurationsmöglichkeiten gegeben als mit Windows. Als Beispiel sei genannt, daß man sehr einfach einen Sambaserver unter zwei verschiedenen Namen in der Netzwerkumgebung erscheinen lassen kann, und beide virtuelle Server unterschiedlich konfigurieren kann. Zu Testzwecken ist es sogar möglich, zwei unterschiedliche Versionen gleichzeitig auf einer Maschine laufen zu lassen.

Der Kostenaspekt ist hier bewußt nicht mit aufgeführt worden. Samba als freie Software[1] ist unter den Bedingungen der GNU General Public License für alle Zwecke kostenlos einsetzbar. Damit entstehen beim Einsatz von Samba keinerlei Lizenzkosten. Samba ist jedoch nicht kostenlos. Es müssen Administratoren eingewiesen werden, wenn Support benötigt wird, kann dieser viel Geld kosten.

Das Hauptaugenmerk sollte hier auf dem Aspekt liegen, daß Samba häufig einfach die technisch beste Lösung ist. Ein Kunde stand beispielsweise vor der Aufgabe, einige bestehende, kleinere NT-Server durch eine größere Lösung zu ersetzen. Durch einen einzigen großen NT-Server wären die bestehenden Server sehr wohl zu ersetzen gewesen. Das Problem bestand nun darin, daß in vielen Dokumenten die vorhandenen Servernamen über Objektreferenzen auf vollständige UNC-Pfadnamen hart kodiert waren. Damit mußten die vorhandenen Servernamen definitiv erhalten bleiben, um nicht jedes Dokument anfassen zu müssen. Ein einziger Server unter NT kam also nicht in Frage, unter Samba jedoch sehr wohl. Samba erlaubt die Einrichtung virtueller Server unter verschiedenen Namen auf einer einzigen Maschine. Mehr dazu ab Seite 26.

2 Eine einfache Konfiguration

Für den Anfang soll hier eine einfache Konfiguration beschrieben werden, mit der ein Samba-Server im Netz erscheint und einige, wenige Dienste anbietet. Diese einfache Konfiguration soll als Startpunkt das Experimentieren in den weiteren Kapiteln erleichtern. Die einzelnen Parameter werden hier kurz erklärt, in weiteren Kapiteln gibt es ausführlichere Erklärungen.

[1] Samba wird hier bewußt als *freie* Software im Sinne des GNU-Projektes verstanden. Samba ist dadurch natürlich auch Open Source Software

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Samba wird mit der Datei `smb.conf` konfiguriert. Je nach Unix oder Linux-Distribution kann diese Datei an unterschiedlichen Orten zu finden sein: `/etc/smb.conf`, `/etc/samba/smb.conf` oder auch `/usr/local/samba/lib/smb.conf`, wenn Samba selbst kompiliert wurde. Wurde die Datei `smb.conf` wie beschrieben angelegt, müssen zwei Dämonen gestartet werden: Der `nmbd` und der `smbd`. An dieser Stelle unterscheiden sich die Unix- und Linuxversionen erheblich, so daß keine allgemeinen Hinweise gegeben werden können. Verschiedene Möglichkeiten sind:

```
/etc/init.d/smb start
/sbin/init.d/smb start
/usr/local/samba/sbin/nmbd -D; /usr/local/samba/sbin/smbd -D
rcsmb start
```

Die `smb.conf` für eine einfache Konfiguration könnte so aussehen:

```
[global]
   workgroup = samba
   netbios name = sambaserver
   interfaces = eth0

   encrypt passwords = yes

[homes]
   valid users = %S
   writeable = yes
   browseable = no

[cdrom]
   path = /cdrom

[public]
   path = /pub
   writeable = yes
```

Wenn man mit dieser Einstellung Zugriff auf den Server ermöglichen möchte, muß man für jeden Benutzer einen Eintrag in der Datei `smbpasswd` machen, da verschlüsselte Paßwörter (`encrypt passwords = yes`) eingesetzt werden. Dies geschieht beispielsweise für den Benutzer **linux** über:

```
delphin:~ # smbpasswd -a linux
New SMB password:
Retype new SMB password:
Added user linux.
delphin:~ #
```

Die einzelnen Zeilen haben folgende Wirkung:

`[global]` leitet globale Servereinstellungen ein. Alle anderen Abschnitte beschreiben Freigaben.

`workgoup = samba` legt die Arbeitsgruppe fest, in der der Server auftauchen soll.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

`netbios name = sambaserver` gibt dem Server einen Namen, unter dem er im Netz erscheint.

`interfaces = eth0` beschreibt das Netzwerkinterface, auf dem Samba Dienste anbieten soll. Selbst wenn der Rechner nur ein einziges Netzwerkinterface hat, sollte dieser Parameter angegeben werden. Die vorhandenen Interfaces bekommt man bei den meisten Unixsystemen über den Befehl `netstat -ian` heraus.

`encrypt passwords = yes` verlangt vom Client, daß keine Klartextpaßwörter übertragen werden. Mit modernen Clients gibt es Probleme, wenn man diese Option nicht aktiviert. Außerdem möchte man aus Sicherheitsgründen seine Paßwörter nicht allen mitteilen.

`[homes]` leitet die Freigabe der Heimatverzeichnisse sämtlicher Benutzer ein. Jeder Benutzer bekommt eine eigene Freigabe unter seinem eigenen Namen und hat damit einen eigenen Bereich, auf dem er schreiben kann.

`valid users = %S` beschränkt den Zugriff auf den Benutzer, der sich verbinden möchte.

`writeable = yes` vergibt Schreibrecht auf die Freigabe. Standardmäßig wird nur Lesezugriff vergeben.

`browseable = no` versteckt die Freigabe `[homes]` in der Netzwerkumgebung. Der Client zeigt sie nicht mehr als `[homes]` an, sondern nur noch unter dem Benutzernamen.

`[cdrom]` leitet eine weitere Freigabe ein.

`path = /cdrom` gibt den genannten Pfad frei. Dieser muß selbstverständlich im Dateisystem existieren.

`[public]` macht noch eine Freigabe im Netz. Die Parameter sollten jetzt selbsterklärend sein.

Mit dieser minimalen `smb.conf` sollte es auf jeden Fall möglich sein, auf den Rechner zuzugreifen. Wenn man Probleme mit der Konfiguration weiterer Dienste bekommt, sollte man von einer möglichst einfachen Konfiguration ausgehen und dann Schritt für Schritt weitere Parameter hinzunehmen.

3 NetBIOS

Sobald Windowsrechner Dateisysteme austauschen, sich gegenseitig in der Netzwerkumgebung sehen oder Drucker freigeben, funktioniert die Kommunikation über NetBIOS[2]. Was ist NetBIOS? Je nachdem, wen man fragt, bekommt man unterschiedliche Antworten. Fragt man IBM, ist NetBIOS ein Protokoll, viele andere bezeichnen NetBIOS als reine Softwareschnittstelle zur Kommunikation von Rechnern. Mit dieser Schnittstelle werden Programmen unterschiedliche Dienste zur Kommunikation zur Verfügung gestellt. NetBIOS wurde entworfen, um in kleinen, lokalen Netzen Kommunikation zu ermöglichen. Beim Entwurf von NetBIOS wurde zunächst darauf geachtet, die Dinge sehr einfach zu halten, um sie in kleinen lokalen Netzen anwendbar zu machen. Auf Skalierbarkeit und die Andwendung in Weitverkehrsnetzen wurde beim Design nicht geachtet.

[2] Dies ist in reinen Windows 2000 Umgebungen nicht mehr richtig. Microsoft hat bei Windows 2000 die NetBIOS-Ebene abgeschafft, Windows 2000 kommuniziert direkt über TCP. Aus Kompatibilitätsgründen kann Windows 2000 jedoch noch über NetBIOS kommunizieren.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

3.1 NetBIOS-Dienste

Die Kommunikation mit NetBIOS wurde in drei Teilbereiche aufgeteilt, den Namens-, den Datagramm- und den Sitzungsdienst.

Namensdienst: Im Rahmen des Namensdienstes sind die Rechner in der Lage, sich gegenseitig im Netz zu identifizieren. Es sei an dieser Stelle betont, daß der NetBIOS-Namensdienst nichts mit der Anzeige in der Netzwerkumgebung zu tun hat. Der Computersuchdienst, der für die Netzwerkumgebung zuständig ist, hängt jedoch sehr stark von einem korrekt funktionierenden Namensdienst ab.

Datagrammdienst: Betrachtet man die Rechnerkommunikation auf dem Netz, so sieht man, daß die versendeten Daten in einzelne Pakete aufgeteilt werden. Diese einzelnen Pakete werden dann vom Netz nach bestem Bemühen an einen Zielrechner ausgeliefert. Geht ein Paket verloren, kann man nichts machen, man bekommt unter Umständen nicht einmal eine Benachrichtigung darüber, daß etwas nicht stimmt. Aufeinanderfolgende Pakete können außerdem in vertauschter Reihenfolge beim Empfänger ankommen. Es kann sogar sein, daß Pakete auf dem Weg dupliziert werden, also mehrfach ankommen.

Ein solches Netzwerk ist folglich zunächst einmal unzuverlässig. Diese Unzuverlässigkeit des Netzes wird durch den Datagrammdienst an die Benutzerprogramme weitergegeben. Das heißt, wenn ein Programm den Datagrammdienst nutzt, kann es nicht sicher sein, daß die Datenübertragung gewährleistet ist. Das Programm muß selbst dafür sorgen, daß mit Paketverlust vernünftig umgegangen wird.

Der Datagrammdienst hat jedoch nicht nur Nachteile. Zwei Vorteile sind der geringe Aufwand, mit dem Pakete verschickt werden können, und die Möglichkeit, ein Datagramm an mehrere Rechner gleichzeitig zu verschicken. Die Applikation muß selbst entscheiden, wie sie mit der Unzuverlässigkeit des Dienstes klarkommt.

Sitzungsdienst: Die Unzuverlässigkeit des Netzes ist für bestimmte Applikationen wie Dateitransfer oder Terminalanwendungen nicht akzeptabel. Wenn man eine Datei überträgt, möchte man sicher sein, daß die Datei komplett und korrekt übertragen wurde. Für diese höheren Anforderungen wurde der Sitzungsdienst entworfen. Zwei Rechner vereinbaren eine NetBIOS-Sitzung. Die Daten, die über diese Verbindung übertragen werden, kommen auf jeden Fall an, und zwar in der richtigen Reihenfolge. Können Daten einmal nicht übertragen werden, so erhält die versendende Applikation eine Fehlermeldung. Die Applikation kann nun versuchen, die abgebrochene Sitzung neu aufzubauen. Dieser Zuverlässigkeit steht ein erhöhter Aufwand beim Sitzungsauf- und -abbau gegenüber.

3.2 NetBIOS-Namensdienst

Zwei Rechner, die kommunizieren wollen, müssen sich zunächst gegenseitig identifizieren. NetBIOS sieht hierfür bis zu 16 Zeichen lange Namen vor. Jede Applikation kann für sich beliebig viele Namen reservieren und unter einem dieser Namen Verbindungen aufbauen und Daten austauschen. Diese Reservierung von Namen gilt sowohl für Server, die vom Netz aus erreichbar sein müssen, als auch für Clients, die Server im Netz erreichen wollen, da Server wissen müssen, wohin die Antworten gehen müssen.

Wollen zwei Anwendungen per NetBIOS miteinander kommunizieren, muß zunächst der Server seine Bereitschaft kundtun, Verbindungen entgegenzunehmen. Dazu meldet er sich im Netz mit seinem

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Namen an. Diese Anmeldung geschieht per Broadcast, so daß alle im Netz mithören können. Jeder Rechner ist frei, beliebige Namen im Netz für sich zu beanspruchen, sofern diese noch nicht belegt sind[3].

Eine Reservierung geschieht, indem ein Rechner per Broadcast ankündigt, daß er unter einem bestimmten Namen erreichbar ist. Dann wartet er auf Protest. Beklagt sich niemand, schickt er einen zweiten Reservierungsversuch und wartet wieder. Nach dem dritten Reservierungsversuch ist der Rechner ausreichend sicher, daß kein anderer den Namen bereits für sich eingenommen hat, und sieht ihn als für sich reserviert an.

Wenn nun ein Client mit einem Server reden möchte, dann muß er sich wie der Server einen eindeutigen Namen ausdenken und im Netz reservieren. Das Verfahren dazu ist identisch. Zusätzlich muß der Client jedoch die MAC-Adresse des Servers herausbekommen. Die Mechanismen, wie dies geschieht, hängen davon ab, wie NetBIOS implementiert ist.

3.3 NetBIOS-Implementationen

NetBIOS kann mit unterschiedlichen Protokollen implementiert werden. NetBEUI, IPX und TCP/IP sind drei heute verwendete Protokolle, wobei für Neuinstallation TCP/IP das bevorzugte Protokoll sein sollte. Der Ablauf der Namensauflösung soll an einem Beispiel verdeutlicht werden.

Auf einem Client soll eine Verbindung zu dem Server `samba` aufgebaut werden. Direkt erreicht man dies, indem man in der Taskleiste Start → Ausführen → `\\samba` eingibt. Im folgenden werden die unterschiedlichen Verfahren betrachtet, mit denen ein Rechner die MAC-Adresse des Servers herausbekommt.

NetBEUI: „Samba“ ⇒ MAC-Adresse

Bei diesem Protokoll findet der Client den Server ausschließlich über Broadcasts. Er verschickt per Broadcast eine Anfrage, wer sich für den gesuchten Namen verantwortlich fühlt. Der Rechner, der diesen Namen tatsächlich als Server reserviert hat, wird aufgrund dieser Anfrage seine eigene MAC-Adresse aus dem ROM seiner Netzwerkkarte auslesen und entsprechend antworten. Daraufhin kann der Client dann die Verbindung aufbauen. Über NetBEUI können also nur Rechner miteinander reden, die in der gleichen Broadcastdomäne liegen. Sobald Router zum Einsatz kommen sollen, kann reines NetBEUI nicht mehr verwendet werden, da dann der Server, der kontaktiert werden soll, von der Namensanfrage nichts mehr mitbekommt, also auch nicht antworten kann.

IPX „Samba“ ⇒ IPX-Knotenadresse ⇒ MAC-Adresse

Bei IPX liegt zwischen Servernamen und der MAC-Adresse die IPX-Knotenadresse. Diese enthält eine 4 Byte lange Netzwerknummer und die 6 Byte lange MAC-Adresse des Rechners. Die Knotenadresse wird anhand des NetBIOS-Namens wie bei NetBEUI per Broadcast im lokalen Netz gesucht. Damit wären Rechner hinter Routern nicht erreichbar, da die Namensanfrage nicht zu ihnen durchdringt. IPX-Router erkennen jedoch diese Namensanfragen und leiten sie per Broadcast in sämtliche angeschlossenen Netze weiter, so daß die Anfrage jedes Teilnetz erreicht.

Jede Anfrage löst einen Broadcast in jedem angeschlossenen Subnetz aus. Einige IPX-Router speichern eine Namenstabelle und können so viele Anfragen selbst beantworten, so daß die Broadcastlast reduziert wird.

[3] Mit dieser Freiheit ergeben sich viele Möglicheiten, von einem beliebigen Rechner aus ein Windows-Netz bis zur Unbenutzbarkeit zu stören. Man muß nur den Namen der Domäne mit dem Namenstyp `1d` zum richtigen Zeitpunkt reservieren und reserviert halten. Dann bootet der PDC nicht mehr sauber.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

TCP/IP „Samba“ ⇒ IP-Adresse ⇒ MAC-Adresse

Bei TCP/IP muß der Client die IP-Adresse des Servers herausfinden. Dies geschieht wie bei den anderen Protokollen per Broadcast im lokalen Netz. IP-Router können nicht angewiesen werden, die Anfragen per Broadcast in alle angeschlossenen Netze weiterzuleiten. Aus diesem Grund gibt es hier andere Mechanismen, die im folgenden beschrieben werden.

Nachdem die IP-Adresse herausgefunden wurde, kommen die bekannten Mechanismen von IP zum Tragen. Befindet sich der Rechner im eigenen Subnetz, wird direkt eine ARP-Anfrage nach der MAC-Adresse ausgelöst. Andernfalls wird der entsprechende Router anhand der Routingtabelle herausgefunden und dann dessen MAC-Adresse per ARP festgestellt.

Wenn NetBIOS über TCP/IP verwendet wird, kommen folgende Protokolle und Ports zum Einsatz:

Dienst	Protokoll	Port	Samba-Prozeß
Namensdienst	UDP	137	`nmbd`
Datagrammdienst	UDP	138	`nmbd`
Sitzungsdienst	TCP	139	`smbd`

Die Protokolle ordnen sich folgendermaßen ein:

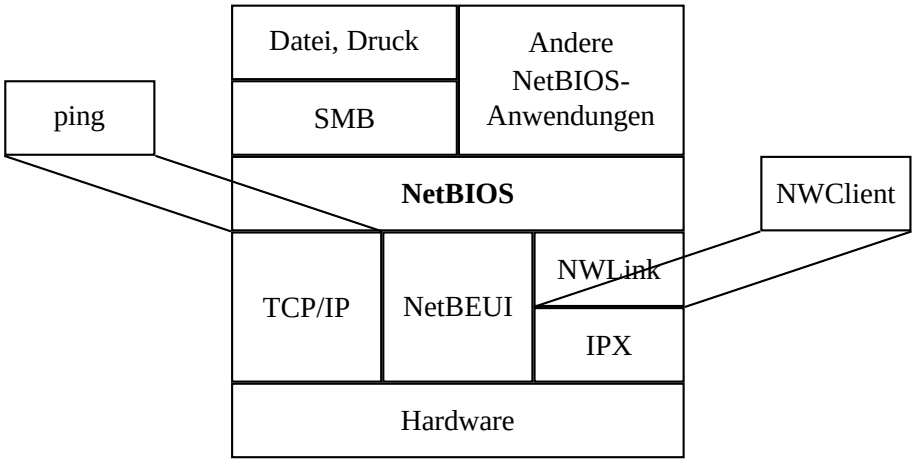


Abbildung 1: Protokollstapel

In dieser Grafik steht das Programm `ping` für beliebige Programme, die direkt auf TCP/IP aufsetzen. Dies gilt beispielsweise für alle WWW-Browser und für die Programme `telnet` und `ftp`.

Man kann festhalten, daß NetBEUI hier das einzige Protokoll ist, das nicht über Routergrenzen hinweg verwendbar ist. Sowohl IPX als auch IP sind für den Einsatz in Weitverkehrsnetzen entworfen worden und können folglich mit Routern umgehen.

Samba ist ausschließlich in der Lage, NetBIOS über TCP/IP zu benutzen. Daher werden die anderen Protokolle ab hier ignoriert. Für ein gut funktionierendes Netzwerk ist es jedoch sehr wichtig, daß auf den Clients nur die Protokolle installiert sind, die *wirklich* benötigt werden. Ist beispielsweise noch NetBEUI zusätzlich zu TCP/IP installiert, ist nicht klar, ob die Netzwerkumgebung in der NetBEUI- oder die in der TCP/IP-Welt gelten soll. Normalerweise ist heute ausschließlich noch TCP/IP notwendig. IPX kann dann noch benötigt werden, wenn es Novellsysteme im Netz gibt.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

4 Bestandteile von Samba

Das Programmpaket Samba besteht aus mehreren Programmen, von denen einige der Serverseite und andere der Clientseite zugeordnet werden können.

4.1 Die Servertools

smbd ist der zentrale Serverprozeß, der für die eigentlichen Datei- und Druckdienste zuständig ist. Sie werden mehrere `smbd`s im System finden. Einer dieser Prozesse hört auf dem TCP-Port 139, und nimmt neue Verbindungen entgegen. Jede neue Verbindung stößt einen neuen `smbd` Prozeß an. Wenn Sie einen Client vom Sambaserver trennen wollen, müssen Sie nur mit `smbstatus` die Prozeßnummer des zuständigen `smbd` erfragen, und diesen einen Prozeß töten.

Jeder *aktive* Client benötigt etwa 1-2 MB Hauptspeicher auf dem Server. Clients, die gerade nicht aktiv Dateien mit dem Sambaserver austauschen, benötigen praktisch überhaupt keine Resourcen. Viel Hauptspeicher kann von Samba selbstverständlich gut als Cache genutzt werden.

nmbd ist für die NetBIOS Namens- und Datagrammdienste zuständig. Dieser Prozeß reserviert beim Start von Samba die entsprechenden NetBIOS-Namen, er kann WINS-Server sein und ist für den Computersuchdienst zuständig.

testparm Mit diesem Programm kann man die `smb.conf` auf syntaktische Korrektheit prüfen. Das Programm liest die Konfigurationsdatei ein und gibt Fehlermeldungen aus, sofern es unbekannte Parameter findet.

smbpasswd wird zur Pflege der verschlüsselten Paßwörter auf Serverseite verwendet. Wie dies funktioniert, wird im Kapitel 20 erklärt.

smbcontrol Mit diesem Programm lassen sich die Dämonen von Samba kontrollieren. Beispielsweise kann man für einzelne Dämonen den Debuglevel gezielt auf einen gewünschten Wert setzen.

4.2 Die Clients

smbclient Mit dem Programm `smbclient` kann man auf Freigaben von NT-Rechnern zugreifen. Man kann auf von NT zur Verfügung gestellten Druckern drucken und man kann NT-Freigaben in tar-Dateien sichern. Weiterhin kann mit `smbclient` die Liste der Server im Netz erfragt werden, analog zu der Netzwerkumgebung unter Windows.

nmblookup ist ein Diagnosewerkzeug für die NetBIOS-Namensauflösung. Wenn zwei Computer mit Windows sich nicht finden können, kann man mit `nmblookup` deren Versuche, sich gegenseitig zu finden, genau nachstellen. Ebenso können WINS-Server befragt werden und ein NetBIOS Node Status abgefragt werden. Das entsprechende Programm auf unter Windows ist das Kommandozeilenprogramm `nbtstat`.

smbcacls: Mit diesem Programm lassen sich von Unix aus Access Control Lists auf Windows-Dateien auslesen und setzen. Ist Samba mit ACL-Support kompiliert, geht dies selbstverständlich auch für die auf Unix abgelegten Dateien.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

4.3 Weitere Serverkomponenten

smb.conf: Die zentrale Konfigurationsdatei von Samba. Ist Samba als fester Systembestandteil installiert, findet sie sich in der Regel unter `/etc/smb.conf`. Wurde Samba selbst kompiliert, so liegt sie häufig unter `/usr/local/samba/lib/smb.conf`.

/var/lock/samba: Samba benötigt ein Verzeichnis, in dem es temporäre Lockdateien und Datenbanken ablegen kann. Wird Samba ohne besondere Optionen selbst kompiliert, liegt dieses Verzeichnis unter `/usr/local/samba/var`.

/etc/smbpasswd ist die Paßwortdatenbank von Samba, sofern mit verschlüsselten Paßwörtern gearbeitet wird. Bei selbst kompilierten Sambaversionen liegt diese Datei häufig im Verzeichnis `/usr/local/samba/private/`.

5 NetBIOS-Konfiguration mit Samba

Als erstes soll eine minimale Konfiguration von Samba erreicht werden, mit der jeder Rechner in der Netzwerkumgebung zu sehen ist. Dazu sollte die Datei `smb.conf` folgendermaßen aussehen:

```
[global]
workgroup = arbeitsgruppe
interfaces = <IP-Adresse>/<Netzmaske>
```

Der grundsätzliche Aufbau der `smb.conf` gleicht dem Aufbau der .INI-Dateien von Windows 3. Die Datei ist in mehrere Abschnitte unterteilt, die jeweils durch einen Abschnittsnamen eingeleitet werden. Dieser Abschnittsname selbst wird in eckige Klammern gesetzt. Der Inhalt jedes Abschnitts besteht nun aus Parameterzuweisungen. Im Beispiel gibt es nur den Abschnitt `global`. In diesem werden Festlegungen getroffen, die den Server als ganzes betreffen. Wenn später Freigaben erstellt werden, geschieht dies durch Anlegen von weiteren Abschnitten.

Mit dem Parameter `workgroup =` wird die Arbeitsgruppe festgelegt, in der sich der Server befinden soll.

Der Parameter `interfaces =` ist einer der wichtigsten Parameter der Sambakonfiguration. Er ist deshalb so wichtig, weil damit das Funktionieren des NetBIOS-Systems innerhalb von Samba garantiert wird. Später wird deutlich werden, daß große Teile der Kommunikation auf Broadcasts basieren. Mit `netstat -ia` bekommt man auf den meisten Unix-Systemen Informationen über die vorhandenen Netzwerkinterfaces. Mit `ifconfig <interface>` kann man sich dann nähere Informationen anzeigen lassen.

Der Parameter `interfaces =` weist Samba an, diese und keine andere Schnittstelle zu nutzen. Darüberhinaus ist Samba nun in der Lage, die Broadcastadresse, die auf dieser Schnittstelle gültig ist, zu bestimmen. Theoretisch könnte Samba die Broadcastadresse selbständig herausfinden, aber es gibt keinen portablen Weg, dies über Systemgrenzen hinweg zu tun. Das sicherste ist, Samba direkt über die Broadcastadresse zu informieren.

Meistens funktioniert zusätzlich zur Notation

```
interfaces = <IP-Adresse>/<Netzmaske>
```

auch `interfaces = <Interface-Name>`.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Mit diesen beiden Einstellungen wird man direkt den Sambarechner in der Netzwerkumgebung sehen. Will man Tests auf NetBIOS-Ebene durchführen, sollte man zur Vereinfachung zwei weitere Parameter setzen. Mit diesen drei Parametern bekommt man einen *komplett offenen* Server. Die Parameter werden in späteren Abschnitten genauer erklärt. Die vollständige `smb.conf` sieht folgendermaßen aus:

```
[global]
workgroup = arbeitsgruppe
interfaces = <IP-Adresse>/<Netzmaske>
security = share
encrypt passwords = yes
```

Mit dieser Konfiguration kann Samba gestartet werden. Es werden die beiden Dämonen `nmbd` und `smbd` benötigt. Diese kann man direkt von der Kommandozeile starten.

Setzt man SuSE Linux ein, so kann man Samba mit dem Aufruf

```
rcsmb start
```

Damit Samba beim nächsten Hochfahren automatisch gestartet wird, sollte die Variable `START_SMB` in der Datei `/etc/rc.config` auf `yes` gesetzt werden.

Es ist denkbar, den Aufruf beider Programme durch den `inetd` durchführen zu lassen. Bei Samba ist dies jedoch nicht sinnvoll. Insbesondere der `nmbd` muß auf jeden Fall beim Start des Systems hochfahren, da dieser im NetBIOS-System Namen für sich reservieren muß. Würde er erst bei der ersten Anfrage gestartet, hätten Windowsrechner keine Möglichkeit, den Sambarechner zu finden. Außerdem wird sich der `nmbd` nicht wieder beenden, sobald er einmal gestartet wurde. Der `smbd` könnte durch den `inetd` gestartet werden. Jedoch ist der Resourcenbedarf nicht so hoch, daß die erhöhte Startzeit damit gerechtfertigt werden könnte.

Nachdem alle Sambaserver gestartet wurden, sollten diese in der Netzwerkumgebung der beteiligten Windowsrechner erscheinen.

6 Namensauflösung per Broadcast

Mit `nmblookup` kann man direkt eine NetBIOS-Namensanfrage auslösen.

```
vlendec@server:/home/vlendec> nmblookup server
querying server on 192.168.1.255
192.168.1.3 server<00>
vlendec@linux:/home/vlendec>
```

An diesem Beispiel wird deutlich, wie die NetBIOS-Namensauflösung normalerweise arbeitet. Es wird ein Paket an der Adresse 192.168.1.255 versendet, das heißt an die Broadcastadresse im lokalen Subnetz. Um NetBIOS-Namensanfragen zu ermöglichen, muß Samba in der Lage sein, die Broadcastadresse herauszufinden, an die das Paket geschickt werden soll. `nmblookup` entnimmt diese Adresse der Zeile `interfaces =` der `smb.conf`. Für Tests kann man `nmblookup` mit dem Parameter -B anweisen, die Anfragen an eine andere Broadcastadresse zu versenden.

```
vlendec@delphin:~ > nmblookup -B 192.168.234.31 server
querying server on 192.168.234.31
name_query failed to find name server
vlendec@delphin:~ >
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

In diesem Beispiel wurde die Broadcastadresse auf 192.168.1.31 gesetzt. Diese Broadcastadresse gilt in Subnetz 192.168.1.0/27. Jedoch fühlte sich der `nmbd`, der für diesen Namen verantwortlich ist, nicht angesprochen. Folglich hat er nicht auf diese Namensanfrage geantwortet.

Unter Windows kann man die Namensanfrage so isoliert nicht auslösen, man muß eine Verbindung aufbauen. Windows unterhält einen Cache, in dem erfolgreiche Anfragen zwischengespeichert werden. Diesen kann man sich mit `nbtstat -c` anzeigen und mit `nbtstat -R` löschen. Man kann eine Anfrage erzwingen, indem man mit leerem Namenscache eine Verbindung aufbaut, beispielsweise durch ein `net view \\samba`.

Die Fehlermeldung, wenn eine NetBIOS-Namensanfrage fehlschlägt, lautet im GUI: „Der Netzwerkpfad wurde nicht gefunden“. Auf der Kommandozeile kommt noch die Fehlermeldung 53 dazu.

Mit `nmblookup` und `nbtstat` kann man sich zusätzlich die von einem Rechner reservierten Namen ausgeben lassen. Die entsprechende Operation nennt sich *Node Status Request* und wird durch den Parameter `nmblookup -A <IP-Adresse>` ausgelöst. Die Ausgabe eines solchen Node Status Request zeigt, daß ein Rechner für sich nicht nur einen einzigen Namen reserviert, sondern gleich mehrere.

```
vlendec@delphin:~ > nmblookup -A 192.168.234.5
Looking up status of 192.168.234.5
received 6 names
        NT4WKS          <00> -          B <ACTIVE>
        SAMBA           <00> - <GROUP> B <ACTIVE>
        NT4WKS          <03> -          B <ACTIVE>
        ADMINISTRATOR   <03> -          B <ACTIVE>
        NT4WKS          <20> -          B <ACTIVE>
        SAMBA           <1e> - <GROUP> B <ACTIVE>
num_good_sends=0 num_good_receives=0

vlendec@delphin:~ >
```

Zunächst gibt es die Einzelnamen, zum Beispiel den Computernamen selbst. Für diese gilt die Regel, daß sie nur ein einziges Mal im gesamten Netz auftauchen dürfen. Sie werden reserviert und stehen dem entsprechenden Rechner dann exklusiv zur Verfügung. Daneben gibt es die Gruppennamen, die im Node Status Request durch `<GROUP>` markiert sind. Diese kann es mehrfach im Netz geben. Die Gruppennamen sind insbesondere als Ziele für NetBIOS-Datagramme interessant. Beispielsweise reserviert jeder Teilnehmer an einer Arbeitsgruppe den NetBIOS-Gruppennamen `arbeitsgruppe<00>`. Damit kann ein Rechner mit einem einzigen verschickten Datagramm an diesen Namen sämtliche Rechner in dieser Arbeitsgruppe erreichen.

Zusätzlich fällt auf, daß beispielsweise der Computername selbst als Einzelname mehrfach reserviert ist. Hier kommen die unterschiedlichen Namenstypen ins Spiel. Das 16. Byte eines NetBIOS-Namens ist für ein Typfeld reserviert. So sind unterschiedliche Anwendungen auf einem Rechner in der Lage, sich Namen zu reservieren, ohne sich gegenseitig zu stören. Der Wert des Typfeldes wird hexadezimal in spitzen Klammern angegeben.

Zunächst die Einzelnamen, die häufig auftauchen:

computername<00> Hiermit tut der Rechner einfach seine Existenz kund. Wenn ein Rechner auf Resourcen anderer Rechner zugreift, wird als Clientname dieser Name benutzt.

computername<20> Dieser Name wird für den Serverdienst reserviert. Wenn ein Rechner als Datei- oder Druckserver angesprochen werden soll, dann wird eine Verbindung zu diesem NetBIOS-Namen aufgebaut.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

computername<03> Unter diesem Namen meldet sich der Nachrichtendienst des Rechners an. Kurze Meldungen, die unter Windows NT mit dem Kommando `net send` abgesetzt werden, und unter Windows 95 mit dem Programm Winpopup verschickt werden, kann der Rechner damit empfangen und am Bildschirm anzeigen.

arbeitsgruppe<1d> Dieser Rechner ist der so genannte *Locale Master Browser*, der die Liste sämtlicher Rechner in der Netzwerkumgebung pflegt.

arbeitsgruppe<1b> Dieser Rechner ist der *Domain Master Browser*, der über Subnetzgrenzen hinweg für die Netzwerkumgebung zuständig ist.

Einige Gruppennamen werden ebenfalls reserviert:

arbeitsgruppe<00> Ein Rechner verkündet hiermit seine Zugehörigkeit zu einer Arbeitsgruppe. Beispielsweise können Winpopup-Meldungen an eine ganze Arbeitsgruppe versendet werden. Dies geschieht per Datagramm an diesen Namen.

arbeitsgruppe<1c> Jeder Domain Logon Server reserviert diesen Namen für sich. Clients finden ihre Domain Controller über diesen NetBIOS-Namen.

arbeitsgruppe<1e> Wahlen zum Local Master Browser werden über diesen Namen abgewickelt. Siehe hierzu Kapitel 7.

Damit sind die für Datei- und Druckerdienste wichtigsten Namenstypen beschrieben. Sobald unter NT andere Dienste installiert werden, kommen andere Namenstypen hinzu. Exchange zum Beispiel nutzt die Namenstypen 22, 23 und 24. Mehr Namenstypen findet man in der Microsoft Knowlegde Base unter Artikel Nummer Q163409.

7 Netzwerkumgebung

Die Netzwerkumgebung ist eine Anzeige, in der sämtliche Server im Netz aufgeführt sind. Alle Rechner, die Datei- oder Druckfreigaben zur Verfügung stellen, erscheinen dort oder sollten es zumindest, wenn alles reibungslos funktioniert. Jeder Client, der auf eine solche Resource zugreifen möchte, kann sich die Liste der Server im Netz geben lassen. Damit diese Anzeige nicht zu unübersichtlich gerät, werden die Rechner in so genannte Arbeitsgruppen aufgeteilt. Jeder Rechner wird einer Arbeitsgruppe zugeordnet, in erscheint und die er als erstes beim Doppelklick auf das Symbol „Netzwerkumgebung“ angezeigt. Die anderen Arbeitsgruppen sind ebenfalls über den Unterzweig „Gesamtes Netzwerk“ sichtbar.

Bezüglich des Zugangs zu Arbeitsgruppen findet keinerlei Authentifizierung statt. Jeder Rechner kann frei für sich entscheiden, in welcher Arbeitsgruppe er erscheinen möchte, jeder im Netz kann sich beliebige Arbeitsgruppen anzeigen. Dies gilt ebenfalls, wenn im Netz mit NT-Domänen gearbeitet wird. NT-Domänen haben nur eher zufällig im Netz ein der Arbeitsgruppe ähnliches Erscheinungsbild.

Das klingt verwirrend, ist es vermutlich beim ersten Lesen auch. Zum Verständnis des Windows-Networking muß man drei Begriffe ganz klar von einander trennen:

NetBIOS Unter jeglicher Kommunikation von Windowsrechnern liegt das API NetBIOS. Mit Hilfe des NetBIOS sind Rechner im Netz ansprechbar, sie können verschiedenste Dienste anbieten. Einer dieser Dienste ist die Netzwerkumgebung.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Arbeitsgruppe Eine Arbeitsgruppe ist eine reine Liste von Rechnern. Sie hat mit NetBIOS *ausschließlich* als Transportmedium zu tun. Der Dienst, der die Netzwerkumgebung bereit stellt, könnte theoretisch vollständig unabhängig von NetBIOS implementiert werden, ist in der Praxis aber sehr von einem funktionierenden NetBIOS abhängig.

Domäne Eine Domäne bezeichnet etwas völlig anderes als eine Arbeitsgruppe.Eine Domäne ist eine gemeinsam genutzte Benutzerdatenbank. Microsoft hat in seiner Implementation einer Domäne die Einschränkung gemacht, daß alle Rechner einer Domäne in einer Arbeitsgruppe auftauchen müssen. Das heißt aber nicht, daß alle Rechner in der Arbeitsgruppe einer Domäne auch die gemeinsame Benutzerdatenbank nutzen müssen.

Das Auftauchen eines Rechners in der Netzwerkumgebung hat nichts mit seiner Erreichbarkeit zu tun, es ist höchstens ein vager Hinweis darauf, daß man es dort einmal versuchen kann. Ein Rechner erreichbar sein, aber nie auftauchen, oder er kann in der Netzwerkumgebung stehen, aber lange nicht mehr erreichbar sein.

Die *Netzwerkumgebung* ist einer der instabileren Aspekte von Windows. Hiermit kann man sich, sofern alles funktioniert, alle Rechner in einer Arbeitsgruppe anzeigen lassen. Dabei dauert es mitunter geraume Zeit, bis ein Rechner in einer Anzeige erscheint, und es dauert unter Umständen noch länger, bis er wieder verschwindet.

Eine naive Implementation könnte so aussehen: Jeder Rechner, der Serverdienste anbietet, teilt dies regelmäßig per Broadcast im Netz mit. Ein solches Vorgehen hat jedoch mehrere Nachteile. Erstens würde die Last im Netz mit jedem zusätzlichen Rechner stark ansteigen. Zweitens muß jeder Rechner, der die Netzwerkumgebung anzeigen will, relativ komplexe Software laufen lassen. Und drittens scheitert dieses Schema auf jeden Fall an Subnetzgrenzen, die für Broadcasts eine Grenze darstellen. Aus diesen Gründen ist man einen anderen Weg gegangen.

Der *Local Master Browser*[4] (im Folgenden auch LMB genannt) ist ein Rechner, der im Netz die Netzwerkumgebung pflegt. Dieser Rechner wird nirgendwo zentral bestimmt, sondern er wird gewählt. Diese Wahl findet immer dann statt, wenn einer der beteiligten Rechner feststellt, daß es im Moment keinen solchen Local Master Browser gibt. Beispielsweise kann der Explorer von Windows eine solche Wahl anstoßen. Wenn Windows 95 beim Öffnen der Netzwerkumgebung die geschwenkte Taschenlampe anzeigt, wird der LMB gesucht. Ist keiner vorhanden, wird eine Wahl angestoßen.

Die Wahl erfolgt mit Datagrammen an den Gruppennamen `arbeitsgruppe<1e>`. Ein Rechner verschickt ein Datagramm an diesen Namen. Jeder Rechner, der diesen Namen reserviert hat, hört dieses Datagramm und entscheidet, wie er selbst vorgehen soll. In dem Datagramm sind verschiedene Kriterien zur Wahl enthalten, beispielsweise das Betriebssystem des versendenden Rechners. Hieraus folgt, daß es in einem Subnetz für jede vorhandene Arbeitsgruppe einen LMB gibt.

Empfängt beispielsweise eine Windows NT Workstation ein Paket von einem Windows NT Server, so entscheidet sie, daß sie die Wahl verloren hat. Damit wird sie selbst nicht mehr aktiv. Kommt dieses Paket jedoch von einem Rechner mit Windows 95, so hält sie sich selbst für geeigneter, den Local Master Browser zu übernehmen. Dann wird sie selbst ein solches Wahlpaket mit ihren Parametern versenden. Der Windows 95 Rechner empfängt dies, und sieht, daß er verloren hat. Auf diese Weise schaukelt sich die Wahl hoch, bis der „beste“ Rechner die Wahl gewinnt.

Wenn es nun mehrere Windows NT Workstations im Netz gäbe, dann wäre die Wahl unentschieden. An dieser Stelle kommt die *Uptime* der Rechner ins Spiel. Der Rechner, der am längsten läuft,

[4] Der Local Master Browser wird in der deutschen Dokumentation von Windows *Computersuchdienst* genannt. Der Domain Master Browser ist der Domänenhauptsuchdienst. Local Master Browser fi nde ich sehr viel handlicher, und daher werde ich den englischen Begriff verwenden.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

gewinnt die Wahl. Nun kann es sein, daß nach einem Stromausfall zwei Rechner genau die gleiche Uptime haben. Dann kommt als letztes und eindeutiges Entscheidungskriterium der NetBIOS-Name des Rechners zum Zug. Der alphabetisch vorne stehende Rechner gewinnt. Mit diesen drei Kriterien ist eine eindeutige Wahl gesichert.

Samba ordnet sich in der Standardeinstellung zwischen Windows 95 und Windows NT ein, das heißt, gegen Windows 95 gewinnt Samba die Wahl, überläßt jedoch Windows NT Rechnern den Local Master Browser.

Drei Parameter in der `smb.conf` bestimmen das Verhalten von Samba in der Wahl zum Local Master Browser:

`os level` Damit wird die Einordnung von Samba in die unterschiedlichen Betriebssysteme geregelt. Diese haben für die Betriebssystemstufe folgende Werte:

Windows for Workgroups	0
Windows 95/98	1
Windows NT Workstation	16
Samba	20
Windows NT Server	32

Diese Werte sind nicht als fest anzusehen. Wenn ein neues Service Pack für ein Betriebssystem herausgegeben wird, ist es möglich, daß in der Software für den Local Master Browser Fehler bereinigt wurden. Dann ist es sinnvoll, daß diese neue Software die Rolle des LMB übernimmt.

Der Parameter `os level` kann Werte von 0 bis 255 annehmen. Setzt man ihn auf 255, wird nach einer erfolgreichen Wahl niemand mehr Local Master Browser werden können.

`local master` Möchte man auf keinen Fall den LMB auf einem Sambarechner haben, so setzt man den Parameter `local master = no`. Dann nimmt Samba an keiner Wahl teil.

`preferred master` Mit der Standardeinstellung `preferred master = no` sucht Samba beim Start nach einem LMB. Findet er einen, meldet er sich dort. Findet er keinen LMB, bleibt Samba passiv. Jemand anders muß eine Wahl anstoßen. Wenn dann eine Wahl stattfindet, nimmt Samba teil und ordnet sich anhand seines `os level` ein.

Es ist sehr sinnvoll, den Local Master Browser ständig auf einer festen Maschine laufen zu lassen. Häufige Wechsel des Local Master Browser lassen die Netzwerkumgebung aus zwei Gründen sehr instabil werden. Erstens müssen sich die Server im Netz häufig an neuen Local Master Browsern anmelden. Diese Anmeldung erfolgt per UDP und kann auch mal fehlschlagen. Zweitens kann es passieren, daß ein Client den Wechsel eines Local Master Browser nicht mitbekommt und Informationen von einem nicht mehr aktuellen Local Master Browser beziehen möchte. Ein Sambaserver ist typischerweise eine Maschine, die als Server durchläuft und auch deutlich stabiler als Windows-Clients ist. Mit den Einstellungen

```
[global]
os level = 100
preferred master = yes
```

kann man sicher sein, daß der Sambarechner immer den Local Master Browser innehat. `preferred master = yes` stellt sicher, daß beim Start von Samba eine Wahl angestoßen wird, und mit `os level = 100` gewinnt Samba diese Wahl gegen alle anderen Maschinen im Netz. Es sei denn, ein anderer Administrator von Samba kommt auf die Idee, einen noch höheren Wert für den `os level` zu benutzen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

8 NetBIOS über Subnetzgrenzen

Die wenigsten Firmen haben heutzutage nur ein einziges LAN. Entweder sind verschiedene Gebäude oder Standorte mit Routern verbunden, oder jemand wählt sich in das Firmennetz ein. In diesen Fällen funktioniert die Namensauflösung nicht mehr wie beschrieben. Wird die Namensreservierung und -auflösung ausschließlich per Broadcast durchgeführt, kann man Rechner, die hinter Routern liegen, nicht erreichen. Broadcasts verbleiben in den Subnetzen, in denen sie ausgesendet wurden.

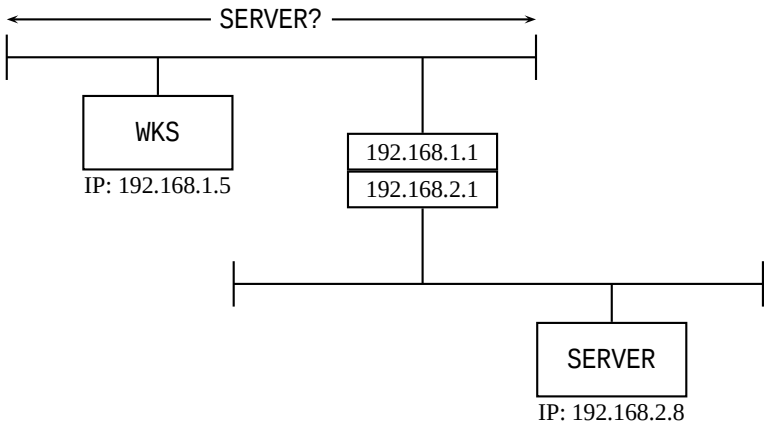


Abbildung 2: Namensanfrage per Broadcast

In der dargestellten Situation sind zwei Netze über einen Router verbunden. Jeder der beiden Rechner reserviert seinen Namen in dem ihm zugeordneten Subnetz. Die Workstation `WKS` schickt ihre Reservierungen per Broadcast an 192.168.1.255, und der Server `SERVER` wird seinen Namen auf 192.168.2.255 reservieren. Der Router zwischen beiden bekommt diese Reservierungen zwar mit, wird sie aber nicht in das jeweils andere Subnetz weiterleiten. Wenn nun `WKS` ihren Server `SERVER` sucht, geschieht dies ebenfalls per Broadcast an 192.168.1.255. Diese Anfrage bleibt wie dargestellt im oberen Subnetz und erreicht `SERVER` gar nicht, so daß dieser auch nicht antworten kann.

8.1 LMHOSTS

Der einfachste Weg, die Namensauflösung über Subnetzgrenzen hinweg zu realisieren, geht über eine statische Tabelle. Unter Windows liegt diese in der Datei `LMHOSTS`. Sie liegt abhängig von der Windowsversion in unterschiedlichen Verzeichnissen und läßt sich am einfachsten mit der Suchfunktion des Desktops finden. Diese Datei ist ähnlich aufgebaut wie die Datei `/etc/hosts` unter Unix. Ein Beispieleintrag ist der folgende:

```
192.168.1.5 samba
```

Die Einträge in der `LMHOSTS` können durch den Zusatz `#PRE` ergänzt werden. Dieser Zusatz legt fest, in welcher Reihenfolge die Namensauflösung vorgenommen wird. Ist kein `#PRE` vorhanden, so wird zunächst eine konventionelle Namensauflösung per Broadcast versucht. Erst, wenn diese fehlschlägt, wird in der `LMHOSTS` nachgeschaut. Ist der Zusatz vorhanden, so wird ohne Namensauflösung direkt der Wert in der `LMHOSTS` verwendet.

Die Namensauflösung über die Datei `LMHOSTS` hat wie die Datei `/etc/hosts` den entscheidenden

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Nachteil, daß sie auf jedem Rechner einzeln gepflegt werden muß. Das macht diese Art der Namenspflege sehr schnell unwartbar. Die Syntax der `LMHOSTS` läßt einen einfachen Trick zu, mit dem zentral eine `LMHOSTS`[5] vorgehalten werden kann, das Statement `#INCLUDE`. Man stellt an zentraler Stelle eine Freigabe zur Verfügung, in der die `LMHOSTS` steht, und fügt sie automatisch bei jedem booten in die Liste auf den Clients ein. Dazu muß einmalig auf den Clients die `LMHOSTS` folgendermaßen aufgesetzt werden:

```
192.168.1.1 samba #PRE
#INCLUDE \\samba\public\lmhosts
```

Die einzelnen Werte sind natürlich den Gegebenheiten vor Ort anzupassen. Es ist darauf zu achten, daß die Worte `#PRE` und `#INCLUDE` in Großbuchstaben geschrieben sind. Bei den Namen selbst die Großschreibung egal.

8.2 WINS

Die zweite Möglichkeit, das Problem zu lösen, ist ein zentraler Server, der die NetBIOS-Namen in einer Datenbank dynamisch pflegt. Dazu gibt es den WINS-Server. Ein solcher Server ist ein Rechner, bei dem sich jede NetBIOS-Applikation im Netz mit ihren Namen anmeldet. Die IP-Adresse dieses Servers muß jedem Rechner mitgeteilt werden. Bei Windows geschieht dies in den Eigenschaften des TCP/IP Protokolls im Reiter WINS-Adresse. Setzt man DHCP-Server ein, kann man ebenfalls den WINS-Server festlegen. Samba bekommt die Adresse mit dem Parameter `wins server = <ip-adresse>` im Abschnitt `[global]` der `smb.conf` mitgeteilt. Sobald ein Client die IP-Adresse des WINS-Servers kennt, ist es völlig gleichgültig, ob sich dieser im gleichen Subnetz befindet oder nicht.

Die Namensreservierung erfolgt nicht mehr per Broadcast, sondern mit einem gerichteten UDP-Paket an den WINS-Server. Gerichtete Pakete leitet der Router wie jedes andere Paket an den WINS-Server weiter. Dieser sieht in seiner Tabelle nach, ob der Name bereits reserviert ist. Ist das nicht der Fall, so wird er spontan eine Bestätigung der Reservierung zurückschicken. Diese Reservierung gilt nun für eine bestimmte Zeit und muß rechtzeitig erneuert werden.

Ist der Name bereits reserviert, wird der WINS-Server den bisherigen Besitzer befragen, ob er den Namen noch benötigt. Bekommt er keine Antwort, wird er dem neuen Besitzer ebenfalls eine Bestätigung schicken. Möchte der alte Besitzer den Namen noch verwenden, so wird der Anfragende eine Ablehnung der Reservierung erhalten. Diese Nachfrage ist notwendig, um einem abgestürzten Rechner das spontane Booten zu ermöglichen, da bei einem Absturz keine Freigabe der Namensreservierung erfolgen kann.

Die Namensanfrage, die in Abbildung 2 den Server nicht erreichte, weil der Router keine Broadcasts weitergibt, wird nun direkt an den WINS-Server gerichtet, der in seiner Tabelle nachsehen kann.

Samba kann als WINS-Server konfiguriert werden, indem der Parameter `wins support = yes` gesetzt wird. Ist dieser Parameter gesetzt, kann Samba nach einem Neustart bei allen Clients und allen sonstigen Servern als WINS-Server eingetragen werden. Werden diese dann neu gestartet, melden sie sich beim WINS-Server an.

Wenn nun ein Rechner mit Samba als WINS-Server konfiguriert ist, und sich die anderen Rechner dort anmelden, werden diese in der Datei `/var/lock/samba/wins.dat` abgelegt. Der `nmbd` pflegt diese Datei dynamisch, je nach Reservierungen und Abmeldungen. Die Datei `wins.dat` wird in regelmäßigen Abständen geschrieben. Wenn es notwendig sein sollte, den wirklich aktuellen Stand unabhängig von diesem Zeitintervall zu erhalten, so kann man dem `nmbd` das `HANGUP`-Signal durch den Befehl `killall -HUP nmbd` senden. Außerdem wird die `wins.dat` beim Beenden des `nmbd` geschrieben.

[5] Zentrale LMHOSTS

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

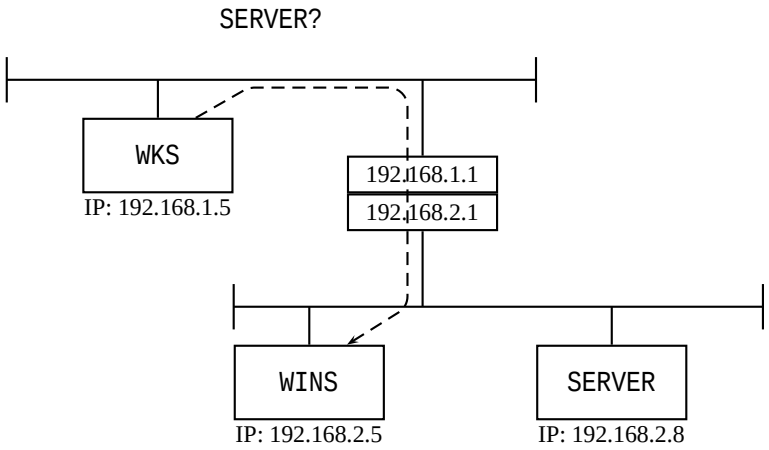


Abbildung 3: WINS-Anfrage

Diese Datenbank wird auf Festplatte gehalten, damit die Daten einen Neustart von Samba überleben. Jeder Rechner, der einen Namen für sich reserviert hat, hat diese Reservierung für einen bestimmten Zeitraum ausgesprochen. Wenn Samba jetzt neu gestartet werden sollte, und dadurch die Datenbank verloren ginge, wäre der gesamte NetBIOS-Namensraum nicht mehr verfügbar. Außerdem kann ein WINS Server die angeschlossenen Clients weder von sich aus finden, noch sie darum bitten, sich erneut zu registrieren. Daher ist die WINS Datenbank über Neustarts von Samba hinaus zu erhalten.

Die Anfrage, die die Workstation `WKS` absetzt, wird nun nicht mehr per Broadcast gestellt, sondern mit einem gerichteten Paket an den WINS-Server, bei dem sich alle Rechner angemeldet haben.

WINS hat gegenüber der broadcastbasierten Namensreservierung einige Vorteile. Namensreservierung per Broadcast benötigt Wartezeiten. Es wird die Reservierung angekündigt, es wird gewartet, die Reservierung wird erneut angekündigt, und es wird wieder gewartet. Dieses Spiel wiederholt sich mehrfach, bis der Rechner sicher sein kann, daß ein eventueller Vorbesitzer des Namens genug Zeit hatte, sich zu beklagen. Beim Einsatz von WINS entfallen diese Wartezeiten, da hier ein einziger Rechner sämtliche reservierte Namen registriert und in seiner Tabelle nachschauen kann. Daher ist die Reservierung per NetBIOS deutlich schneller, und auch weniger netzbelastend. Selbst wenn man also nur ein einziges Subnetz hat, sollte man zur Reduzierung der Netzlast den Einsatz eines WINS-Servers in Erwägung ziehen.

Zusätzlich sei hier angemerkt, daß es netzwerkweit nur einen einzigen WINS-Server geben darf. Selbst wenn es unterschiedliche Arbeitsgruppen oder Domänen gibt, darf es nicht mehr als einen WINS-Server geben. Setzt man mehrere WINS-Server ein, hat man getrennte Namensräume und handelt sich damit massive Probleme ein, da Windows Namen sowohl beim WINS als auch per Broadcast auflöst.

Rechner im einen Namensraum können mit Rechnern, die an einem anderen WINS-Server angeschlossen sind, nicht kommunizieren, da die Namen nicht aufgelöst werden können. Namen, die beim WINS nicht bekannt sind, werden von Windows zusätzlich lokal per Broadcast aufgelöst. Das heißt, man findet beim einige Rechner nur per WINS, andere auch lokal. Die Fehlerdiagnose wird dadurch stark erschwert.

Unter Windows NT kann man mehrere WINS-Server einsetzen, die sich gegenseitig abstimmen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Diese Replikation stellt sicher, daß die Clients unabhängig von der Anzahl der WINS-Server nur eine einzige Namensdatenbank sehen. Die WINS-Server stellen sich somit gegenüber den Clients als eine konsistente Datenbank dar.

Die Abfrage eines WINS-Servers durch `nmblookup` erfolgt beispielhaft folgendermaßen:

```
nmblookup -R -U 192.168.1.5 samba
```

Hiermit wird der WINS-Server, der auf dem Rechner 192.168.1.5 liegt, nach dem Namen `samba` befragt.

Samba kennt zwei zusätzliche Funktionen, die es im Zusammenhang mit WINS interessant machen. Einerseits kann Samba als WINS Proxy eingerichtet werden, indem `wins proxy = yes` gesetzt wird. Ist diese Einstellung aktiv, dann wird Samba sämtliche Reservierungen und Anfragen, die es aus dem lokalen Netz per Broadcast erhält, an den mit `wins server =` konfigurierten WINS-Server weiterleiten. Stellt man mit dieser Einstellung einen Samba-Server in ein Subnetz, werden sämtliche Rechner in diesem Netz werden nun beim WINS angemeldet, und nutzen diesen auch. Dies ist auch dann der Fall, wenn sie entweder selbst keinen WINS-Server ansprechen können oder nicht dafür konfiguriert sind. Man sollte jedoch in jedem Fall eine echte Konfiguration des WINS-Servers auf dem Client vorziehen. Ein WINS Proxy kann nur eine Behelfslösung sein, da man sich damit auf einen weiteren Rechner verläßt.

Unter Windows kann man statische Einträge im WINS vornehmen. Dies geht so direkt unter Samba nicht. Man muß hierzu den Parameter `dns proxy = yes` auf dem WINS-Server setzen. Empfängt der WINS Server nun eine Anfrage, die er nicht aus seiner Datenbank beantworten kann, wird er eine ganz normale Unix-Hostnamenanfrage machen. Typischerweise wird er in der `/etc/hosts` nachschauen und danach dann das DNS anhand der Konfiguration in der Datei `/etc/resolv.conf` befragen. Damit ist es durch einen Eintrag auf dem WINS-Server möglich, den gesamten DNS-Namensraum auch in der NetBIOS-Namenswelt zur Verfügung zu stellen.

9 Windows-Namensauflösung im Detail

Um die Namensauflösung unter Windows zu verstehen, muß man zwei Arten von Anwendungen unterscheiden:

NetBIOS-Anwendungen: Dies sind die klassischen Windows-Programme, zum Beispiel um Laufwerke mit einem Server zu verbinden, oder um Outlook mit dem Exchange-Server zu verbinden. Die gesamte Netzwerkumgebung gehört ebenfalls zu den NetBIOS-Anwendungen.

TCP/IP-Anwendungen: Telnet, ping und Netscape gehören zu den Anwendungen, die es nur in der TCP/IP-Protokollfamilie gibt. Bei diesen funktioniert die Namensauflösung etwas anders als bei den NetBIOS-Anwendungen.

Wenn eine **NetBIOS-Anwendung** einen Namen auflösen will, dann geschieht dies in mehreren Schritten, die nacheinander ausgeführt werden, bis der Name gefunden ist.

1. Das System schaut im NetBIOS-Namenscache nach. Dieser kann durch `nbtstat -c` vom Benutzer abgefragt werden.

2. Ist ein WINS-Server konfiguriert, so wird dieser befragt.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

3. Kann der Name im WINS nicht aufgelöst werden, so wird eine Broadcast-Anfrage ausgelöst.
4. Es wird in der Datei `LMHOSTS` nachgesehen.
5. Sofern in den Eigenschaften von TCP/IP die DNS-Auflösung für NetBIOS-Namen aktiviert ist, wird nun an das Auflösungssystem für TCP/IP-Anwendungen übergeben.

Wenn man Namen in die Datei `LMHOSTS` einträgt, so werden diese erst nach den WINS- und Broadcast-Timeouts berücksichtigt. Wenn man diese sofort aufgelöst haben möchte, so kann man sie mit dem Zusatz `#PRE` versehen. Dann werden sie beim nächsten Reboot dauerhaft in den NetBIOS-Namenscache geladen. Im laufenden Betrieb kann man dieses Laden in den Namenscache durch ein `nbtstat -R` erzwingen.

Setzt man für die IP-Adreßvergabe DHCP ein, kann man Windows-Clients die IP-Adresse des WINS-Servers auf diesem Weg mitteilen. Tut man dies, muß man den Clients ebenfalls einen Knotentyp zuweisen. Die oben beschriebene Tabelle gilt für den Knotentyp 8, den sogenannten H-Knoten. Setzt man den Knotentyp auf 4, so bekommt man einen M-Knoten, der zuerst Broadcast und dann WINS ausführt. Diese Einstellung ist jedoch nur in Ausnahmefällen sinnvoll, da jede Anfrage beim WINS die Broadcastlast im Netz reduziert.

Die Namensauflösung für **TCP/IP-Anwendungen** ist einfacher.

1. Zunächst wird in der Datei `HOSTS` nachgesehen.
2. Ist ein DNS-Server konfiguriert, wird dieser befragt.
3. Der DNS-Name wird, so wie er ist, an die NetBIOS-Namensauflösung übergeben. Damit kann für interne Systeme vermeiden, sie ins DNS aufnehmen zu müssen. Will man etwa einen Proxy unter dem Namen „proxy“ einrichten, genügt es, auf dieser Maschine einen korrekt konfigurierten `nmbd` zu installieren, der den Namen „proxy“ registriert. Damit kann man auf allen Browsern einfach „proxy“ eintragen.

Die Namensauflösung von Samba ist weit weniger kritisch als die von Window-Systemen, da Samba in der Regel ausschließlich als Server auftritt. Samba als Server ist es gleichgültig, wie Namen aufgelöst werden können. Es gibt zwei Situationen, in denen Samba Namen auflösen muß:

smbclient Samba als Client muß offensichtlich Namen auflösen.

Samba als Domänenmitglied Mit dem Parameter `password server` wird Samba als Domänenmitglied mitgeteilt, welcher Domänencontroller für Paßwörter zuständig ist. Es ist enorm wichtig, daß für diese Funktion die Namensauflösung korrekt funktioniert.

Wie Windows kennt Samba vier Mechanismen zur Namensauflösung: Broadcast, WINS, LMHOSTS und die normale Unix-Namensauflösung. Die Reihenfolge, in der die Mechanismen abgefragt werden, wird durch den Parameter `name resolve order` festgelegt. Mit den vier Werten `bcast`, `wins`, `lmhosts` und `host` werden die vier Mechanismen beschrieben. Die Standardreihenfolge ist

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

und legt fest, daß vor der Windows-Namensauflösung zunächst das DNS befragt wird. Dies ist häufig ein Problem für `smbclient`, da man möglicherweise auf einen DNS-Timeout warten muß, bevor die Windows-Namensauflösung benutzt wird. In vielen Fällen kann es von Vorteil sein, für Samba als Client vollständig auf die DNS-Namensauflösung zu verzichten oder sie ans Ende der Liste zu stellen:

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

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

10 Browsing über Subnetzgrenzen

So, wie die Netzwerkumgebung in Abschnitt 7 betrachtet wurde, funktioniert sie nur in einem einzigen lokalen Netz. Die Wahl zum Local Master Browser funktioniert per Datagramm, das an den Namen `arbeitsgruppe<1e>` gesendet wird. `arbeitsgruppe<1e>` ist ein Gruppenname, der von mehreren Rechnern reserviert sein kann. Das heißt, daß ein Datagramm an diesen Namen mehrere Rechner erreichen muß. Dies geschieht bei NetBIOS über TCP/IP mit einem UDP-Paket an die Broadcastadresse im lokalen Netz. Allein hieraus ergibt sich, daß es pro Arbeitsgruppe in jedem Subnetz einen eigenen LMB geben muß. Jeder LMB bekommt aus seinem Subnetz die Informationen über vorhandene Server.

Um diese Einschränkung zu umgehen, gibt es den Domain Master Browser (DMB). Der DMB ist ein Rechner, der die Serverlisten von allen LMBs einsammelt und auf Anforderung wieder herausgibt. Dabei sitzt der DMB nur passiv da und wartet darauf, daß sich ein LMB mit ihm synchronisieren will. Es ist Aufgabe der LMBs, sich regelmäßig danach zu erkundigen, wo der DMB sitzt, und mit diesem dann die Serverlisten abzugleichen.

Die Vorgänge werden am deutlichsten, wenn man ein Beispiel betrachtet. Dieses Beispiel ist im wesentlichen der Originaldokumentation von Samba aus der Datei `BROWSING.txt` entnommen.

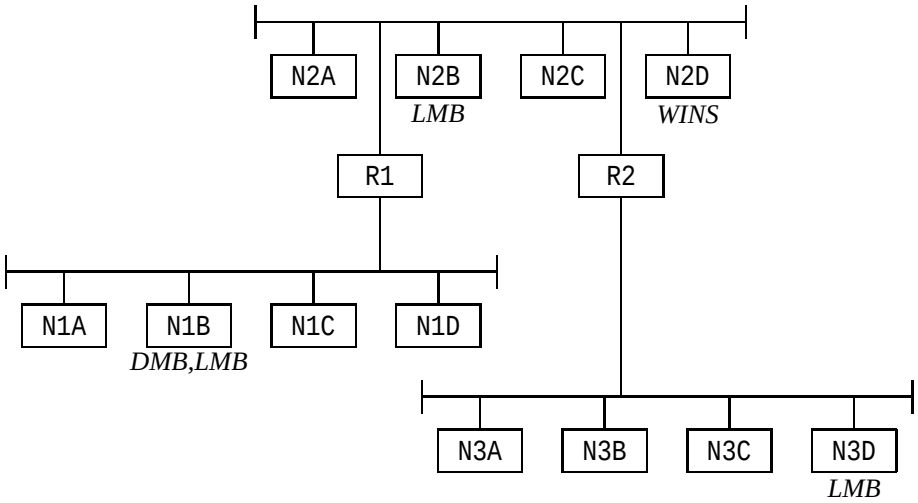


Abbildung 4: Domain Master Browser

Dieses Netz besteht aus drei Subnetzen (1,2,3), die durch zwei Router (R1 und R2) verbunden sind. Die Router lassen keine Broadcasts durch. Alle Subnetze bestehen aus jeweils vier Maschinen. Nehmen wir der Einfachheit halber an, daß alle Maschinen in der gleichen Arbeitsgruppe konfiguriert sind. Rechner `N1B` im Subnetz 1 ist als Domain Master Browser konfiguriert. Das heißt, daß er die Browserliste für die ganze Arbeitsgruppe aufsammelt. Rechner `N2D` ist als WINS Server konfiguriert und alle anderen Maschinen registrieren ihre NetBIOS Namen dort.

Wenn alle diese Maschinen gebootet werden, werden in jedem der drei Subnetze Wahlen um einen Local Master Browser abgehalten. Nehmen wir an, im Subnetz 1 gewinnt `N1B`, im Subnetz 2 gewinnt `N2B` und im Subnetz 3 gewinnt `N3D`. Diese Maschinen sind als Local Master Browser in ihrem Subnetz bekannt. Im Subnetz 1 liegen der LMB und der DMB auf der gleichen Maschine, was nicht der Fall sein muß. Diese beiden Rollen sind vollständig unabhängig voneinander.

Alle Maschinen, die Serverdienste anzubieten haben, kündigen dies per Broadcast auf ihrem Sub-

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

netz an. Der Local Master Browser in jedem Subnetz empfängt diese Broadcasts und trägt alle Server in einer Liste ein. Diese Liste von Einträgen ist die Basis für die Browserliste. In unserem Fall nehmen wir an, daß alle Maschinen Serverdienste anbieten, das heißt, daß alle Maschinen in der Liste erscheinen.

Für jedes Subnetz wird der Local Master Browser als *maßgeblich* angesehen, und zwar für alle Namen, die er per lokalem Broadcast empfängt. Broadcasts verlassen das Subnetz nicht, und die Broadcasts im lokalen Subnetz werden als maßgeblich angesehen. Daher wird dem Local Master Browser bei diesen Servern geglaubt. Rechner, die sich in anderen Subnetzen befinden, und über die der Local Master Browser von anderen Local Master Browsern informiert wurde, werden als nicht maßgeblich angesehen.

An diesem Punkt sieht die Browse Liste folgendermaßen aus: (dies sind die Maschinen, die Sie in Ihrer Netzwerkumgebung sehen würden, wenn Sie sie in einem bestimmten Subnetz ansehen)

Netz	LMB	Liste
1	N1C	N1A, N1B, N1C, N1D
2	N2B	N2A, N2B, N2C, N2D
3	N3D	N3A, N3B, N3C, N3D

An diesem Punkt sind alle Subnetze vollständig separat, keine Maschine wird in anderen Subnetzen gesehen. Die Microsoft-Dokumentation spricht davon, daß die Arbeitsgruppen in den Subnetzen getrennt sind.

Sehen wir uns nun Subnetz zwei an. Sobald N2B der Local Master Browser geworden ist, sucht er den Domain Master Browser, um mit ihm die Browse Listen zu synchronisieren. Dies tut er, indem er den WINS Server (N2D) nach der IP-Adresse fragt, die zum NetBIOS-Namen arbeitsgruppe<1B> gehört. Diesen Namen hat der Domain Master Browser (N1C) beim WINS-Server für sich beim booten registriert.

N2B kennt nun den Domain Master Browser. Er kündigt sich als Local Master Browser für Subnetz 2 bei ihm an. Dann synchronisiert N2B sich mit N2D, indem er einen NetServerEnum2-Aufruf abschickt. Der Domain Master Browser schickt alle Server, die er kennt, zurück. Sobald der Domain Master Browser die Ankündigung von N2B als Lokaler Master Browser erhalten hat, wird auch er sich mit dem Local Master Browser synchronisieren. Nachdem beide Synchronisationen stattgefunden haben, sehen die Browse Listen so aus:

Netz	LMB	Liste
1	N1C	N1A, N1B, N1C, N1D N2A*, N2B*, N2C*, N2D*
2	N2B	N2A, N2B, N2C, N2D N1A*, N1B*, N1C*, N1D*
3	N3D	N3A, N3B, N3C, N3D

Die mit * bezeichneten Einträge werden als nicht maßgeblich angesehen, da sie von anderen Master Browsern erhalten wurden. Für den Client macht dies jedoch keinen Unterschied. Nur der LMB darf diese Einträge selbstverständlich beim nächsten Abgleich nicht an den DMB als seine eigenen zurückmelden.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Zu diesem Zeitpunkt werden Benutzer in den Subnetzen 1 und 2, die die Netzwerkumgebung ansehen, die Server in beiden Subnetzen sehen, Benutzer im Subnetz 3 sehen immer noch nur die Server in ihrem eigenen Subnetz.

Der lokale Master Browser im Subnetz 3 (N3D) macht nun exakt das gleiche wie N2B. Wenn er die Browse Listen mit dem Domain Master Browser (N1B) abgeglichen hat, bekommt er sowohl die Server in Subnetz 1, als auch die im Subnetz 2. Nachdem sich N3D mit N1C synchronisiert hat und umgekehrt, sehen die Browse Listen folgendermaßen aus:

Netz	LMB	Liste
1	N1C	N1A, N1B, N1C, N1D N2A*, N2B*, N2C*, N2D* N3A*, N3B*, N3C*, N3D*
2	N2B	N2A, N2B, N2C, N2D N1A*, N1B*, N1C*, N1D*
3	N3D	N3A, N3B, N3C, N3D N1A*, N1B*, N1C*, N1D* N2A*, N2B*, N2C*, N2D*

Jetzt sehen Benutzer in den Subnetzen 1 und 3 alle Server in allen Subnetzen, Benutzer im Subnetz 2 sehen jedoch immer noch nur die Server von Subnetz 1 und 2, nicht jedoch die im Subnetz 3.

Zum guten Schluß wird sich der lokale Master Browser im Subnetz 2 (N2B) erneut mit dem Domain Master Browser abstimmen, und die fehlenden Servereinträge bekommen. Endlich sehen die Browse Listen als stabiler Zustand so aus:

Netz	LMB	Liste
1	N1C	N1A, N1B, N1C, N1D N2A*, N2B*, N2C*, N2D* N3A*, N3B*, N3C*, N3D*
2	N2B	N2A, N2B, N2C, N2D N1A*, N1B*, N1C*, N1D* N3A*, N3B*, N3C*, N3D*
3	N3D	N3A, N3B, N3C, N3D N1A*, N1B*, N1C*, N1D* N2A*, N2B*, N2C*, N2D*

Synchronisationen zwischen dem Domain Master Browser und den Local Master Browsern wird weiterhin auftreten, aber dies sollte den stabilen Zustand nur bestätigen.

Wenn Router R1 oder R2 ausfallen, wird das folgende passieren:

1. Namen der Computer auf beiden Seiten der nicht mehr erreichbaren Subnetze werden für 36 Minuten weiter in den Browse Listen gehalten, so daß sie in der Netzwerkumgebung weiterhin erscheinen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

2. Versuche, Verbindungen zu diesen Rechnern aufzubauen, werden scheitern, aber die Namen werden nicht von den Browse Listen entfernt werden.

3. Wenn ein Subnetz vom WINS-Server getrennt wird, wird es nur noch auf die lokalen Server zugreifen können, deren Namen mit lokaler Broadcast NetBIOS-Namensauflösung aufgelöst werden können. Das ist vergleichbar mit der Situation, keinen Zugriff auf einen DNS Server mehr zu haben.

10.1 Browsing mit vielen Arbeitsgruppen

Wenn man in der Netzwerkumgebung auf das Microsoft Windows Netzwerk klickt, bekommt man eine Liste sämtlicher Arbeitsgruppen im Netz angezeigt. Diese Liste der Arbeitsgruppen wird vom Local Master Browser vorgehalten. Wie bekommt er diese Liste?

Jeder Local Master Browser reserviert für sich einen speziellen Gruppennamen, der folgendermaßen dargestellt wird: `..__MSBROWSE__.<01>`. Die Punkte stehen dabei für die Ascii-Werte eins und zwei. Regelmäßig wird jeder Local Master Browser seine Existenz an diesen Gruppennamen senden. Alle anderen Local Master Browser im Netz sammeln diese Ankündigungen, damit sie Clients die Liste der vorhandenen Arbeitsgruppen und Local Master Browser mitteilen können.

Wenn Domain Master Browser ins Spiel kommen, wird das Bild etwas komplizierter. Samba hat Erweiterungen implementiert, mit denen das Browsing über Subnetzgrenzen stabiler gemacht werden soll. Samba fragt den WINS-Server regelmäßig nach allen Domain Master Browsern. Diese werden in zufälligen Abständen kontaktiert, um die Browse Listen mit ihnen abzugleichen. Dadurch kann es passieren, daß Arbeitsgruppen, die nicht mehr existieren, weiterhin in der Netzwerkumgebung auftauchen und sich nicht löschen lassen. Samba kennt den Parameter `enhanced browsing = no`, mit dem sich dieses Verhalten abstellen läßt.

11 Virtuelle Sambaserver

Manchmal kann es notwendig sein, mehr als einen Sambaserver gleichzeitig auf einem Rechner laufen zu lassen. Zur Serverkonsolidierung kann es notwendig sein, unter mehreren Namen in der Netzwerkumgebung zu erscheinen. Dies ist mit dem Parameter `netbios aliases` sehr einfach möglich. Wenn es nötig ist, in mehr als einer Arbeitsgruppe aufzutauchen, dann scheitert dies Verfahren jedoch, da der Parameter `workgroup` nur einmal angegeben werden kann.

Eine andere Konfiguration ist die Einbindung von virtuellen Servern in eine Hochverfügbarkeitsumgebung. Es kann wünschenswert sein, zwei physikalisch vorhandene Server unabhängig voneinander arbeiten und sich gegenseitig überwachen zu lassen. Jeder der beiden Server hat seinen eigenen Namen und seine eigenen Freigaben. Stellt ein Server fest, daß sein Partner defekt ist, muß er dessen Aufgaben übernehmen. Dies ist am einfachsten möglich, wenn die Aufgaben des defekten Servers isoliert in einer eigenen Samba-Instanz wahrgenommen werden. Die Hochverfügbarkeitssoftware muß nur dafür sorgen, daß die Platten übernommen werden und der ausgefallene Dienst auf dem noch lebenden Server gestartet wird. Es ist keine Neukonfiguration des bereits laufenden Servers notwendig.

Hier soll ein Beispiel aufgebaut werden, mit dem Samba auf einem Rechner für verschiedene Arbeitsgruppen Local Master Browser wird. Ist dieser Rechner ein Unixserver, der 24 Stunden durchläuft, kann so mit sehr einfachen Mitteln eine recht stabile Netzwerkumgebung für beliebig viele Arbeitsgruppen erreicht werden.

Zunächst wird ein isolierter Local Master Browser für die Arbeitsgruppe `GOETTINGEN` installiert. Der Name dieses Rechners soll der Einfachheit halber `GOE` heißen. Die gesamte Konfiguration wird

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

unter /samba/goe abgelegt, so daß sie recht einfach duplizierbar ist. Die Datei /samba/goe/smb.conf hat folgenden Aufbau:

```
[global]
workgroup = goettingen
netbios name = goe

interfaces = eth0:1
bind interfaces only = yes

encrypt passwords = yes
smb passwd file = /samba/goe/smbpasswd

log file = /samba/goe/var/log.smb
lock directory = /samba/goe/locks

os level = 100
preferred master = yes
```

In dieser Konfigurationsdatei gibt es einige Einstellungen, die die Voreinstellungen vom Kompilieren überschreiben. Normalerweise finden sich die Logdateien unter Linux in /var/log oder bei selbstkompilierten Sambas in /usr/local/samba/var, hier sollen sie pro Server separat in einem eigenen Verzeichnis abgelegt werden.

Die nicht offensichtlichen Einstellungen bedeuten:

bind interfaces only: Normalerweise nimmt der smbd auf jeder im System konfigurierten IP-Adresse Verbindungen entgegen. Den Vorgang, mit dem der smbd dies dem Kernel mitteilt, nennt sich „An eine Adresse binden“. Um auf jeder Adresse Verbindungen entgegen zu nehmen, bindet Samba an die spezielle Adresse 0. Jede konfigurierte IP-Adresse kann nur von einem einzigen Prozeß gebunden werden. Versucht ein smbd, eine bereits verwendete Adresse zu binden, wird dies mit der Fehlermeldung **Address already in use** verweigert. Mit bind interfaces only = yes wird der smbd nur die im Parameter interfaces angegebenen Adressen beziehungsweise Interfaces verwenden.

Der Unterschied wird im Vergleich zweier Ausgaben des Programms netstat -nat (hier unter Linux) deutlich. Zunächst der relevante Teil *ohne* bind interfaces only = yes:

```
vlendec@server:~ > netstat -natu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN

vlendec@server:~/ >
```

Im Vergleich dazu die Ausgabe des gleichen Programmaufrufs *mit* bind interfaces only = yes:

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
vlendec@server:~ > netstat -natu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 192.168.42.1:139        0.0.0.0:*               LISTEN

vlendec@server:~/ >
```

Mit `bind interfaces only = yes` wird ausschließlich an die im Parameter `interfaces` referenzierte IP-Adresse gebunden, so daß sich mehrere `smbd`s nicht stören.

log file: Hier wird das Logfile nur für den `smbd` festgelegt. Es ist möglich, für alle Samba-Instanzen ein gemeinsames Logfile zu verwenden, das kann jedoch sehr schnell unübersichtlich werden. Der `nmbd` ignoriert diese Einstellung. Sein Logfile muß über den Kommandozeilenparameter `-l` festgelegt werden.

lock directory: Die verschiedenen Dämonen von Samba kommunizieren über viele Datenbanken miteinander. Sie haben die Endung `.tdb`, und ihr Verzeichnis ist durch das `lock directory` festgelegt. Jede Instanz von Samba benötigt ihr eigenes `lock directory`, da die Datenbanken jeweils nur für eine Samba-Instanz ausgelegt sind.

Diese Samba-Instanz kann über die folgende Startdatei kontrolliert werden:

```
#!/bin/sh
DIR=/samba/goe
case "$1" in
  start)
    echo "Starte Samba in $DIR"
    /usr/local/samba/bin/smbd -D -s $DIR/smb.conf
    /usr/local/samba/bin/nmbd -D -s $DIR/smb.conf -l $DIR/var
    ;;
  stop)
    echo "Fahre Samba in $DIR herunter"
    kill -TERM $(cat $DIR/locks/smbd.pid)
    kill -TERM $(cat $DIR/locks/nmbd.pid)
    ;;
  *)
    echo "Usage: $0 [start|stop]"
    ;;
esac
```

Diese Installation von Samba ist so weit isoliert, daß eine zweite ungestört gleichzeitig laufen kann. Um jetzt eine zweite Installation zu bauen, müssen folgende Dinge angepaßt werden:

workgroup: Die Arbeitsgruppe muß nur in dem aktuell verwendeten Beispiel der Local Master Browser geändert werden. Ein zweites Samba kann selbstverständlich auch in der gleichen Arbeitsgruppe sein.

netbios name: Jede Instanz braucht zwingend ihren eigenen Namen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

interfaces: Jede Instanz benötigt ihre eigene IP-Adresse.

smb passwd file: Falls jede der Instanzen ihre eigene Benutzerdatenbank möchte, so muß die Datei `smbpasswd` separat angelegt werden. Die Unix-Benutzerdatenbank teilen jedoch alle Instanzen. Das heißt, Benutzer **meier** auf der einen Instanz wird immer der gleiche Unixbenutzer wie Benutzer **meier** auf allen anderen Instanzen sein. Wenn man die gleiche Benutzerdatenbank benötigt, kann man auf die gleiche `smbpasswd` zugreifen. Empfehlenswerter ist es jedoch, eine der beiden Instanzen als Domänencontroller einzurichten und die andere als Domänenclient. Dann kann man völlig ohne Unterbrechung die gesamte Konfiguration komplett auf einen anderen Rechner migrieren, ohne daß irgend etwas geändert werden müßte. Insbesondere für Hochverfügbarkeitslösungen ist dies die Konfiguration der Wahl.

log file: Dies kann für alle Instanzen gleich sein, meistens wird man jedoch separate logfiles für die einzelnen `smbd`s haben wollen.

lock directory: Dieses muß zwingend für jede Instanz separat angelegt werden.

Als letztes ist die Variable DIR in der Startdatei anzupassen, und mehreren Instanzen von Samba steht nichts mehr im Wege.

12 Browsing im WAN – schneller

Das im Kapitel 10 beschriebene Verfahren, mit dem über Subnetzgrenzen hinweg die Netzwerkumgebung gepflegt wird, ist außerordentlich träge. Jede Änderung muß vom Local Master Browser an den Domain Master Browser übergeben werden und von dort aus wieder an die anderen Local Master Browser zurück. Bis diese Änderung beim Client ankommt, kann es sehr lange dauern.

Zudem ist bei einem komplexen Setup die Zahl der beteiligten Rechner sehr hoch. Als Beispiel sei ein Netz auf 4 Subnetze verteilt. Jeder Mitarbeiter ist einer von 5 verschiedenen Arbeitsgruppen zugeteilt. Nun ist es gefordert, daß die Mitarbeiter sich im Netz frei bewegen können müssen, daß sie also unabhängig von ihrem Standort im Netz immer ihre eigene Arbeitsgruppe vorfinden müssen. Dazu muß selbstverständlich ein WINS-Server eingerichtet sein. Damit das Browsing funktioniert, muß es zudem für jede Arbeitsgruppe einen Domain Master Browser geben, der sich mit den jeweiligen Local Master Browsern abgleicht. Die Zahl der Local Master Browser ist hier recht hoch. Da jeder Mitarbeiter in jedem Subnetz seine Arbeitsgruppe sehen soll, muß es in jedem Subnetz für jede Arbeitsgruppe einen eigenen Local Master Browser geben. Das heißt, es werden 20 Local Master Browser benötigt.

Um das folgende Beispiel zu verstehen, sollte man sich vergegenwärtigen, von welchen Rechnern welche Information bezogen wird, wenn man im Explorer die Netzwerkumgebung durchklickt. Man kann die Vorgänge sehr gut nachvollziehen, wenn man an einer frisch angemeldeten Sitzung mit `nbtstat -s` die aktiven NetBIOS-Sitzungen nach jedem Schritt nachvollzieht. Direkt nach dem anmelden sollte keine NetBIOS-Sitzung aktiv sein, mit jedem Klick in der Netzwerkumgebung kommt gegebenenfalls eine Verbindung hinzu.

Netzwerkumgebung: Hier wird die eigene Arbeitsgruppe dargestellt. Diese Information liefert der eigene Local Master Browser. Dieser wird über eine Broadcast-Anfrage auf den Namen der eigenen Arbeitsgruppe vom Typ `<#1d>` herausgefunden.

Gesamtes Netzwerk: Dieser Schritt liefert nur die lokal installierten Clientsysteme. Wenn ein Novell-Client installiert ist, wird hier das Novell-Netz neben dem Microsoft Windows-Netzwerk ange-

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

boten, ansonsten nur das Microsoft-Windows Netzwerk. Da dies rein lokal passiert, wird es keine zusätzliche Verbindung geben.

Microsoft Windows Netzwerk: Hier wird die Liste der verfügbaren Arbeitsgruppen angezeigt. Diese Information liefert ebenfalls der eigene Local Master Browser. Das kann man sich mit einem `smbclient -L` *lmb* verdeutlichen. Neben der Liste der Freigaben und der Server liefert der LMB eine Liste der Arbeitsgruppen, die er kennt. Zusätzlich gibt er noch den jeweils zuständigen Local Master Browser heraus.

Arbeitsgruppe: Diese Information liefert der jeweilige Local Master Browser. Der eigene Local Master Browser hat im letzten Schritt dessen Namen herausgegeben. Dessen IP-Adresse findet der Client durch eine normale NetBIOS-Namensanfrage heraus.

Freigabeliste: Ein Rechner ist für seine Freigaben selbst verantwortlich, nur der Rechner selbst kann die Liste der von ihm freigegebenen Verzeichnisse herausgeben.

Gibt ein Rechner Informationen der genannten Art heraus, dann geschieht dies über eine vollständig aufgebaute SMB-Sitzung, die auf einer NetBIOS-Sitzung aufbaut. Kapitel 14 beschreibt dies im Detail. Wie auf Seite 10 dargestellt, nutzt der NetBIOS-Sitzungsdienst TCP über Port 139.

12.1 Trennung von `nmbd` und `smbd`

Die folgende Situation läßt sich erheblich einfacher und stabiler lösen als mit einem Domain Master Browser. Das Beispielnetz besteht aus zwei Filialen einer Firma in Göttinen und Heidelberg. Für das später vollständig aufgebaute Beispiel seien die beiden Netze 192.168.1.0/24 in Göttingen und 192.168.2.0/24 in Heidelberg vergeben.

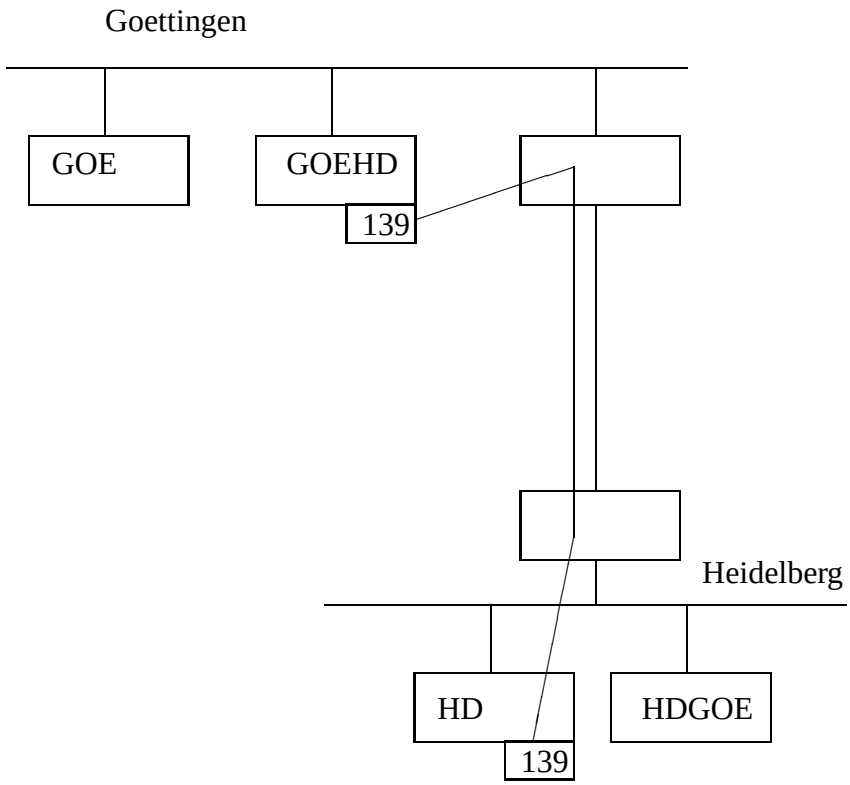
In jeder Filiale gibt es eine Arbeitsgruppe, also die Gruppen `GOETTINGEN` und `HEIDELBERG`. In Göttingen stehen nur Rechner der Arbeitsgruppe `GOETTINGEN`, in Heidelberg nur Rechner der Arbeitsgruppen `HEIDELBERG`. Nun soll auf beiden Seiten jeweils die eigene und die entfernte Arbeitsgruppe sichtbar sein, um sich im Netz mit dem Explorer frei bewegen zu können. Dazu muß es sowohl in Göttingen als auch in Heidelberg jeweils einen Local Master Browser für `GOETTINGEN` und `HEIDELBERG` geben. Es gibt im Beispiel vier Local Master Browser, die hier auch bereits mit IP-Adressen versehen wurden:

	`GOETTINGEN`	`HEIDELBERG`
Ort: Göttingen	`GOE`, 192.168.1.1	`GOEHD`, 192.168.1.2
Ort: Heidelberg	`HDGOE`, 192.168.2.2	`HD`, 192.168.2.1

Die Idee für die Konfiguration ist nun, die Göttinger Anfragen an den Local Master Browser für `HEIDELBERG` (Rechner `GOEHD`) direkt nach Heidelberg an den Rechner `HD` umzuleiten. In Göttingen muß nur ein `nmbd` behaupten, er sei Local Master Browser für die Arbeitsgruppe `HEIDELBERG`. Dies tut er, indem er auf UDP Port 137 die NetBIOS-Namensanfragen für `HEIDELBERG#1D` beantwortet. Der TCP-Port 139 auf dem Rechner `GOEHD` in Göttingen wird dann an den echten Local Master Browser `HD` weitergeleitet.

Das Weiterleiten von TCP Port 139 auf dem Rechner `GOEHD` an Port 139 des Rechners `HD` kann unterschiedlich geschehen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)



12.2 Konfiguration

Als Beispiel soll hier die vollständige Konfiguration am Standort Göttingen mit beiden Local Master Browsern beschrieben werden, die am Standort Heidelberg kann dann spiegelverkehrt aufgesetzt werden.

Der Local Master Browser in Göttingen hat die beiden IP-Adressen 192.168.1.1 (Interface eth0) für den LMB der Arbeitsgruppe `GOETTINGEN` und 192.168.1.2 (Interface eth0:1) für die Arbeitsgruppe `HEIDELBERG`. Die Interface-Bezeichnungen sind hier Linux-spezifisch. Andere Unix-Versionen vergeben virtuelle IP-Adressen möglicherweise anders. Die beiden virtuellen Sambaserver werden mit ihren Konfigurationen in den Verzeichnissen `/samba/goe` und `/samba/goehd` abgelegt.

Es müssen nun zwei Dateien `smb.conf` erstellt werden, für jeden Local Master Browser eine. Für die Arbeitsgruppe `GOETTINGEN` kann direkt die `smb.conf` von Seite 27 verwendet werden. Nur die Zeile `interfaces =` muß angepaßt werden, so daß sich die folgende `/samba/goe/smb.conf` ergibt:

```
; /samba/goe/smb.conf
[global]
workgroup = goettingen
netbios name = goe
interfaces = eth0
bind interfaces only = yes
encrypt passwords = yes
smb passwd file = /samba/goe/smbpasswd
log file = /samba/goe/var/log.smb
lock directory = /samba/goe/locks
os level = 100
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
preferred master = yes
```

Entsprechend ist die Datei `/samba/goehd/smb.conf` aufgebaut. Um der Kürze willen sind sämtliche Einstellungen, die ausschließlich den `smbd` betreffen, weggelassen worden. In Göttingen soll für die Arbeitsgruppe `HEIDELBERG` kein `smbd` gestartet werden, dafür ist der `smbd` auf dem Rechner `HDGOE` in Heidelberg zuständig.

```
; /samba/goehd/smb.conf
[global]
workgroup = heidelberg
netbios name = goehd
interfaces = eth0:1
bind interfaces only = yes
lock directory = /samba/goe/locks
os level = 100
preferred master = yes
```

Die Startdatei für die Local Master Browser kann folgendermaßen aussehen. Es werden drei Prozesse gestartet, ein vollständiges Samba für den Rechner `GOE` und nur den `nmbd` für `GOEHD`.

```
#!/bin/sh
SMBD=/usr/local/samba/bin/smbd
NMBD=/usr/local/samba/bin/nmbd
case "$1" in
  start)
    echo "Starte Samba"
    $SMBD -D -s /samba/goe/smb.conf
    $NMBD -D -s /samba/goe/smb.conf
    $NNBD -D -s /samba/goehd/smb.conf -l /samba/goehd/var
    ;;
  stop)
    echo "Fahre Samba herunter"
    kill -TERM $(cat /samba/goe/locks/smbd.pid)
    kill -TERM $(cat /samba/goe/locks/nmbd.pid)
    kill -TERM $(cat /samba/goehd/locks/nmbd.pid)
    ;;
  *)
    echo "Usage: $0 [start|stop]"
    ;;
esac
```

Die Weiterleitung des TCP-Ports 139 von der IP-Adresse 192.168.1.2 in Göttingen an die Adresse 192.168.2.1 Port 139 in Heidelberg kann mit unterschiedlichen Methoden geschehen. Die einfachste Methode mit dem Programm `netcat` und dem `inetd` funktioniert hier leider nicht, da dem `inetd` leider nicht gesagt werden kann, daß er bitte nur an ein spezielles Interfaces binden soll. Gäbe es für den Rechner `GOEHD` eine eigene Maschine, könnte man den `inetd` jedoch problemlos verwenden. Die Zeile

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
netbios-ssn stream tcp nowait nobody /usr/bin/netcat netcat 192.168.2.1 139
```

in der `/etc/inetd.conf` zusammen mit

```
netbios-ssn     139/tcp
```

in der `/etc/services` leiten eingehende TCP-Verbindungen auf Port 139 zum Port 139 des Rechners 192.168.2.1 weiter.

Die zweite Möglichkeit der Portweiterleitung bietet das Programm `rinetd`. Der `rinetd` ist für genau diesen Zweck geschaffen worden und ist bei SuSE-Linux als fertiges Paket mitgeliefert. Im Gegensatz zum `inetd` kann der `rinetd` an spezielle Interfaces binden, so daß sein Einsatz auch mit virtuellen Sambaservern möglich ist. Der `rinetd` wird über die Datei `/etc/rinetd.conf` konfiguriert. Die notwendige Datei besteht nur aus einer einzigen Zeile:

```
192.168.1.2 139 192.168.2.1
```

Alternative drei besteht beim Einsatz des `xinetd`, der den `inetd` vollständig ersetzt und erheblich leistungsfähiger ist. Der `xinetd` beherrscht einerseits das Binden an einzelne Interfaces, andererseits kennt er bereits die Möglichkeit, TCP-Verbindungen weiterzuleiten. Der Abschnitt in der Konfigurationsdatei `/etc/xinetd.conf` könnte beispielsweise so aussehen:

```
service goehd
{
   socket_type = stream
   protocol    = tcp
   wait        = no
   port        = 139
   redirect    = 192.168.2.1 139
   bind        = 192.168.1.2
}
```

Für welche der Alternativen man sich entscheidet, hängt von der Umgebung ab. Setzt man virtuelle Server ein, fällt der `inetd` aus. Die Entscheidung zwischen `rinetd` und `xinetd` wird vermutlich danach fallen, ob der eventuell vorhandene `inetd` abgelöst werden soll. Die Kombination von `inetd` und virtuellen Servern läßt nur die Wahl, den `rinetd` einzusetzen. Wird der `xinetd` bereits verwendet, sollte man ihn selbstverständlich auch für die Portweiterleitung nutzen.

13 Einfache Freigaben

Warum setzt man Samba überhaupt ein? Einer der wichtigsten Dienste von Samba ist, Festplattenbereiche für Clients zur Verfügung zu stellen. Damit ein Client Plattenplatz eines Servers erreichen kann, muß man eine sogenannte *Freigabe* erstellen.

Beispielsweise möchte man den Inhalt des Unix-CDROM-Laufwerks an Clients exportieren. Das Laufwerk sei unter `/cdrom` eingebunden, und soll für Clients unter `\\servername\cd` erreichbar sein. Dazu muß man in der `smb.conf` einen neuen Abschnitt einleiten, der den Namen `[cd]` trägt. Damit wird eine Freigabe eingeleitet, die im Netz unter dem Namen `cd` zu sehen ist.

Das folgende Beispiel gibt genau dieses Verzeichnis frei. Dabei ist zusätzlich die Zugriffskontrolle so angelegt, daß wirklich **jeder** darauf zugreifen kann. Wenn Sie irgend eine Art von schützenswerten

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Daten auf der exportierten CD haben, sollten Sie sich auf jeden Fall das Kapitel 15 zu Rechten an Freigaben und das Kapitel 14 ansehen, um die Freigabe sinnvoll schützen zu können.

```
[global]
workgroup = arbeitsgruppe
interfaces = <IP-Adresse>/<Netzmaske>
security = share
encrypt passwords = yes

[cd]
path = /cdrom
guest ok = yes
```

14 SMB-Sitzungen

Sobald ein Rechner Freigaben im Netz zur Verfügung stellt, können Clients darauf zugreifen. Bevor ein Client tatsächlich auf eine Freigabe zugreifen kann, werden sechs Schritte durchlaufen. Diese sechs Schritte im Detail zu verstehen, ist für die Konfiguration einfacher Server nicht wirklich notwendig. Sobald es aber darum geht, Fehlerdiagnose zu betreiben, ist das Wissen um die genaue Fehlerursache sehr wertvoll. Die genaue Stelle, an der eine Freigabeverbindung scheitert, kann bei der Fehlersuche gute Hinweise geben.

14.1 NetBIOS-Namensauflösung

Ein Benutzer an einem Client gibt den Namen des Servers mit unterschiedlichen Methoden an. Ein typischer Weg geht über die Netzwerkumgebung über einen Doppelklick auf den Rechner. Das Erscheinen in der Netzwerkumgebung ist jedoch nicht notwendig, da ein Client auch auf der Kommandozeile über ein

```
net use h: \\server\freigabe
```

das Laufwerk H verbinden kann. Genauso kann im Explorer durch den Menüpunkt „Netzwerklaufwerk verbinden“ eine direkte Verbindung geöffnet werden. Ein weiterer Weg ist über Menüpunkt „Ausführen“ im Startmenü von Windows 95. Wenn man dort `\\server` angibt, bekommt man die Liste der Freigaben des Servers angezeigt, unabhängig, in welcher Arbeitsgruppe sich der Server befindet.

14.2 TCP-Verbindung

Wenn die IP-Adresse klar ist, wird eine TCP-Verbindung zu Port 139 des Servers aufgebaut. Um vorhandene TCP-Verbindungen anzuzeigen, gibt es sowohl auf Unix- als auch auf Windowsrechnern das Werkzeug `netstat`.

Ob die TCP-Verbindung klappt, prüft man am besten mit

```
telnet <ip> 139
```

und einem Test mit `netstat`, ob die Verbindung im Zustand `ESTABLISHED` ist.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

14.3 NetBIOS-Sitzung

Auf einem Serverrechner arbeiten unter Umständen mehrere Applikationen, die Namen für sich reserviert haben. Diese sind alle unter der IP-Adresse des Rechners und dem TCP-Protokoll auf Port 139 erreichbar. Anhand des TCP-Verbindungsaufbaus ist nicht klar, welche Serverapplikation angesprochen werden soll. Die Unterscheidung wird durch den Servernamen getroffen, der in der TCP-Verbindung als erstes übertragen wird.

Daß der Servername übertragen wird, kann man ganz einfach mit Hilfe des Programms `smbclient` sehen. Man versucht, sich die Liste der Freigaben eines realen Windowsrechners geben zu lassen, indem man folgendes aufruft:

```
smbclient -L smallwin
```

Damit wird zunächst eine NetBIOS-Namensanfrage ausgelöst, und dann eine Verbindung zum entsprechenden Server ausgelöst. `smbclient` hat jedoch die Möglichkeit, einen Server unter einem anderen Namen anzusprechen, indem man

```
smbclient -L test -I ip-adresse
```

eingibt. `smbclient` wird zunächst versuchen, eine Verbindung zum NetBIOS-Namen `test` aufzubauen, und zwar ohne daß eine NetBIOS-Namensanfrage ausgelöst wird. Stattdessen wird die angegebene IP-Adresse auf Port 139 direkt angesprochen, und der Name `test` als Servername angegeben. Windows merkt, daß das nicht stimmen kann und verweigert den Verbindungsaufbau mit einer Fehlermeldung. Erst im zweiten Versuch wird es `smbclient` gelingen, eine Verbindung aufzubauen, da diese Verbindung zum allgemeinen Namen `*smbserver`[6] aufgebaut wird.

Auch der Clientname wird in der Verbindung übergeben. Dies testet man am besten mit

```
smbclient //win/c\$ -n blafasel
```

und schaut sich die Verbindungstabelle auf der Windowsmaschine mit `nbtstat -s` an.

Mit dem übergebenen Servernamen kann man sehr nette Tricks anstellen. Man stelle sich vor, daß einige Freigaben nur für bestimmte Clientrechner sichtbar sein sollen. Dies ist mit Bordmitteln von Samba so nicht möglich. Man kann zwar mit dem Parameter `browseable` festlegen, ob bestimmte Freigaben in der Netzwerkumgebung erscheinen. Dieser Parameter hat aber zwei Nachteile. Erstens sind die Freigaben nur unsichtbar geworden, darauf zugreifen kann man immer noch. Zweitens kann man Freigaben nur für alle Rechner verstecken oder freigeben.

Samba bietet die Option, unter zwei oder mehreren verschiedenen Namen in der Netzwerkumgebung zu erscheinen. Mit dem Parameter `netbios name` gibt man einen Namen für den Server an. Zusätzliche Namen kann man mit `netbios aliases` vergeben. Mit

```
netbios name = fichte
netbios aliases = birke eiche kiefer buche
```

handelt man sich einen ganzen Wald in der Netzwerkumgebung ein. Klickt man auf die einzelnen Server, sieht man überall die gleichen Freigaben und Zugriffsrechte. Nun kann man für jeden dieser virtuellen Rechner eine eigene Konfigurationsdatei anlegen. Beispielsweise kann man diese Dateien `/etc/smb.conf.birke`, `/etc/smb.conf.eiche` und so weiter nennen. Die Datei `/etc/smb.conf` ist für den Rechner `fichte` zuständig und enthält neben den Einstellungen für `fichte` den Parameter

```
config file = /etc/smb.conf.%L
```

[6] Das SMB-Protokoll wurde als Antwort auf das WebNFS von SUN in Common Internet File System umbenannt. Im Gegensatz zur Firma SUN, die tatsächlich das NFS-Protokoll verbessert hat, hat sich Microsoft die Arbeit einfacher gemacht. Der Name `*SMBSERVER` ist der einzige echte Unterschied, der CIFS von seinem Urvater SMB unterscheidet. Mit Windows 2000 werden diese NetBIOS-Namen beim Verbindungsaufbau gar komplett unterschlagen. Dafür war es aber notwendig, einen weiteren Port zu reservieren, und zwar Port 445.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Dabei steht `%L` für den Servernamen, unter dem Samba angesprochen wird. Wenn es eine passende Datei gibt, dann bewirkt der Parameter `config file`, daß die komplette Konfiguration neu eingelesen wird. Existiert keine passende Datei, so wird der Parameter einfach ignoriert. Um nun den Zugriff nur für einzelne Clients zu erlauben, kann bei den einzelnen virtuellen Servern mit den Parametern `hosts allow` und `hosts deny` der Zugriff geregelt werden.

14.4 Negotiate Protocol

Die NetBIOS-Sitzung ist nun aufgebaut, und es können Daten übermittelt werden. Innerhalb dieser NetBIOS-Sitzung wird eine SMB-Sitzung schrittweise aufgebaut. SMB ist ein Protokoll, bei dem im Prinzip der Client jede Aktion durch eine Anfrage anstößt, und der Server diese beantwortet[7].

SMB (Server Message Block) ist ein gewachsenes Protokoll. Es ist mit den Fähigkeiten der Betriebssysteme gewachsen, die damit arbeiten. Zunächst ist es entstanden, um die Dateisystemaufrufe der MS-DOS Systemschnittstelle INT 0x21 auf das Netz zu verlagern. Mit einer gewissen Weitsicht hat man jedoch vorausgesehen, daß die Entwicklung nicht bei MS-DOS stehen bleiben würde, sondern sich die Dateisystemaufrufe ändern würden. Man hat im Protokoll also eine Möglichkeit vorgesehen, mit der unterschiedliche Protokollvarianten ausgehandelt werden können. Die unterschiedlichen Protokolle orientieren sich immer an den Fähigkeiten der jeweiligen Betriebssysteme. Beispielsweise wurde mit dem LAN Manager, der eine Benutzerverwaltung besitzt, das Konzept des Benutzers im Protokoll aufgenommen. OS/2 hat ein recht weitgehendes Konzept der Druckerverwaltung, das entsprechend mit Protokollerweiterungen bedacht wurde. Sogar für XENIX gibt es einen eigenen Protokolldialekt, der das Unix-Zugriffsrechtekonzept im SMB-Protokoll abbildet. Diese Protokollvariante beherrscht nur leider kein moderner Client. Mit Ausnahme des ausgestorbenen XENIX-Dialektes lassen sich die Protokolle gut in eine Hierarchie einordnen. Spätere Protokolle beherrschen alle Aspekte der vorherigen Varianten.

Im Jahr 1996 wurde SMB in CIFS umbenannt. CIFS ist die Abkürzung für Common Internet File System. Warum diese neue Bezeichnung, und warum zu diesem Zeitpunkt? Kurz vorher hatte Sun Microsystems sein Protokoll NFS angepaßt, um über Weitverkehrsstrecken besser benutzbar zu sein. NFS setzt voraus, daß zwischen Client und Server nur sehr kurze Pingzeiten vorliegen. Für jeden Dateizugriff sind mehrere Anfragen notwendig. Auch wenn jede Anfrage nur sehr kurz ist und wenig Bandbreite verbraucht, muß doch jedesmal die Antwort des Servers abgewartet werden. Hohe Pingzeiten belasten so die Leistung des NFS erheblich. Sun hat das NFS so verändert, daß die Anzahl der Anfragen erheblich reduziert wurde. Das Ergebnis nannten sie WebNFS und haben um dieses „neue“ Protokoll eine große Marketinginitiative gestartet. Kurz vorher hatte Microsoft die Kröte namens Java von SUN schlucken müssen und wollte sich nicht ein zweites Mal von SUN eine Technologie aufzwingen lassen. Daher hat man einfach das hauseigene Datei- und Druckprotokoll so umbenannt, daß das Wort Internet im Namen vorkam. Im Gegensatz zu SUN hat sich Microsoft bis auf ein kleines Detail[8] nicht die Mühe gemacht, das Protokoll wirklich in Richtung Internet zu optimieren.

Die erste Anfrage, die der Client an den Server schickt, ist ein *Negotiate Protocol Request*. In dieser Anfrage schickt der Client an den Server eine Liste der Protokollvarianten, die er beherrscht. Der Server wählt nun aus dieser Liste der Protokolle eins aus, und schickt eine entsprechende Antwort zurück. Die verschiedenen Protokolle bauen aufeinander auf. Daher kann man mit dem Parameter `protocol` das

[7] Im Prinzip deshalb, da mit Oplocks auch der Server von sich aus aktiv werden kann.

[8] Dies Detail hat nichts mit SMB, sondern mit NetBIOS zu tun. SMB-Server wollen im NetBIOS-Sitzungsaufbau mit ihrem eigenen NetBIOS-Namen angesprochen werden. Ein CIFS-Server im Internet ist aber nur unter seinem DNS-Namen oder seiner IP-Adresse bekannt. Der NetBIOS-Name ist normalerweise nicht publiziert. Daher lauschen alle CIFS-Server auf den eigentlich illegalen NetBIOS-Namen `*SMBSERVER`. Das ist der ganze Unterschied zwischen SMB und CIFS.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

höchste Protokoll festlegen, mit dem Samba arbeiten soll.

In der Antwort auf diese erste Anfrage werden zwei weitere Einstellungen verschickt, die Teile des weiteren Ablaufs festlegen.

Der Server entscheidet, ob er die Zugriffssteuerung auf Benutzer- oder auf Freigabeebene regeln möchte. Damit wird festgelegt, zu welchem Zeitpunkt der Benutzer ein Paßwort liefern muß. Entweder kann es beim direkt folgenden *Session Setup* erfolgen, oder erst beim *Tree Connect* danach.

Der Parameter `security` legt fest, welche Art der Zugriffssteuerung gewählt wurde. Mit `security = share` wird die Freigabeebene eingestellt, `security = user` legt die Clients auf die Benutzerebene fest.

Sichtbar wird diese Unterscheidung in der Windowswelt nur bei Windows 95 und Windows 98. Diese Betriebssysteme beherrschen zunächst einmal nur die Zugriffssteuerung auf Freigabeebene, da sie nicht über eine Benutzerdatenbank verfügen. Es ist nicht möglich, einzelnen Benutzern den Zugriff auf Freigaben zu gewähren oder zu verweigern. Um trotzdem benutzerbasiert Zugriffssteuerung zu ermöglichen, muß ein Server angegeben werden, der für Windows die Benutzerdatenbank pflegt. Damit können Paßwörter benutzerbasiert überprüft werden.

Weiterhin gibt der Server dem Client vor, ob Klartextpaßwörter verwendet werden sollen, oder ob die Paßwörter verschlüsselt werden. Wenn der Server festlegt, daß verschlüsselte Paßwörter verwendet werden, wird zusätzlich die Herausforderung für das *Challenge Response* Verfahren mitgeschickt.

Die Entscheidung über Klartextpaßwörter muß also getroffen werden, ohne daß der Server den Benutzernamen, der sich anmelden will, kennt. Es ist also nicht möglich, für einige Benutzer Klartextpaßwörter und für andere Benutzer verschlüsselte Paßwörter zu verwenden.

14.5 Session Setup

Nachdem die Protokollversion ausgehandelt ist, wird vom Client ein *Session Setup* verschickt. In diesem Session Setup schickt der Client seinen Benutzernamen an den Server. Sofern dieser `security = user` verlangt hat, wird an dieser Stelle das Paßwort mitgeschickt. Damit ist der Server in der Lage, die Identität des Benutzers festzustellen. Wenn `security = share` vereinbart wurde, dann ignoriert der Server ein hier eventuell mitgeschicktes Paßwort.

14.6 Tree Connect

Als letztes legt der Client fest, welche Freigabe er ansprechen will. Der entsprechende Aufruf heißt *Tree Connect*. Sofern `security = share` vereinbart wurde, wird an dieser Stelle das Paßwort überprüft. Der Benutzername kann in diesem Fall nicht zur Zugriffsregelung verwendet werden. Dieser wurde unter Umständen gar nicht übermittelt, da der Client den Session Setup komplett auslassen darf. Andererseits hat er bei einem durchgeführten Session Setup kein Paßwort angeben müssen, anhand dessen die Identität des Benutzers zweifelsfrei hätte festgestellt werden können.

15 Rechte an Freigaben

Bei Windows NT kann man mit zwei unterschiedlichen Mechanismen Rechte vergeben. An einer Freigabe kann man über Schreib- und Lesezugriff entscheiden. Innerhalb des Dateisystems kann man detailiert Rechte vergeben.

Ist bei Samba `security = user` gesetzt, so hat der Server die Möglichkeit, anhand des angemeldeten Benutzers Zugriffsrechte zu vergeben oder zu verweigern. Wenn bei der Einstellung einer

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Freigabe keine Parameter für die Zugriffsrechte gesetzt sind, hat jeder korrekt angemeldete Benutzer Leserecht. Man kann auch Gastbenutzern Leserecht geben, indem man `guest ok = yes` setzt.

Mit den Optionen zur Rechtevergabe an Freigaben hat man die Möglichkeit, einzelnen Benutzern und ganzen Unixgruppen Rechte zu geben oder zu nehmen. Die Möglichkeiten sind hier deutlich weitergehend als die Semantik, die Unix mit den Rechtemasken für den Dateibesitzer, die besitzende Gruppe und den Rest der Welt bereit stellt. Von den möglichen Anwendungen sollen hier drei häufig benötigte Fälle dargestellt werden:

- ***Alle* Benutzer haben gleichen Zugriff**

  ```
  [projekt]
  path = /data/projekt
  ```

  Bei dieser Freigabe bekommen alle Benutzer, die sich mit Namen und Paßwort am Server angemeldet haben, *Leserecht* auf die Freigabe. Schreibrecht vergibt man, indem man den Parameter `writeable = yes` setzt:

  ```
  [projekt]
     path = /data/projekt
     writeable = yes
  ```

- ***Einige* Benutzer haben gleichen Zugriff**

  Will man den Zugriff auf einige Benutzer einschränken, erstellt man eine Liste `valid users` auf:

  ```
  [projekt]
  path = /data/projekt
  valid users = mueller, meier
  ```

  Zu dieser Freigabe haben die Benutzer mueller und meier Lesezugriff. Sollen diese Benutzer Schreibzugriff bekommen, so ist wie im vorangegangenen Beispiel der Parameter `writeable = yes` zu setzen:

  ```
  [projekt]
  path = /data/projekt
  valid users = mueller, meier
  writeable = yes
  ```

  Für den Parameter `valid users` spielt der Benutzer root keine besondere Rolle. Das heißt, daß er auf die Freigabe `projekt` keinen Zugriff hat. Soll er Zugriff bekommen, muß man ihn wie jeden anderen Benutzer in die Liste `valid users` mit aufnehmen.

  Der Parameter `valid users` gibt die Möglichkeit, ganze Unixgruppen in den Zugriff mit aufzunehmen. Um dies zu erreichen, muß man das at-Zeichen voranstellen:

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
[projekt]
path = /data/projekt
valid users = root, @users
writeable = yes
```

Mit dieser Einstellung haben alle Benutzer, die in der Unixgruppe users sind, Schreibzugriff auf die Freigabe. Zusätzlich kann der Benutzer root schreiben.

- **Einige Benutzer haben Leserecht, andere Schreibrecht**

  Will man differenziert Rechte vergeben, so muß man sämtliche Benutzer, die überhaupt Zugriff auf die Freigabe bekommen sollen, in die Liste `valid users` aufnehmen, und mit `writeable = no` nur Leserechte vergeben. Die Benutzer, die über diese Standardeinstellung hinaus Schreibrecht bekommen sollen, müssen in die `write list` aufgenommen werden.

```
[projekt]
path = /data/projekt
valid users = @users, @admins
write list = @admins
```

Mit diesen Einstellungen haben die Benutzer der Gruppe users Leserecht, und die Benutzer der Gruppe admins haben Schreibrecht.

16 Zugriffsrechte im Dateisystem

Unter Windows NT gibt es zwei Möglichkeiten, Netzzugriff auf Dateien zu kontrollieren. Über eine Freigabe kann ein Lese- oder ein Schreibrecht vergeben werden. Ist das freigegebene Dateisystem mit NTFS formatiert, können durch Access Control Lists im Dateisystem Rechte vergeben werden. Damit muß ein Benutzer sowohl durch die Freigabe- als auch durch die Dateisystemrechte zu einer Operation berechtigt sein.

Auch bei Samba auf Unix gibt es zwei Stellen, an denen Zugriff auf Dateien und Verzeichnisse geregelt ist. Die im Kapitel 15 beschriebenen Zugriffsrechte beziehen sich ausschließlich auf die von Samba selbst vergebenen Rechte. Diese von Samba vergebenen Rechte können die darunter liegenden Unixrechte nicht erweitern. Das heißt, Samba kann für bestimmte Freigaben Schreibrecht vergeben. Der Benutzer, der zugreift, kann aber nur dann wirklich in den freigegebenen Dateien und Verzeichnissen schreiben, wenn er dies unter Unix ebenfalls darf. Diese Einschränkung durch Unixrechte ist ein wichtiges Prinzip von Samba: Im Dateisystem implementiert Samba keine eigenen Zugriffskontrollen, sondern verläßt sich auf die Unixmechanismen.

Samba könnte theoretisch eine eigene Datenbank von Zugriffsrechten führen. In dieser Datenbank könnte die vollständige NT-Semantik von Access Control Lists abgelegt und implementiert werden. Zwei Gründe sprechen gegen diesen Ansatz:

- Wenn Samba tatsächlich Dateisystemrechte implementieren würde, wären die Entwickler dafür verantwortlich, daß diese korrekt eingehalten werden. Zugriffsrechte auf Dateien werden vom

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Betriebssystem bereits hervorragend implementiert, warum sollte man sich also die zusätzliche Komplexität einhandeln?[9]

- Sobald Samba eine eigene ACL-Datenbank implementiert, gilt diese ausschließlich für den Dateizugriff via SMB. Es ist nicht möglich, Samba-ACL synchron mit dem Unix-Dateisystem zu halten, wenn auch noch Zugriff von Unixprozessen aus erlaubt wird. Wenn sich Verzeichnisbäume ändern, ohne daß Samba involviert ist, wie soll Samba dann die ACLs korrekt anpassen?

Eng verwoben mit den Unix-Zugriffsrechten auf Dateien ist in der Implementation von Samba die Behandlung der DOS-Attribute. Diese Attribute sind Eigenschaften von Dateien, die es in dieser Form unter Unix nicht gibt. Viele Applikationen, die auf ein Netzwerklaufwerk zugreifen, setzen jedoch funktionierende Attribute voraus. Insbesondere das Archiv-Attribut wird von vielen Programmen verwendet. Insgesamt kennt DOS vier verschiedene Attribute, die für Dateien vergeben werden können:

Read-Only Der Inhalt dieser Datei kann nur gelesen, aber nicht geschrieben werden. Die Datei kann nicht gelöscht werden.

System Diese Datei ist für spezielle Betriebssystemzwecke vorgesehen.

Hidden Diese Datei wird mit dem Kommando 'DIR' nicht angezeigt.

Archiv Das Archivbit wird bei jedem Schreibzugriff gesetzt. Backupprogrammen ist es freigestellt, dieses Bit zurückzusetzen. Damit kann eine inkrementelle Sicherung ermöglicht werden.

Diese Bits können unter DOS von jedem Benutzer frei gesetzt und wieder zurückgesetzt werden. Das Schreibschutzbit ist also nicht als echter Zugriffschutz zu verstehen, sondern nur als kleine Hilfestellung gegen Fehlbedienungen.

16.1 Abbildung DOS-Attribute zu Unix-Rechten

Unix führt mit jeder Datei einen Satz von Zugriffsrechten mit. Diese sind aufgeteilt in drei Gruppen von Benutzern: Der Dateibesitzer, die besitzende Gruppe und alle anderen. Jeder Gruppe können drei Rechte zugeteilt werden: Lesen, Schreiben und Ausführen.

Unter DOS werden Ausführungsrechte nicht verwendet. Sie stehen für Samba zur Verfügung, um die DOS-Attribute im Unix-Dateisystem abzubilden. Das Schreibschutzbit unter DOS hat mit dem Schreibrecht des Dateibesitzers unter Unix eine Entsprechung. Bis auf die Umsetzung des Schreibschutzbits kann die Umsetzung der Attribute unter Samba mit den entsprechenden Parametern `map <xxx>` gesteuert werden, wobei das Archivbit ohne Zusatzangabe umgesetzt wird, die anderen beiden Attribute nicht. Die Attributumsetzung erfolgt anhand der folgenden Tabelle:

DOS-Attribut	Unix-Recht	Maske	Parameter	Standard
Schreibschutz	Schreibrecht Besitzer	200	-	immer
Archiv	Ausführung Besitzer	100	`map archive`	`yes`
System	Ausführung Gruppe	010	`map system`	`no`
Versteckt	Ausführung Andere	001	`map hidden`	`no`

[9] Unter Marketinggesichtspunkten kann es wichtig sein, vollständige NT-Kompatibilität zu implementieren, die Samba mit dem Unix-Rechtemodell bisher nicht bietet. Es existieren Patches, die eine eigene ACL-Datenbank implementieren. Diese sind jedoch leider momentan noch nicht frei verfügbar. Dies ist mit der GPL durchaus möglich, da sie niemandem zugänglich gemacht wurden. Es wird jedoch in der Zukunft eine veröffentlichte Version geben.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Samba muß nun diese beiden Dateiattribute ineinander überführen. Samba muß neu erstellten Dateien Unixrechte zuordnen. Wird eine Datei neu erstellt, dann gibt der Client dem Server die DOS-Attribute mit, mit der er die Datei erstellt haben möchte. Daraus formt Samba einen Satz von Unix-Zugriffsrechten. Diese Rechte werden vom Parameter `create mask` eingeschränkt. Die Standardvorgabe für die `create mask` ist gleich `744`, was der Rechtemaske `rwxr--r--` entspricht. Der Dateieigentümer hat Schreib- und Leserecht, alle anderen haben reines Leserecht. Samba schränkt die Rechte ein, indem der gewünschte Satz an Rechten mit einer logischen UND-Operation mit der `create mask` verknüpft wird. Nur die Rechte, die in der `create mask` gesetzt sind, können möglicherweise in der neu erzeugten Datei auftauchen. In einem weiteren Schritt setzt Samba explizit gewünschte Zugriffsrechte anhand des Parameters `force create mode`, dessen Standardwert auf `000` steht. Dies geschieht durch eine ODER-Verknüpfung mit diesem Wert.

Diese Zusammenhänge werden an einem Beispiel deutlicher. Es kann gewünscht sein, daß auf neu erstellten Dateien nur der Dateibesitzer und die Gruppe Leserecht haben sollen. Der Rest der Welt soll diese Dateien nicht lesen können. Das wird dadurch erreicht, daß man die `create mask = 740` setzt, also das Leserecht für den Rest der Welt ausmaskiert. Es kann darüber hinaus gewünscht sein, daß die besitzende Gruppe ein Schreibrecht eingeräumt bekommt. Das kann man durch `force create mode = 020` erreichen. Tabellarisch dargestellt heißt dies:

Wunsch			`rw-r--r--`
create mask	740	UND	`rw-r-----`
			`rw-r-----`
force create mode	020	ODER	`----w----`
Ergebnis			`rw-rw----`

Die Ausführungsrechte auf Dateien werden unter DOS nicht verwendet, sie können also verwendet werden, um DOS-Attribute im Unix-Dateisystem abzulegen. Ausführungsrechte auf Dateiverzeichnissen wirken sich jedoch auf das Verhalten von Samba aus, da durch sie der Zugriff zu den Verzeichnissen geregelt wird. Daher kann es wünschenswert sein, daß die Rechtezuweisung auf Dateien und Verzeichnissen unterschiedlich geregelt wird. Die Parameter `create mask` und `force create mode` wirken daher nur auf neu angelegte Dateien. Für Verzeichnisse sind die Parameter `directory mask` und `force directory mode` verantwortlich. Der Vorgabewert für `directory mask` ist hierbei `755`, um den Zutritt für die Gruppe und den Rest der Welt zu ermöglichen, die Vorgabe für `force directory mode` besetzt mit dem Wert `000` kein zusätzliches Recht.

16.2 Beispiel: Ein Projektverzeichnis

Häufig muß man einer Anzahl von Benutzern gemeinsamen Schreibzugriff auf eine Freigabe, beispielsweise auf die Freigabe `fibu`, geben. Das Beispiel der Projektverzeichnisse wird noch mehrfach betrachtet werden. Es gibt mit Samba viele Möglichkeiten der Realisation, die alle für unterschiedliche Situation geeignet sind.

Ein einfaches Projektverzeichnis läßt sich folgendermaßen realisieren:

```
[fibu]
   path        = /data/fibu
   writeable   = yes
   valid users = @fibu, mueller, meier
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Damit darf die Gruppe **fibu** das Recht, auf diese Freigabe schreibend zuzugreifen. **mueller** und **meier**, die nicht Mitglied der Finanzbuchhaltung sind, dürfen ebenfalls schreiben. Damit problemloser gemeinsamer Zugriff möglich ist, muß die Rechtevergabe im Unix-Dateisystem geregelt werden. Dabei wird hier vorausgesetzt, daß im Unix selbst nur die Benutzer der Gruppe **fibu** auf `/data/fibu` zugreifen sollen. **meier** und **mueller** sind *nicht* Mitglieder der Gruppe **fibu**, sollen aber trotzdem schreiben können. Für sie muß eine Sonderregelung geschaffen werden, die sich mit Standard-Unixrechten nicht abbilden läßt. Dazu benötigt man die ACLs aus Kapitel 18.

Hat man keine ACLs zur Verfügung, gibt es eine sehr einfache Möglichkeit, jegliche Probleme im gemeinsamen Dateizugriff zu vermeiden, ist der Parameter `force user`. Will man diesen Parameter anwenden, so sollte man für diese Freigabe oder für alle solchen Gruppenfreigaben einen separaten User anlegen, und diesem dann das freigegebene Verzeichnis übergeben:

```
root@delphin:~ > mkdir -p /data/fibu
root@delphin:~ > useradd fibuuser
root@delphin:~ > chown projektuser /data/fibu/
root@delphin:~ > chmod 770 /data/fibu
```

Die Freigabe sieht dann folgendermaßen aus:

```
[fibu]
   path        = /data/fibu
   writeable   = yes
   valid users = @fibu, mueller, meier
   force user  = fibuuser
```

Die Zugriffskontrolle wird bei dieser Definition ganz normal anhand von `valid users` vorgenommen. Nur die dort erwähnten Benutzer bekommen Zugriff auf die Freigabe. *Nachdem* der Zugriff gewährt wurde, vergißt Samba den Namen, mit dem sich der Benutzer angemeldet hat. Samba schaltet für jegliche Zugriffe im Dateisystem in den Benutzer **fibuuser**. Man muß sich damit nicht mehr um gemeinsame Zugriffsrechte im Unix kümmern, da man ohnehin nur unter einer einzigen Userid arbeitet. Man verliert jedoch die Nachvollziehbarkeit. Alle Dateien gehören **pcuser**. Dies wird insbesondere auch so im entsprechenden Dialog von Windows angezeigt.

Mit etwas mehr Aufwand kann man es schaffen, den Dateibesitzer korrekt zu behalten und gleichzeitig gemeinsames Schreiben zu ermöglichen. Das Verzeichnis `/data/fibu` selbst kann mit den korrekten Gruppenschreibrechten angelegt werden:

```
root@delphin:~ > mkdir -p /data/fibu
root@delphin:~ > groupadd fibu
root@delphin:~ > chgrp fibu /data/fibu/
root@delphin:~ > chmod 770 /data/fibu
```

Die Benutzer der Gruppe **fibu** können in diesem Verzeichnis einwandfrei Dateien anlegen und ihre eigenen Dateien auch ändern. Es gibt jedoch noch zwei Probleme.

- **mueller** und **meier** können nicht auf das Verzeichnis zugreifen, da Unix ihnen den Zugriff verweigert.
- Die Benutzer aus der Gruppe **fibu** müssen nicht notwendigerweise diese Gruppe als Hauptgruppe haben. Das heißt, neu angelegte Dateien gehören möglicherweise anderen Gruppen an. Dieses spezielle Problem ließe sich mit dem set-group-id Bit auf dem Verzeichnis `/data/fibu` lösen:

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
chmod g+s /data/fibu
```

mueller und **meier** blieben jedoch immer noch außen vor, da sie nicht in der Gruppe **fibu** sind, also auf dem Verzeichnis `/data/fibu` kein Schreibrecht haben.

Beide Probleme bekommt man mit dem Parameter `force group = fibu` in den Griff. Dieser Parameter arbeitet genau so wie `force user`, nur daß er sich anstatt der User-ID auf die Group-ID bezieht. Jegliche Dateisystemzugriffe werden damit als Gruppe **fibu** vorgenommen, die User-ID bleibt unangetastet.

Als letztes muß nun noch sichergestellt werden, daß die Gruppe, in diesem Fall **fibu**, immer schreiben kann, und daß der Rest der Welt keinen Zugriff bekommt. Die vollständige Freigabedefinition sieht demnach folgendermaßen aus:

```
[fibu]
   path                = /data/fibu
   writeable           = yes
   valid users         = @fibu, mueller, meier
   force group         = fibu
   create mask         = 740
   directory mask      = 750
   force create mode   = 020
   force directory mode = 020
```

17 Projektverzeichnisse, zum zweiten

Folgendes Problem stellt sich bei der Migration von Novell zu Samba recht häufig. Unter Novell kann man anhand von Gruppenzugehörigkeiten den Zugriff auf Verzeichnisse regeln. Dies ist unter Samba anhand von Unixrechten ebenfalls möglich. Was Unix leider nicht zur Verfügung stellt, ist die Möglichkeit, Verzeichnisse vor Benutzern zu verstecken. Ein Benutzer sieht grundsätzlich alle Verzeichnisse, bekommt aber bei vielen dieser Verzeichnisse die Meldung, daß der Zugriff verweigert wurde. Wenn es jetzt anhand der Gruppenzugehörigkeit des Benutzers möglich wäre, nur die Verzeichnisse anzuzeigen, auf die er tatsächlich Zugriff hat, könnten die Verzeichnisse deutlich übersichtlicher werden.

Die Flexibilität von Samba ermöglicht es, diese von Unix vorgegeben Beschränkung zu umgehen, und zwar unter Benutzung von Skripten, die vor dem Verbinden mit einer Freigabe ausgeführt werden.

Folgendes Szenario wird vorausgesetzt: Jeder Benutzer ist in mehrere Gruppen eingeteilt, die jeweils Projekte, Arbeitsgruppen oder Abteilungen darstellen können. Jede dieser Gruppen hat unter `/data/groups` ein eigenes Verzeichnis, auf das sie schreiben darf. Die einzelnen Verzeichnisse haben das Set Group ID Bit gesetzt, damit die neu angelegten Dateien den jeweiligen Gruppen angehören.

Als Beispiel gebe es die drei Gruppen `edv`, `fibu` und `verkauf`. Unter `/data/groups` kann man folgende Gruppenverzeichnisse anlegen:

```
root@server:/data/groups> ls -l
total 12
drwxrws---   2 root     edv          4096 Jan 31 06:43 edv
drwxrws---   2 root     fibu         4096 Jan 31 06:43 fibu
drwxrws---   2 root     verkauf      4096 Jan 31 06:43 verkauf
root@server:/data/groups>
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Die korrekten Rechte erreicht man unter Unix durch:

```
root@server:/root> mkdir /data/groups/edv
root@server:/root> chgrp edv /data/groups/edv
root@server:/root> chmod 2770 /data/groups/edv
```

Eine Freigabe, die jedem Benutzer anhand seiner Rechte hierauf Zugriff gewährt, kann folgendermaßen aussehen:

```
[allgroups]
path = /data/groups
writeable = yes
create mode = 740
directory mode = 750
force create mode = 020
force directory mode = 020
```

Zu beachten ist hier, daß keine zusätzlichen Einschränkungen anhand von `valid users` notwendig sind, da der Zugriff durch die Unixrechte beschränkt ist. Die Parameter `create mask` und `directory mask` sind nicht strikt notwendig, da bereits auf der Ebene `/data/share` die Benutzer abgewiesen werden. Die Parameter `force create mode` und `force directory mode` sind hingegen notwendig, da ohne sie neu angelegte Dateien nicht die notwendigen Gruppenschreibrechte erhalten würden, die zum gemeinsamen Zugriff notwendig sind.

Diese Freigabe erfüllt funktional genau die Anforderungen, daß jeder in die Verzeichnisse schreiben darf, für die er die Gruppenmitgliedschaft hat. Der Nachteil an diesem Verfahren ist, daß er alle anderen Verzeichnisse sieht, was bei großen Servern mit vielen Gruppen recht unübersichtlich werden kann.

Die preexec-Skripte von Samba ermöglichen die übersichtliche Darstellung der Gruppenstruktur. Ein preexec-Skript wird ausgeführt, bevor der Benutzer tatsächlich mit der Freigabe verbunden wird.

```
[gruppen]
path = /data/users/%U
root preexec = /usr/local/bin/mklinks %U
writeable = yes
```

Die Datei `mklinks` hat folgenden Inhalt:

```
#!/bin/sh
umask 022
cd /data/users
rm -rf "$1"
mkdir "$1"
cd "$1"
for i in $(groups $1)
do
        ln -s "/data/groups/$i" .
done
```

Beim Verbinden an die Freigabe wird das Verzeichnis `/data/users/username` frisch erstellt, das anhand der Gruppenzugehörigkeit des Benutzers eine Liste von symbolischen Links erstellt, die auf die eigentlichen Gruppenverzeichnisse verweisen. Damit bekommt er nur die Verzeichnisse im Explorer angezeigt, auf die er tatsächlich Zugriff hat. Durch die Angabe `path = /data/users/%U` ist zudem

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

sichergestellt, daß die Freigabe für alle Benutzer gleich heißt, aber für jeden Benutzer auf ein eigenes Verzeichnis verweist.

Das Skript wird in diesem Beispiel als `root preexec` ausgeführt, um den Verwaltungsaufwand beim Anlegen neuer Benutzer zu minimieren. Mit einem reinen `preexec` ohne Rootrechte wäre es notwendig, für jeden Benutzer unterhalb von `/data/users` ein eigenes Verzeichnis mit den notwendigen Rechten anzulegen. Jedoch darf dieses Verfahren nur dann angewendet werden, wenn die Benutzernamen unter vertrauenswürdiger Kontrolle stehen. Wenn es möglich ist, daß Benutzernamen beispielsweise von einem NIS-Server bezogen werden, kann über einen Benutzernamen **../..** das gesamte Dateisystem gelöscht werden. Ist ein NIS-Server beteiligt, muß man das Verfahren ohne `root preexec` und nur mit `preexec` ohne Root-Rechte verwenden.

Alternativ könnte man das Verzeichnis mit der Gruppenliste im Heimatverzeichnis des Benutzers anlegen, wobei dabei Zweifel bezüglich der Übersichtlichkeit angebracht sind. Ein weiteres Argument, das Skript unter Rootrechten auszuführen, ist die Betriebssicherheit. Ohne dies wäre es dem Benutzer möglich, sich vollständig von einem Gruppenverzeichnis auszuschließen indem er das gesamte Verzeichnis inklusive symbolischem Link löscht. Mit der dargestellten Version gehört das Verzeichnis mit den symbolischen Links dem Benutzer root, und Fehlbedienungen in dieser Ebene sind ausgechlossen.

Wenn man die Freigabe `[allgroups]` auf `browseable = no` setzt, so hat man maximale Übersichtlichkeit bei vollem Zugriff auf sämtliche Gruppenverzeichnisse durch den Administrator gegeben.

Ändern sich die Gruppenzugehörigkeiten eines Benutzers, so kann er einfach durch ein Neuverbinden an die Freigabe die neue Sicht auf die Verzeichnisstruktur bekommen. Dieses Neuverbinden kann erzwungen werden, indem der richtige Serverprozess getötet wird. Dieser kann anhand des Programms `smbstatus` leicht herausgefunden werden.

18 ACLs

Die Zugriffsrechte unter Unix werden durch den Dateimodus bestimmt. Dieser Dateimodus enthält neun Bits, die den Zugriff auf die Datei regeln. Dazu kommen drei zusätzliche Bits für spezielle Anwendungen. Mit diesen neun Bits können Zugriffsrechte für drei Benutzerklassen vergeben werden: Den Dateibesitzer, die besitzende Gruppe und den Rest der Welt. Mit dem Befehl `chmod` werden diese Rechte gesetzt.

Dieser Mechanismus hat einen unschätzbaren Vorteil: Er ist einfach. Mit insgesamt zwölf Bits kann ein sehr großes Spektrum an Szenarien abgedeckt werden. Jedoch ist es oft notwendig, Zugriffsrechte feiner zu vergeben, als dies mit Unix möglich ist. Insbesondere haben viele Unternehmensanwendungen komplexere Anforderungen an die Zugriffsrechte.

Beispielsweise soll auf einem Verzeichnis die Gruppe **fibu** Schreibrecht haben und die Gruppe **controlling** soll Leserecht bekommen. Der Benutzer **mueller** ist nun in der Gruppe **controlling** und hat sich bei der Finanzbuchhaltung unbeliebt gemacht. Er soll auf dieses Verzeichnis keinen Zugriff mehr haben. Eine solche Konfiguration ist mit den traditionellen Unix-Zugriffsrechten nicht mehr abzubilden.

Die Hersteller von Unix haben sich irgendwann zusammengefunden, um das beschränkte Rechtemodell zu erweitern. Geplant war eine Erweiterung, die sich in das vorhandene Rechtemodell von Unix nahtlos einbinden läßt, aber die dem Benutzer erheblich mehr Möglichkeiten zur Rechtesteuerung gibt.

Zugriffskontrollisten (Access Control Lists oder ACLs) unterstützen genau diese weitergehenden Zugriffsrechte. Beliebige Benutzer und Gruppen können Rechte auf Dateien und Verzeichnissen bekommen oder verweigert bekommen. Die klassischen drei Benutzerklassen kann man als drei Einträge in einer ACL ansehen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Das Modell der ACLs erweitert Möglichkeiten, wem man Rechte geben kann. Es erweitert nicht die Art der Rechte, die vergeben werden können. Es geht weiterhin nur um die Rechte Lesen, Schreiben und Ausführen, mit der bekannten Bedeutung auf Dateien und Verzeichnissen.

18.1 Rechte unter Unix

Die Auswertung der Zugriffsrechte unter Unix funktioniert, indem zuerst entschieden wird, welche der drei Rechtegruppen Benutzer, Gruppe und Andere benutzt werden soll. Im zweiten Schritt wird nachgesehen, ob das gewünschte Recht auf der Datei gesetzt ist.

Die Zugriffsrechte eines Benutzers werden bestimmt durch seinen Sicherheitskontext. Dieser Sicherheitskontext besteht aus seiner effektiven User ID (EUID), seiner primären Gruppe (EGID) und seinen zusätzlichen Gruppen (GIDs). Die Entscheidung für eine Rechtegruppe funktioniert in drei Schritten:

- Ist EUID gleich dem Dateibesitzer? In diesem Fall wird die erste Rechtegruppe, die für den User benutzt.
- Ist der Benutzer in der besitzenden Gruppe? Dann wird die zweite Rechtegruppe für Group benutzt. Die tatsächliche Prüfung passiert, indem die besitzende Gruppe in der Liste GID's gesucht wird, in der der Benutzer aufgenommen ist.
- Ist beides nicht der Fall, so wird die dritte Rechtemaske für Others benutzt.

Wenn entschieden wurde, welche Rechtemaske verwendet werden soll, wird nicht mehr versucht, eine andere Rechtemaske zu verwenden. Wenn ein Benutzer sich selbst das Leserecht auf einer Datei genommen hat, und dem Rest der Welt über die Maske Others Leserecht gegeben hat, wird er die Datei nicht lesen können.

18.2 Einträge in einer ACL

Da ACLs eine reine Erweiterung des Unix-Rechtemodells sind, gibt es weiterhin einen Dateibesitzer und eine besitzende Gruppe für jede Datei. Eine Access Control List kennt eine Anzahl unterschiedlicher Einträge:

ACL_USER_OBJ: Dies ist die Rechtemaske für den Dateibesitzer. Sie entspricht der ersten Rechtemaske für den User im klassischen Rechtemodell.

ACL_GROUP_OBJ: Dies ist die Entsprechung der Group-Rechtemaske im klassischen Modell.

ACL_OTHER: Die Rechtemaske für den Rest der Welt unter Unix. Von diesen ersten drei Einträgen gibt es jeweils genau einen in jeder ACL.

ACL_USER: Ein Eintrag für einen benannten Benutzer. Von diesem Eintrag kann es mehrere geben, mit denen für unterschiedliche Benutzer unterschiedliche Rechte vergeben werden. Gibt es einen Benutzereintrag ohne jegliche Rechte, kann dieser auf die Datei nicht zugreifen.

ACL_GROUP: Eintrag für eine Gruppe. Auch von diesem Eintrag kann es mehrere geben.

ACL_MASK: Die Maske für die effektiven Rechte. Sobald für eine Datei ACLs verwendet werden, wird `ls -l` diese Rechtemaske als Gruppenrecht anzeigen. Sobald mit `chmod` die Rechte für die besitzende Gruppe verändert werden (etwa per `chmod g-rx`), wird die ACL_MASK verändert.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

18.3 Rechte mit ACLs

Wenn ein Prozess auf eine Datei zugreifen will, wird mit dem folgenden Algorithmus festgestellt, ob der Zugriff zugelassen oder verweigert wird.

- Ist die EUID des Prozesses gleich dem Dateibesitzer, wird der Zugriff gewährt, wenn der Eintrag für ACL_USER_OBJ die benötigten Zugriffsrechte enthält.
- Wenn es in der ACL einen ACL_USER Eintrag gibt, der der EUID entspricht, wird dieser Eintrag verwendet. Ist dass gewünschte Recht in diesem Eintrag vorhanden, wird der Zugriff gewährt, sofern es *auch* im Eintrag ACL_MASK vorhanden ist. Ist das Recht entweder im ACL-Eintrag oder in ACL_MASK *nicht* vorhanden, wird das Recht verweigert.
- Ist der Benutzer in der besitzenden Gruppe der Datei ist (Eintrag für ACL_GROUP_OBJ), oder wenn der Benutzer in einer Gruppeneinträge vom Typ ACL_GROUP ist, wird folgendes getestet: Sind die gewünschten Rechte in einem der Einträge vollständig vorhanden, so wird der Zugriff gewährt, sofern das Recht ebenfalls im Eintrag ACL_MASK vorhanden ist. Ansonsten wird der Zugriff verweigert.
- Wenn kein Eintrag gefunden werden konnte, wird der Zugriff anhand des Eintrags ACL_OTHER gewährt.

Es ist hier wichtig festzustellen, daß die Benutzereinträge in einer ACL immer *vor* Gruppeneinträgen durchsucht werden. Damit werden die spezifischeren Einträge grundsätzlich vorrangig vor den weniger spezifischen Einträgen behandelt.

18.4 ACL-Beispiel

Linux unterstützt mit dem richtigen Kernelpatch die beschriebenen ACLs auf dem Ext2-Dateisystem. Der Kernelpatch ist zu diesem Zeitpunkt (Sommer 2001) notwendig, da Linus Torvalds ihn noch nicht in den Standardkernel aufgenommen hat. Unter http://acl.bestbits.at findet man den entsprechenden Kernelpatch und die notwendigen Utilities. ACLs setzen kann man mit `setfacl`, mit `getfacl` werden ACLs angezeigt.

Das oben beschriebene Beispiel eines Verzeichnisses für die Finanzbuchhaltung läßt sich folgendermaßen erstellen:

```
root@delphin:~ > cd /
root@delphin:/ > mkdir fibu
root@delphin:/ > chmod o-rwx fibu
root@delphin:/ > setfacl -m group:fibu:rwx fibu
root@delphin:/ > setfacl -m group:controlling:rx fibu
root@delphin:/ > setfacl -m user:mueller:--- fibu
root@delphin:/ > getfacl fibu
# file: fibu
# owner: root
# group: root
user::rwx
user:mueller:---
group::r-x
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
group:fibu:rwx
group:controlling:r-x
mask:rwx
other:---
```

Obwohl der Benutzer **mueller** Mitglied der Gruppe **controlling** ist, hat er keinen Zugriff auf dieses Verzeichnis, da der ACL-Eintrag für ihn keinen Zugriff erlaubt, und dieser vor seinem Gruppeneintrag gefunden wird.

Interessant an diesem Beispiel ist die Behandlung der ACL-mask. Sie wird in der Anzeige von `ls -l` als Gruppenberechtigung angezeigt.

```
root@delphin:/ > ls -ld fibu
drwxrwx---   2 root     root         4096 Aug 28 07:56 fibu
```

An der Ausgabe von `getfacl` ist zu sehen, daß die besitzende Gruppe **root** nur Lese- und Ausführungsrechte hat. Trotzdem zeigt `ls -l` für die Gruppe ein `rwx` an. Dies wird gemacht, um Benutzer vor Überraschungen zu schützen. Über ACLs sind auf dem Verzeichnis `fibu` mehr Rechte vergeben, als aus der Ausgabe von `ls -l` ersichtlich ist. Will man die Rechtevergabe durch ACLs ausschalten, genügt es, mit `chmod g-rwx fibu` die Rechte für die besitzende Gruppe komplett wegzunehmen.

18.5 Default ACLs

Das vorangegangene Beispiel ist noch nicht vollständig. Für das Verzeichnis `fibu` sind die Rechte korrekt gesetzt. Wenn jedoch Benutzer in diesem Verzeichnis Dateien anlegen, gelten wieder nur die normalen Regeln von Unix. Erreichen möchte man jedoch in der Regel, daß für neue Dateien unterhalb eines solchen Verzeichnisses wieder die gleichen Regeln gelten wie für das Verzeichnis selbst.

Wenn eine neue Datei oder ein Verzeichnis erstellt wird, muß über die Zugriffsrechte entschieden werden, die darauf gesetzt werden. Im traditionellen Unixmodell wird die Rechtemaske auf neuen Dateien und Verzeichnissen durch die so genannte *umask* eingeschränkt. Ein Programm, das eine Datei erzeugt, kann einen Satz von Zugriffsrechten bestimmen, mit dem die Datei erzeugt werden soll. Bevor die Datei tatsächlich mit diesen Rechten erzeugt wird, wird dieser Satz von Rechten um die Bits eingeschränkt, die in der *umask* gesetzt sind. Wenn ACLs verwendet werden, ist dieses einfache Modell nicht mehr verwendbar. Stattdessen kann man für jedes Verzeichnis eine so genannte *Default ACL* vergeben. Diese werden an Dateien und Verzeichnisse weitergegeben.

Setzen kann man die Default ACL, indem man dem Befehl `setfacl` den Schalter `-d` mitgibt:

```
root@delphin:/ > setfacl -d -m group:fibu:rwx fibu/
root@delphin:/ > setfacl -d -m group:controlling:r-x fibu
root@delphin:/ > setfacl -d -m user:mueller:--- fibu
root@delphin:/ > getfacl fibu
# file: fibu
# owner: root
# group: root
user::rwx
user:mueller:---
group::r-x
group:fibu:rwx
group:controlling:r-x
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
mask:rwx
other:---
default:user::rwx
default:user:mueller:---
default:group::r-x
default:group:fibu:rwx
default:group:controlling:r-x
default:mask:rwx
default:other:---
```

Mit diesen Einträgen ist das Beispielverzeichnis vollständig. Die Default-Einträge werden an neue Dateien und Verzeichnisse weitergegeben.

Zu beachten ist hier noch die umask des Benutzers. Sobald ACLs benutzt werden, wirken die Gruppenrechte, die im `ls` dargestellt und mit `chown` geändert werden, als *mask* einschränkend auf die ACLs. Wenn die umask eines Unix-Benutzers so gesetzt ist, daß die Gruppe eingeschränkt wird, so wirkt sich das beim Einsatz von ACLs so aus, daß bei neu angelegten Dateien und Verzeichnissen diese Gruppeneinschränkungen als *mask* wirken und somit die default ACLs beschränken.

18.6 ACLs aus Windows-Sicht

Was hat das ganze mit Samba zu tun? Zunächst einmal sind Windows-Administratoren gewohnt, deutlich stärker mit ACLs zu arbeiten, als Unixbenutzer. Dies mag daran liegen, daß Unix lange Zeit keine ACLs unterstützt hat und man vieles auch mit den traditionellen Zugriffsrechten erreichen kann. Bei der Planung eines Sambaservers als Ersatz für einen NT-Server stellt sich so zwangsläufig die Frage nach der Unterstützung von ACLs.

Der Zusammenhang mit Samba ergibt sich nun daraus, daß mit Samba 2.2 diese ACLs von Windows aus angeschaut und sogar gesetzt werden können. Dazu muß bei der Kompilation von Samba die Option `--with-acl-support` an das Skript `configure` übergeben worden sein, und beim Kompilieren muß die ACL-Unterstützung in den Headerfiles des Kernels vorhanden sein. Ist beides der Fall, kann man über die Sicherheitseigenschaften von Dateien und Verzeichnissen deren ACLs editieren. Dabei ergibt sich bei der Umsetzung von den sehr komplexen NT-ACLs zu den deutlich einfacheren Posix-ACLs häufig eine gewisse Einschränkung der Funktionalität, aber dies ist leider nicht zu vermeiden. Die Anforderungen, die im obigen Beispiel skizziert wurden, werden jedoch korrekt umgesetzt. Wer sich für die genaue Umsetzung der ACLs zwischen der NT- und der Posix-Welt interessiert, mag sich die Datei `samba-acls.ag.pdf` aus dem Unterverzeichnis `slides` eines Samba-FTP Servers ansehen. Dies sind die Folien eines Vortrags von Jeremy Allison über jene Umsetzung, den er bei der CIFS 2001 Konferenz in Bellevue gehalten hat.

19 oplocks

Dateizugriffe über ein Netzwerk sind trotz leistungsfähiger Netzwerkhardware meistens deutlich langsamer als auf einer lokalen Festplatte. Der lokale Hauptspeicher, der heutzutage in den meisten Workstations im Überfluß vorhanden ist, ist nochmal um Größenordnungen schneller. Für lokale Festplatten gibt es in allen Systemen Caches im Hauptspeicher, und so wäre es Verschwendung, Caching nicht auch auf Netzwerkdateien anzuwenden. Netzwerkzugriffe, die gar nicht erst gemacht werden müssen, sind die schnellsten Zugriffe. Im Gegensatz zu einem lokalen Festplattencache kann ein Cache von Netz-

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

werkdateien nicht davon ausgehen, die Datei alleine zu benutzen[10]. Zugriffe unterschiedlicher Clients müssen koordiniert werden.

Opportunistic Locks (Oplocks) sind ein Mechanismus, mit dem Clients erlaubt werden kann, Dateiinhalte zu cachen. Mit einem Oplock bekommt der Client eine Datei solange exklusiv für sich, bis der Server ihn auffordert, die Änderungen zurückzuschreiben und die Sperre freizugeben.

Ein Client A möchte eine Datei öffnen und beantragt ein Oplock auf die Datei. Wenn der Server dieses Oplock gewährt, ist das die Zusage, daß niemand anders auf die Datei zugreift. Damit muß Client A weder bei jedem Lesezugriff den Server befragen, noch muß er jeden Schreibzugriff unverzüglich an den Server liefern. Moderne Windowsclient nutzen dieses Feature sehr ausgiebig und erreichen damit in typischen Applikationen zwischen dreißig und vierzig Prozent mehr Geschwindigkeit.

Wenn ein zweiter Client, B, auf die Datei öffnen möchte, schickt der Server dem Client A ein so genanntes Oplock Break. Dies ist die Anweisung, sämtliche lokalen Änderungen zurückzuschreiben und den Schreibcache auf dieser Datei in Zukunft auszuschalten. Erst nachdem Client A alle Änderungen zurückgeschrieben hat, wird Client B die Datei öffnen können. Da keiner von beiden noch ein Oplock bekommt, sehen beide sämtliche Änderungen sofort.

Dieses Schema funktioniert innerhalb von Samba hervorragend. Sobald Unix-Prozesse ebenfalls auf Dateien zugreifen müssen, die von Samba freigegeben sind, gibt es Probleme mit Oplocks. Beispielsweise wird es nicht möglich sein, vernünftig Datensicherung zu betreiben, da Clients möglicherweise nicht alle Daten zum Server geschickt haben.

Es gibt mehrere Möglichkeiten, dieses Problem in den Griff zu bekommen:

Keine Oplocks: Durch den Parameter `oplocks = no` können Oplocks für eine Freigabe komplett abgeschaltet werden. Damit handelt man sich aber massive Performanceprobleme ein.

Keine Oplocks für einzelne Dateien: Der Parameter `veto oplock files` verweigert Oplocks für einzelne Dateien. Dieser Parameter verlangt eine Liste von Unix-Pfadnamen oder DOS-Wildcards. Die Syntax ist in der Beschreibung zum Parameter `veto files` zu finden.

Kernel Oplocks: Das Problem mit Oplocks liegt darin, daß Samba vom Kernel nicht informiert werden kann, wenn ein Unixprozeß eine Datei öffnen will. Samba könnte für diese Datei ein Oplock vergeben haben und müßte es vom Client zurückfordern, bevor der Unixprozeß die Datei öffnen kann. IRIX und Linux 2.4 sind um ein API erweitert worden, das genau dies leistet. Um Kernel Oplocks zu nutzen, muß Samba entsprechend kompiliert werden. Liegen bei der Kompilation die Quellen des Kernel 2.4 vor, erkennt Samba diese automatisch. Sollten sich bei der Nutzung dieses Features Probleme herausstellen, kann es mit `kernel oplocks = no` abgeschaltet werden, obwohl dies nie notwendig sein sollte.

Bevor Samba Oplocks unterstützt hat, konnte man mit `fake oplocks = yes` für read only Freigaben jegliche Oplocks vergeben, ohne sie jemals zurückzufordern. Dies sollte man heutzutage *nicht* mehr einsetzen.

20 Paßwörter

Protokolle der IP-Welt wie telnet, ftp und pop3 übertragen die Paßwörter zur Benutzerauthentifizierung im Klartext. Damit kann jeder, der den Netzverkehr abhören kann, sämtliche Paßwörter mitschreiben. Dafür existieren fertige Programme, die Benutzernamen und dazugehörige Paßwörter ausgeben. In

[10] Geteilte Blockgeräte in einem Storage Area Network sei hier einmal außen vor gelassen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

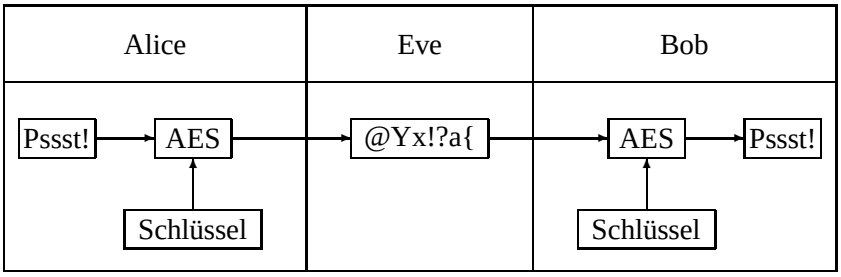


Abbildung 5: Symmetrische Verschlüsselung

der Unixwelt wurde dies zunächst nicht als problematisch angesehen, da zum Zugriff auf das Netz Administratorrechte oder physikalischer Zugriff zum Netz notwendig sind. Beides war historisch oft nicht gegeben, so daß das Risiko als relativ gering eingeschätzt wurde. Seit dem Aufkommen von DOS und Ethernet hat jeder Benutzer Administratorrechte, kann also den Netzverkehr mitschneiden.

Benutzerauthentifizierung muß vor allem eins leisten: Der Benutzer muß beweisen, daß er sein Paßwort kennt. Ein Authentifizierungsprotokoll kann es dabei ermöglichen, daß das Paßwort nicht übertragen werden muß.

Klassische symmetrische Verschlüsselung funktioniert folgendermaßen: Jemand möchte einem Bekannten[11] eine geheime Nachricht zukommen lassen. Das heißt, jemand, der die Übertragung abhört, soll diese Nachricht nicht lesen können. Dazu kann man ein symmetrisches Verfahren wie DES, IDEA oder AES einsetzen. Diese Verfahren zerstückeln die Nachricht so, daß sie niemand mehr lesen kann, außer jemand weiß, mit welchem Verfahren die Nachricht verschlüsselt wurde. Zu jedem Verschlüsselungsverfahren gibt es nämlich ein Gegenstück, das aus der zerstückelten Nachricht das Original wieder herstellt. Es gibt auch Verfahren, bei denen es keinen Rückweg gibt. Diese sind zwar für die genannte Anwendung nicht brauchbar, denn man kommt nicht mehr an die Nachricht, aber in anderen Bereichen sind diese so genannten Hashverfahren sehr weit verbreitet.

Alle heute verwendeten symmetrischen Verschlüsselungsverfahren verwenden zusätzlich Schlüssel. Erst mit einem Schlüssel wird die genaue Methode festgelegt, mit der die Nachricht verschlüsselt wird. Jemand, der die verschlüsselte Nachricht liest, muß also nicht nur das grundsätzliche Verfahren kennen, sondern insbesondere muß er den Schlüssel herausbekommen, um die Nachricht lesen zu können. Das Raten des Schlüssels ist meistens der viel schwierigere Teil, da es sehr viel mehr Schlüssel gibt als Verschlüsselungsverfahren. An dieser Stelle kommt die vielzitierte Schlüssellänge ins Spiel. Wenn ein Schlüssel wie bei DES 56 Bit lang ist, dann gibt es $2^56 = 72.057.594.037.927.936$ verschiedene Schlüssel. Mit heutiger Technologie können diese Schlüssel alle in kurzer Zeit ausprobiert werden, daher arbeiten moderne Verfahren mit mindestens 128 Bit langen Schlüsseln. Man nimmt an, daß 2^128 Schlüssel auch in der absehbaren Zukunft nicht alle durchprobiert werden können. Abbildung 5 verdeutlicht die symmetrische Verschlüsselung.

20.1 Challenge-Response Verfahren

Werden im SMB-Protokoll verschlüsselte Paßwörter verwendet, so wird die symmetrische Verschlüsselung trickreich eingesetzt. Der Client möchte eine Verbindung zum Server aufbauen. Bevor dies

[11] In der Literatur heißen diese beiden Bekannten Alice und Bob. Es gibt ganze Abhandlungen dazu, was diesen beiden schon alles passiert ist...

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

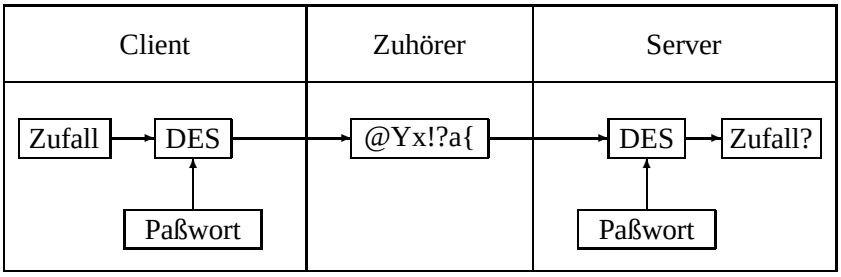


Abbildung 6: Verschlüsselung der Herausforderung

geschieht, muß der Benutzer seinen Namen und sein Paßwort eingeben. Erst danach baut der Client die Verbindung zum Server auf. In der Antwort auf die erste Anfrage des Clients, der Negotiate Protocol Response, schickt der Server dem Client eine Zufallszahl. Diese Zufallszahl wird Herausforderung genannt.

Der Client verfügt nun über drei Werte: Den Benutzernamen, das Paßwort des Benutzers und die Herausforderung. Das Paßwort soll nun verschlüsselt über das Netz übertragen werden. Ein naiver Ansatz wäre, die Herausforderung als Schlüssel für ein symmetrisches Verschlüsselungsverfahren einzusetzen. Der Server kennt die Herausforderung, da er sie selbst verschickt hat, kann also das verschlüsselte Paßwort wieder entschlüsseln. Das Problem liegt darin, daß jeder Zuhörer ebenfalls den Schlüssel kennt, also auch das Paßwort entschlüsseln kann. Daher wird anders vorgegangen: Das Paßwort wird als Schlüssel benutzt, um die Herausforderung zu verschlüsseln. Diese mit dem Paßwort verschlüsselte Herausforderung schickt der Client im Session Setup zusammen mit dem Benutzernamen an den Server.

Worüber verfügt der Server, wenn er den Session Setup erhalten hat? Er hat sich die Zufallszahl gemerkt, und der Client hat im den Benutzernamen geschickt. Aus der Benutzerdatenbank kann er damit das Paßwort des Benutzers auslesen. Mit diesem ausgelesenen Paßwort als Schlüssel entschlüsselt der Server die verschlüsselte Herausforderung und prüft, ob wieder die versendete Zufallszahl herauskommt. Ist dies der Fall, stimmen die beiden Schlüssel überein. Das heißt, der Client hat die als Herausforderung gesendete Zufallszahl mit dem gleichen Paßwort verschlüsselt, das auch der Server in seiner Benutzerdatenbank gespeichert hat. Stimmt der entschlüsselte Wert nicht mit der gesendeten Zufallszahl überein, wurde für die Verschlüsselung ein anderer Schlüssel, also ein anderes Paßwort, benutzt, als für die Entschlüsselung. Das am Client eingegebene Paßwort stimmt also nicht mit dem überein, das der Server in seiner Benutzerdatenbank gespeichert hat. Der Server bekommt nicht heraus, welches Paßwort der Client benutzt hat, aber das muß er auch gar nicht. Das Paßwort war in jedem Fall falsch.

Abbildung 6 verdeutlicht die verwendete Verschlüsselung, Abbildung 7 den zeitlichen Ablauf des Protokolls.

Warum ist das Verfahren sicher? Die mit dem Paßwort verschlüsselte Herausforderung hat den Server davon überzeugt, daß der Benutzer sein Paßwort kennt. Man könnte vermuten, daß man diese verschlüsselte Herausforderung einfach nochmal schicken muß, um die Rechte des Benutzers zu bekommen. Dieser so genannte Replay-Angriff schlägt jedoch fehl, da bei jeder neuen Anmeldung eine neue Herausforderung verschlüsselt werden muß. Dies gilt natürlich nur, wenn der Server sich jedes Mal eine neue Herausforderung ausdenkt…

Windows NT verhält sich diesbezüglich vernünftig. Windows 95 denkt sich jedoch nur alle 15

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

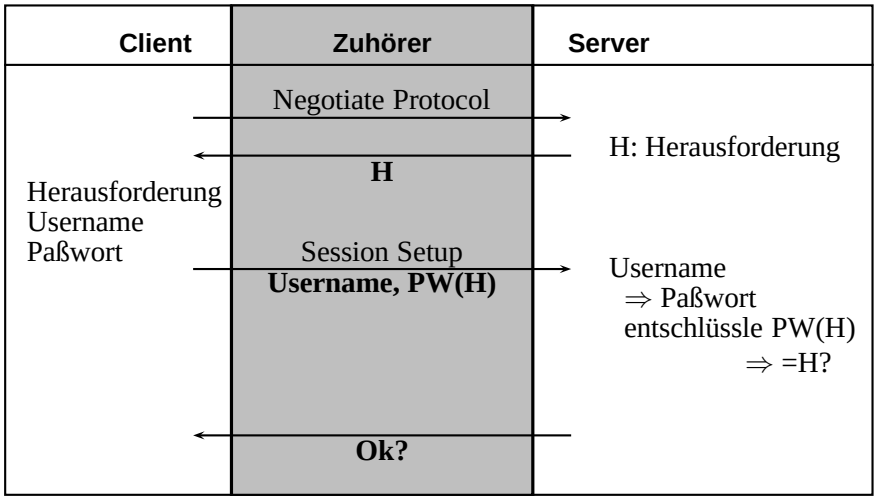


Abbildung 7: Challenge-Response Verfahren

Minuten eine neue Herausforderung aus. Das heißt, daß jemand nur einen Verbindungsaufbau mitschneiden muß, und sich sofort danach mit der gleichen Benutzerkennung bei der gleichen Maschine anmelden kann. Man kann sich fast sicher darauf verlassen, die gleiche Herausforderung zu bekommen, und mit der mitgeschnittenen Antwort Zugriff zu erhalten. Dies gilt selbstverständlich nur für die Zugriffe, bei denen Windows 95 als Server benutzt wird. Und wer tut das schon?

Ein Zuhörer verfügt über die Herausforderung und den verschlüsselten Wert. Mit diesen beiden Werten könnte er einen Known-Plaintext-Angriff gegen die Verschlüsselung starten. Das heißt, es muß ein Verschlüsselungsalgorithmus gewählt werden, der gegen einen solchen Angriff für alle praktischen Belange immun ist.

20.2 Vor- und Nachteile von verschlüsselten Paßwörtern

Das Challenge-Response Verfahren setzt voraus, daß der Server über das Benutzerpaßwort im Klartext verfügt. Unter Unix tut er das nicht, sondern der Server kennt nur eine zerhackte Version des Paßwortes. Die meisten Linuxsysteme speichern diesen Wert in der Datei `/etc/shadow`, andere Unixe können die Paßwortdatenbank in anderen Dateien abspeichern. Der Wert, der dort gespeichert wird, ist für die Authentifizierung benutzbar. Der Server ist jedoch nicht in der Lage, daraus das Klartextpaßwort des Benutzers zu berechnen.

Die Authentifizierung unter Unix benutzt eine Hashfunktion, die drei Eigenschaften erfüllt:

1. Sie ist leicht zu berechnen. Dies ist notwendig, damit die Paßwortüberprüfung nicht zu lange dauert.

2. Sie ist nur sehr schwer umkehrbar. Das heißt, aus dem zerhackten Paßwort ist das Klartextpaßwort nicht berechenbar. Als Beispiel für eine solche Einwegfunktion soll hier die Multiplikation herhalten. 98453*34761=3422324733 ist relativ einfach zu berechnen. Daß die Zahl 3422324733 aus den beiden Ursprungszahlen entstanden ist, ist schon sehr viel schwieriger herauszufinden. Es gibt Verfahren, bei denen es keinen Rückweg gibt, der irgendwie berechnet werden kann. Um

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

für einen Funktionswert den Ausgangswert herauszubekommen, muß man alle möglichen Ausgangswerte durchprobieren oder gleich eine Wertetabelle mit allen Ausgangswerten anlegen.[12].

Mit dieser Eigenschaft war es zu rechtfertigen, daß in den frühen Tagen von Unix die Hashwerte der Paßwörter für alle Benutzer lesbar waren, da niemand daraus etwas ableiten konnte. Mit dem Überfluß an Rechenleistung kann man aber so genannte crack-Programme verwenden, die die erste Eigenschaft der Hashfunktion ausnutzen: Sie probieren einfach tausende von Paßwörtern pro Sekunde aus. Schlechte Paßwörter können so sehr schnell gefunden werden. Daher hat man die Paßwörter in die nicht allgemein lesbare Datei `/etc/shadow` ausgelagert.

3. Zwei verschiedene Paßwörter führen zu zwei verschiedenen Hashwerten. Damit kann das Loginprogramm ausreichend sicher sein, daß ein korrekter Hashwert aus dem korrekten Paßwort entstanden ist.

Authentifizierung unter Unix setzt voraus, daß der Client dem Server das Klartextpaßwort präsentiert. Der Server kann daraus den Hashwert berechnen, und mit dem gespeicherten Wert vergleichen. Leider verfügt er nicht über das Klartextpaßwort des Benutzers, um das Challenge-Response Verfahren durchführen zu können. Daher muß unter Samba für die Paßwortverschüsselung eine zweite Paßwortdatenbankgepflegt werden, die Datei `smbpasswd`.

Was bis jetzt beschrieben wurde, entspricht nur fast der Wahrheit. Oben wurde beschrieben, daß die Verschlüsselung der Herausforderung mit dem Paßwort des Benutzers geschieht. Dies ist so nicht ganz richtig. Die Verschlüsselung geschieht mit einem Hashwert des Paßwortes. Dieser Hashwert wird vom Client direkt nach Eingabe des Paßwortes gebildet und gespeichert. Das gesamte oben beschriebene Verfahren wird dann mit diesem Hashwert durchgeführt. Das heißt, daß auch in der Datei `smbpasswd` keine echten Klartextpaßwörter gespeichert werden müssen, sondern diese Hashwerte. Das heißt, daß man mit den dort enthaltenen Werten so direkt nicht mehr anfangen kann als mit den Werten aus der Datei `/etc/shadow` unter Unix. Wenn man sie als Paßwort eingeben würde, würde Windows sofort wieder den Hash darauf anwenden, und einen anderen, also falschen Wert daraus errechnen. Das Programm `smbclient` muß diese Operation ebenfalls durchführen, nur hat man hierzu den Quellcode und kann die entsprechenden Stellen auskommentieren. So hat man die Möglichkeit, sich anhand der Werte in der `smbpasswd` ohne Einsatz von crack bei einem NT-Rechner anzumelden. Damit sind die Werte aus der `smbpasswd` so gut wie Klartextpaßwörter.

Alles nicht dramatisch, sagt Microsoft. Das Äquivalent zur Datei `smbpasswd` liegt unter NT verschlüsselt vor. Diese Verschlüsselung muß jedoch reversibel sein, um das Challenge-Response Verfahren durchführen zu können. Ein Teil der Sicherheitsargumentation liegt darin, daß dieses Verschlüsselungsverfahren nicht offengelegt wurde. Das Verfahren war solange geheim, bis Jeremy Allison das Programm `pwdump` veröffentlicht hat. Dieses Programm extrahiert aus der Benutzerdatenbank von NT eine Datei, die direkt als `smbpasswd` verwendet werden kann [13].

Das heißt, der Administrator unter NT verfügt direkt über die Paßwörter aller Benutzer oder zumindest über etwas Gleichwertiges. Damit hat er automatisch die Möglichkeit, sich bei fremden Systemen anzumelden, sofern dort das Paßwort gleich ist. Bei Unix kann sich der Administrator zwar in die Identität jedes Benutzers versetzen. Dies bleibt aber auf das lokale System beschränkt, da er das Paßwort des Benutzers nicht kennt. Windows 2000 mit dem dort eingesetzten Kerberos-Verfahren ist in dieser Hinsicht übrigens nicht besser. Der Domänencontroller kennt hier ebenfalls die Klartextpaßwörter der Benutzer. Ihm wird also genau so vertraut wie einem NT4-Domänencontroller.

[12] Wie überall in der Kryptographie gilt dies auch nur so lange, bis jemand den Rückweg gefunden hat.
[13] Allerdings nur für Samba 1.9, zu 2.0 hin wurde das Format geändert. Es gibt in Samba 2.0 aber ein Konvertierungsskript.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Sollte ein neugieriger Administrator einmal an den tatsächlichen Klartextpaßwörtern seiner Benutzer interessiert sein, dann macht NT es ihm deutlich einfacher als Unix dies tut. Unix verwendet so genannte versalzene Paßwörter. Wenn ein Paßwort geändert wird, dann wird ein Zufallswert berechnet, dem Paßwort hinzugefügt und dann die Hashfunktion durchgeführt. Der Zufallswert wird der Datei `/etc/shadow` im Klartext hinzugefügt, damit die Überprüfung die gleichen Operationen durchführen kann. So kann man keine Tabelle von Paßwörtern und den zugehörigen Hashwerten anlegen. Man kann auch nicht erkennen, wenn zwei Benutzer das gleiche Paßwort verwenden. Windows NT verwendet dieses Verfahren nicht.

Aus Kompatibilitätsgründen muß NT auch noch zusätzlich einen sehr schlechten Hashwert mitführen. Bei alten Windowsversionen konnte das Paßwort bis zu 14 Zeichen lang sein. War es kürzer, wurde es mit Leerzeichen aufgefüllt. Dann wurde mit den ersten 7 Zeichen ein Hashwert berechnet, und dann mit den zweiten 7 Zeichen. Das heißt, es sind sofort alle Paßwörter erkennbar, die weniger als 7 Zeichen haben, da die zweite Hälfte des Hashwertes immer gleich ist.

20.3 NT4 Service Pack 3

Um die Paßwortverschlüsselung im Zusammenhang mit Windows NT 4 Service Pack 3 und Windows 95 in späteren Versionen gibt es immer noch weitverbreitete Mißverständnisse. Beispielsweise daß alle Systeme vorher nicht in der Lage waren, mit verschlüsselten Paßwörtern zu arbeiten. Richtig ist folgendes:

- *Alle* Clients sind in der Lage, mit verschlüsselten Paßwörtern umzugehen. Das gilt für alle aktuellen Clients sowieso. Aber sogar der DOS-LanManager-Client, den man sich heute noch von Microsofts FTP-Server laden kann, kann Paßwörter verschlüsseln.
- Auch die neuen Clients können sowohl mit verschlüsselten Paßwörtern, als auch mit Klartextpaßwörtern umgehen.
- Windows NT4 Service Pack 3 ist das erste NT-System, das sich in der Default-Einstellung weigert, Klartextpaßwörter zu verschicken.

Ein Client wirkt an der Entscheidung über verschlüsselte Paßwörter zunächst einmal überhaupt nicht mit. Der Server wird für verschlüsselte oder für Klartextpaßwörter mit der Einstellung `encrypt passwords` konfiguriert. In der Antwort zum Negotiate Protocol teilt der Server dem Client seine Entscheidung mit. Der Server verschickt im Falle der verschlüsselten Paßwörter noch eine Herausforderung mit. Der Server teilt dem Client möglicherweise mit, daß er ein Klartextpaßwort sehen will. Der Client kann nur noch die Verbindung sofort abbrechen, sofern er keine Paßwörter im Klartext verschicken möchte. Windows NT tut dies ab Service Pack 3 mit der Fehlermeldung, daß man sich micht diesem Konto nicht an dem Server anmelden kann.

Windows 95 und folgende fragen immer wieder nach dem Kennwort für die Freigabe `IPC$`. Für alle Clientbetriebssysteme liefert Samba im Unterverzeichnis `docs/` Registrierungsdateien mit, mit denen diese Verweigerung von Klartextpaßwörtern abgestellt werden kann.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Mit Klartextpaßwörtern bekommt man den großen Vorteil, daß man nicht zwei verschiedene Paßwortdatenbanken pflegen muß. Einige Nachteile handelt man sich jedoch ein:

- Man muß heutzutage jeden Client anfassen, um die Registrierung zu ändern.
- Man versendet Paßwörter im Klartext, man hat also ein möglicherweise erhebliches Sicherheitsproblem. Tools wie `dsniff`[14] sammeln die Paßwörter automatisch auf.
- Man verliert jegliche Domänenfunktionalität, auf die weiter unten noch genauer eingegangen wird. Ein Domänencontroller kann nur mit verschlüsselten Paßwörtern funktionieren.

Insgesamt kann man nur zu verschlüsselten Paßwörtern raten, wenn nicht wirklich wichtige Gründe für Klartextpaßwörter sprechen.

20.4 Migration zu verschlüsselten Paßwörtern

Sind momentan Klartextpaßwörtern auaf einem Server im Einsatz, und ist die Migration zu verschlüsselten Paßwörtern geplant, gibt es einen sehr einfachen Weg, dies binnen einer Woche ohne große Arbeit zu erledigen. Direkt aus der `/etc/shadow` bekommt man die die NT- und LanManager-Hashes leider nicht heraus, da man dazu die Klartextpaßwörter benötigt. Nur aus diesen können die Windows-Hashwerte berechnet werden. Die Clients liefern die Paßwörter jedoch bei jedem Anmelden im Klartext zum Server, und daraus können dann die Hashwerte berechnet werden. Dazu muß man aus der `/etc/passwd` mit dem Skript `mksmbpasswd.sh` zunächst eine Datei `smbpasswd` machen, in der alle Benutzer mit leerem Paßwort angelegt sind. Dieses Skript findet sich in den Samba-Quellen im Unterverzeichnis `source/script`. Die neu angelegte `smbpasswd` muß dann mit den korrekten Zugriffsrechten versehen werden:

```
sh mksmbpasswd.sh < /etc/passwd > /etc/smbpasswd
chown root.root /etc/smbpasswd
chmod 700 /etc/smbpasswd
```

Die Migration leitet man mit den folgenden Einstellungen ein:

```
[global]
   encrypt passwords = no
   update encrypted = yes
```

Jeder Benutzer, der sich anmeldet, liefert sein Paßwort im Klartext an den Server. Dieser berechnet daraus die beiden Windows-Hashwerte und trägt sie in der `/etc/smpasswd` ein. Das heißt, man muß jetzt nur abwarten, bis sich alle Benutzer einmal angemeldet haben, und kann dann verschlüsselte Paßwörter aktivieren:

```
[global]
   encrypt passwords = yes
   update encrypted = no
```

[14] Suchen Sie einmal auf http://freshmeat.net nach dsniff und lesen Sie das README.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

21 Druckfreigaben

Um Drucker unter Samba zur Verfügung zu stellen, müssen diese von Unix aus ansprechbar sein. Unter Linux mit einem BSD-kompatiblen Drucksystem geschieht dies durch Einträge in der Datei `/etc/printcap`. Alle Drucker, die dort definiert sind, kann man als Netzwerkdrucker für Windowsclients freigeben.

Unter Linux ist die Frage der Druckertreiber noch nicht zufriedenstellend gelöst. Druckertreiber unter Windows würde man unter Linux nicht als solche bezeichnen. In der Linuxwelt sind Treiber Softwaremodule, die direkt Hardware wie Netzwerkkarten oder den parallelen Port ansprechen. Druckertreiber im Sinne von Windows sind unter Linux so genannte Filter, die Druckdaten in ein für den Drucker akzeptables Format aufbereiten. Das einheitliche Druckformat unter Linux ist Postscript, das mit dem Programm Ghostscript in viele druckereigene Formate umgewandelt werden kann. Druckertreiber unter Windows gehen vom Windows Metafile-Format aus, und wandeln dies entsprechend um. Das Windows Metafile-Format enthält Aufrufe an die Graphische Komponente von Windows, das GDI.

Wenn man einen Drucker, der über Unix angesprochen wird, von Windows aus nutzen möchte, muß man planen, wo die Aufbereitung in das druckereigene Format geschehen soll. Zwei Wege sind denkbar.

- Auf den Arbeitsplätzen wird ein generischer Postscripttreiber installiert. Die Clients müssen nicht wissen, welches Druckermodell sich hinter einer Freigabe verbirgt. Die Umwandlung findet auf dem Druckerserver mittels `ghostscript` statt.
- Der Druckertreiber reicht die Daten weiter, ohne sie weiter zu behandeln. Auf den Arbeitsplätzen werden für jeden Netzdrucker die korrekten Treiber installiert.

Beide Wege haben Vor- und Nachteile. Im ersten Fall hat man weniger Aufwand mit der Administration auf Clientseite. Man muß den korrekten „Druckertreiber“ nur einmal definieren, am Druckerserver. Beim zweiten Weg kann man die bessere Unterstützung der Druckerhersteller für die Windowsplattformen nutzen. Druckertreiber für Windows bieten in der Regel die Möglichkeit, Sonderfunktionen wie die Auswahl des Papierschachtes zu nutzen. Dieser erhöhte Komfort zieht jedoch nach sich, daß auf jedem Client der korrekte Druckertreiber installiert ist.

Nutzt eine Windows NT Workstation einen Drucker, der von einem Windows NT Server freigegeben wurde, so gibt es noch die Möglichkeit, die Druckaufbereitung komplett vom NT Server vornehmen zu lassen, und trotzdem sämtliche Komfortfunktionen auf der Workstation zu nutzen. Dazu muß auf der Workstation kein Druckertreiber installiert sein. Diese so genannten EMF-Druckerwarteschlangen kann Samba zur Zeit nicht exportieren. Samba wird dies voraussichtlich auch nicht so schnell ermöglichen, da hierfür große Teile von Windows, nämlich das GDI, auf Sambaseite implementiert werden müßte.

Eine Druckfreigabe wird genau wie eine Dateifreigabe in einem eigenen Abschnitt erstellt, wobei für die Druckfunktion drei Optionen notwendig sind:

```
[deskjet]
printable = yes
printer = lp
path = /tmp
```

Zu einer Druckfreigabe wird die Definition durch die Angabe `printable = yes`.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Mit der Option `printer =` wird festgelegt, welche Druckerwarteschlange unter Unix angesprochen werden soll. Diese Warteschlange muß das Format verstehen, das vom Windowsdruckertreiber geliefert wird. Also sollte hier entweder Postscript angenommen werden, oder die Daten sollten per so genannter Raw-Queue direkt ohne Umwandlung an den Drucker weitergeleitet werden.

Die Option `path =` legt einen Spoolbereich fest. Ein Druckjob, den ein Windowsrechner an Samba schickt, muß zunächst in einer Datei abgespeichert werden. Wenn diese Datei geschlossen wird, teilt der Client dem Server mit, daß diese nun zum Drucker geschickt werden soll. Samba realisiert dies, indem das Programm `lpr` mit der Druckdatei als Argument aufgerufen wird. Samba muß also für sich die Möglichkeit haben, Druckjobs in Dateien zu speichern, bevor sie an den `lpd` übergeben werden. Dies sollte nicht das Spoolverzeichnis sein, das der `lpd` selbst für den Drucker vorsieht.

22 Windows NT Domänen

Installiert man eine Arbeitsgruppe von Windows NT Rechnern, dann bekommt man komplett getrennte Benutzerdatenbanken auf den einzelnen Rechnern. Erstellt man auf einem Server eine Freigabe und möchte für diese Freigabe Rechte vergeben, so muß man zunächst die Benutzer[15] einrichten, die Rechte auf dieser Freigabe bekommen sollen. Greift ein Benutzer von einer anderen Workstation auf die Freigabe zu, so probiert die Workstation das so genannte transparente Anmelden: Die Workstation versucht es erst einmal mit dem lokal angemeldeten Benutzer und seinem Paßwort. Dadurch sieht es so aus, als ob man nur ein Benutzerkonto verwenden würde.

Die Administration der Benutzerdatenbanken kann komplett von einem zentralen Rechner aus erfolgen. Dazu benötigt man den Benutzermanager für Domänen[16], der normalerweise bei Windows NT Server mitgeliefert wird. Man kann sich diesen aber auch kostenlos von Microsoft von der Webseite `http://www.microsoft.com/` beziehen. Man muß zu dem Rechner, den man administrieren möchte, eine Verbindung als Administrator aufbauen. Dazu muß man auf der Workstation, von der aus man administriert, auf der Kommandozeile mit

```
net use \\remote\ipc$ /user:administrator
```

eine Verbindung aufgebaut werden. Kommt dann die Fehlermeldung *Die Referenzen passen nicht zu einer bestehenden Referenzenmenge*, so besteht unter einer anderen Benutzerkennung bereits eine Verbindung. In diesem Fall muß man sich ab- und neu anmelden, und den Befehl als allererstes absetzen, bevor irgend eine Verbindung zum entfernten Rechner `remote` aufgebaut werden kann. Hat man eine solche Verbindung, kann man im Benutzermanager für Domänen im Menüpunkt *Domäne auswählen* mit `\\remote` die Benutzerdatenbank von `remote` auswählen und voll administrieren.

Diese Art der Administration skaliert nicht besonders gut. Jeden Benutzer muß es auf jedem Server geben, die lokalen Workstations brauchen ebenfalls separat gepflegte Benutzer. Mit Windows NT wurde, um dieses Problem zu lösen, das Domänenkonzept eingeführt. Mit einer Windows NT Domäne bekommt jeder Benutzer ein zentrales Konto, das auf allen Domänenmitgliedern gültig ist.

Realisiert ist die Domäne durch einen speziellen Rechner, den Primary Domain Controller PDC, der seine Benutzerdatenbank für andere im Netz zur Verfügung stellt. Alle Domänenmitglieder importieren diese Benutzerdatenbank. Somit sind auf den Domänenmitgliedern zwei Benutzerdatenbanken gültig: Die lokale und die des PDC.

[15] Windows NT benutzt grundsätzlich `security = user`
[16] Benutzermanager für Domänen

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Die Kommunikation zwischen der Workstation und dem Primary Domain Controller läuft verschlüsselt ab. Um eine solche Verschlüsselung zu ermöglichen, muß ein gemeinsamer Schlüssel vereinbart werden. Um sich über einen Schlüssel einig zu werden, gibt es spezialisierte Protokolle, wie beispielsweise den Diffie-Hellmann Schlüsselaustausch. Um jeglichen Problemen mit Patenten oder Exportrestriktionen zu umgehen, ist Microsoft einen anderen Weg gegangen. Beim Schlüsselaustausch geht es im wesentlichen darum, sich über ein gemeinsames Geheimnis einig zu werden. Um ein gemeinsames Geheimnis zu wahren und zu prüfen, kennt Microsoft bereits eine Gruppe von Protokollen: Die Protokolle zum Prüfen und Austauschen von Benutzerpaßwörtern. Genau diese Protokolle werden verwendet, um die Kommunikation zwischen PDC und Workstation zu sichern. Das heißt, es muß für jedes Domänenmitglied ein Benutzerkonto auf dem PDC geben, damit für dieses Konto ein Paßwort vergeben werden kann. Dieses Benutzerkonto heißt üblicherweise Computerkonto.

23 Samba als Primary Domain Controller

Um Samba als PDC zu konfigurieren, sind in der `smb.conf` im Abschnitt `[global]` 2 Einstellungen notwendig:

```
domain logons = yes
domain master = yes
```

Eine vollständige `smb.conf` für einen PDC sieht damit folgendermaßen aus:

```
[global]
  workgroup = samba
  encrypt passwords = yes
  domain master = yes
  domain logons = yes
```

Daß ein PDC auch gleichzeitig Domain Master Browser sein muß, ist eine Einschränkung der Implementation der Microsoft-Clients. Eigentlich hat die Funktion des Domain Master Browsers (siehe Abschnitt 10) nichts mit der Funktion als zentraler Server für die Benutzerdatenbank zu tun. Die Clientimplementation von Microsoft setzt aber voraus, daß beide Funktionen auf einer Maschine vereinigt sind. Auch funktionieren die Domänenfunktionen ausschließlich mit verschlüsselten Paßwörtern. Ist man auf Klartextpaßwörter angewiesen, kann man Samba nicht als PDC einsetzen.

Befinden sich Windows 9x Clients im Netz, können diese den Samba-PDC sofort ohne weitere Konfiguration als Anmeldeserver nutzen. Dazu trägt man in den Eigenschaften des Clients für Microsoft-Netzwerke ein, daß sich die Clients an der Samba-Domäne anmelden müssen. Ist dies erfolgreich, so kann man über die Systemsteuerung des Clients direkt sein SMB-Paßwort auf dem Server ändern.

23.1 Manuelles Erstellen der Computerkonten

Für Domänenmitglieder unter Windows NT oder 2000 müssen noch die Computerkonten erstellt werden. Jedes Maschinenkonto muß unter Unix als normaler Benutzer existieren. Dieser Benutzer braucht weder ein Unixpaßwort, noch eine Login-Shell oder ein Heimatverzeichnis. Der Name des Benutzers ist der Name der Workstation, ergänzt um ein $-Zeichen. Erstellt wird ein solcher Benutzer für die Workstation `WKS` unter Linux beispielsweise mit

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
root@erde: useradd -d /dev/null -s /bin/false wks\$
root@erde: smbpasswd -a -m wks
```

Der Befehl `smbpasswd -a -m wks` fügt den Benutzer mit einem Standardpaßwort in die Datei `smbpasswd` ein. Das Standardpaßwort für Computerkonten ist der Name der Workstation, in diesem Fall also `wks`. Man beachte, daß beim Befehl `useradd` ein Dollarzeichen, maskiert durch den Backslash, hinzugefügt wurde. Der Befehl `smbpasswd` fügt diesen bei Verwendung des Parameters `-m` selbst hinzu.

Nachdem das Computerkonto auf dem PDC erstellt wurde, kann in den Eigenschaften der Netzwerkumgebung in die Domäne gewechselt werden. Dabei wird das Paßwort des Computerkontos geändert. Sollte aus irgendwelchen Gründen ein erneutes Betreten der Domäne notwendig sein, dann muß der Befehl `smbpasswd -a -m wks` erneut ausgeführt werden, um das Paßwort des Computerkontos auf den Anfangswert zurückzusetzen.

23.2 Automatisches Erstellen der Computerkonten

Windows NT 4 bietet in dem Dialog, in dem in die Domäne gewechselt wird, die Möglichkeit, das Computerkonto automatisch erstellen zu lassen. Dies geschieht unter Angabe eines Benutzers und Kennwortes, der auf dem PDC berechtigt ist, Computerkonten zu erstellen. Dies ist unter Unix nur *root*, da die `/etc/passwd` hierzu geändert werden muß.

Um das Computerkonto automatisch erstellen zu lassen, müssen zwei Dinge auf dem PDC konfiguriert sein:

- *root* oder ein anderer Benutzer mit der UID 0 muß ein Paßwort in der `smbpasswd` haben. Dieses Paßwort muß nicht mit dem Systempaßwort von *root* übereinstimmen. Wenn man nicht *root*, sondern beispielsweise einen Benutzer *admin* mit der UID 0 verwendet, braucht dieser Benutzer nicht einmal eine Login-Shell auf Unix. Er muß nur in die `/etc/passwd` schreiben dürfen.
- Der Parameter `add user script` muß korrekt konfiguriert werden. Mit `add user script` wird ein Unix-Script angegeben, mit dem das Computerkonto in der `/etc/passwd` angelegt wird. Beispielsweise kann man mit

  ```
  add user script = useradd -d /dev/null -s /bin/false %u
  ```

  die gleiche Wirkung erzielen wie mit der manuellen Konfiguration aus dem letzten Abschnitt.

23.3 BDCs mit Samba

In einer echten NT Domäne gibt es zwei Arten von Domänencontrollern: Primäre Domänencontroller (PDCs) und Backup Domänencontroller (BDCs). Der PDC besitzt die Hauptkopie der Benutzerdatenbank SAM, die BDCs halten read-only Kopien der SAM vor, um Authentifizierungsanfragen von Workstations und Mitgliedsservern beantworten zu können. Alle Änderungen an der Benutzerdatenbank, beispielsweise Paßwortänderungen, müssen auf dem PDC durchgeführt werden. Der PDC überträgt diese Änderungen dann an die BDCs, damit diese wieder über den aktuellen Stand der Datenbank verfügen.

Samba 2.2.2 ist noch kein voller Ersatz für einen Backup Domain Controller in einer echten NT-Domäne. Auch kann Samba als PDC keinen echten NT-BDC mit Domänendaten versorgen. Die Protokolle zur Replikation der Benutzerdatenbank sind noch nicht vollständig implementiert. Das Samba

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Team, insbesondere Tim Potter, arbeitet jedoch daran, die Protokolle zu verstehen und in Samba zu integrieren. Vermutlich ist mit Erscheinen dieses Buches die echte BDC-Funktionalität bereits in Samba integriert.

Wird Samba als PDC eingesetzt, können weitere Sambaserver jedoch als Backup Domain Controller eingesetzt werden. Die Replikation der Benutzerdatenbank zwischen den Servern kann mit Unix-Bordmitteln vorgenommen werden. Die wesentliche Idee beim Einsatz eines BDC ist seine `smb.conf`:

```
workgroup = samba
encrypt passwords = yes
domain master = no
domain logons = yes
```

Der Unterschied zum PDC ist die Zeile `domain master = no` im Gegensatz zu `domain master = yes`. Mit dieser `smb.conf` sehen die Workstations den BDC als Domänencontroller für die Domäne `SAMBA` an.

Wenn eine Workstation einen Benutzer authentifizieren muß, tut sie dies nicht selbst, sondern sucht ihren Domänencontroller für die Bestätigung der Identität des Benutzers. Dies tut sie, indem sie eine NetBIOS-Namensanfrage nach dem Namen `SAMBA<1c>` absetzt. `SAMBA<1c>` ist ein NetBIOS-Gruppenname, den jeder Domänencontroller per Broadcast oder beim WINS-Server reserviert. Diese Reservierung wird bei Samba durch den Parameter `domain logons = yes` angestoßen. Im nächsten Schritt authentifizieren sich die Workstation und der Domänencontroller gegenseitig anhand des Workstationkontos. Dieses Workstationkonto muß somit sowohl auf dem PDC, als auch auf den BDCs vorhanden sein, damit die Workstation auch die BDCs als Domänencontroller akzeptiert. Auch die gesamte restliche Benutzerdatenbank muß vom PDC auf die BDCs übertragen werden.

23.4 Hintergrund: Benutzerdatenbanken und SIDs

Unter Unix besteht ein Benutzer im wesentlichen aus einer numerischen Userid, und nicht mehr. Das Programm `login` muß beim Anmelden des Benutzers anhand seines Namens herausfinden, welche numerische Userid er hat. Dazu sieht es in der Datei `/etc/passwd` nach. Mit der Datei `/etc/shadow` prüft `login` das Paßwort. Ist es korrekt, wird in die gefundene Userid umgeschaltet und die Loginshell des Benutzers gestartet. Nach diesem Vorgang ist es Unix völlig egal, wie der Benutzer heißt, das einzige, was interessiert, ist der numerische Wert. Damit hängt an jedem Prozeß eine endeutige Identifikation der Rechte, die er hat.

Unter Unix ist es so, daß Userids nur auf dem Rechner gelten, auf dem sie zugeordnet wurden. Es gibt keine Möglichkeit, Rechte von einem Rechner auf den nächsten zu übernehmen oder global Benutzer zuzuordnen. Die einzige Möglichkeit, die man zu Vereinheitlichung hat, ist der Austausch der jeweils auf einem Rechner geltenden Tabellen über verschiedene Rechner hinweg. Genau das tut NIS. Die Benutzerdatenbank wird im Netz kopiert, gilt aber auf jedem Rechner rein lokal.

Unter NT ist das zunächst genau so. Es gibt eine numerische Userid, der Name des Benutzers ist nur während der Anmeldung für das System interessant. Nach der Anmeldung ist nur noch die numerische Userid relevant. Windows NT Benutzer sind jedoch im Gegensatz zu Unix Benutzern über Rechnergrenzen hin gültig. Um dies zu erreichen, wird der Benutzer nicht durch eine kleine Zahl beschrieben, sondern durch einen so genannten Security Identifier SID. Dieser SID besteht aus zwei wesentlichen Teilen. Der erste Teil besteht aus einer 96 Bit langen Zahl, die die Benutzerdatenbank des SID eindeutig identifiziert. Der zweite Teil ist der so genannte Relative Identifier RID. Der RID ist mit der numerischen Userid unter Unix vergleichbar. Da eine Userid unter NT jedoch *nur* zusammen mit den 96 Bit

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

der Benutzerdatenbank verwendet werden, sind Benutzer unterschiedlicher Maschinen oder Domänen unterscheidbar.

Mit dieser eindeutigen Zuordnung von Benutzern zu ihren jeweiligen Benutzerdatenbanken wird es möglich, daß eine Workstation gleichnamige Benutzer aus mehreren Benutzerdatenbanken lokal völlig gleichwertig verwenden kann. Je nachdem, ob sich ein Domänenbenutzer oder ein lokaler Benutzer an der Workstation anmelden möchte, wird die lokale Benutzerdatenbank oder die des PDC um Bestätigung des Kennwortes gebeten. Ist dies erfolgt, ist der Benutzer dem System nur noch unter dem numerischen SID bekannt. Dabei ist es völlig gleichgültig, ob es sich bei diesem SID um einen lokalen, oder einen Domänen-SID handelt.

Jeder Sambaserver generiert beim ersten Start seine eigene Maschinenkennung und speichert sie in der Datei `MACHINE.SID` ab. Die Maschinenkennung wird spätestens dann benötigt, wenn der Sambaserver als Domänencontroller konfiguriert wird. Die Benutzer, die sich an den Workstations anmelden, müssen eine eindeutige Domänenkennung als Teil ihres SID bekommen. Selbst wenn der Sambaserver nicht als Domänencontroller fungiert, wird die Maschinenkennung verwendet. Beispielsweise bei der Anzeige der ACLs in den Sicherheitseigenschaften von Dateien und Verzeichnissen wird die Liste der Benutzer in Form eine SID-Liste übergeben. Diese SIDs müssen eindeutig sein und mit separaten Aufrufen in Benutzernamen übersetzt werden können.

24 Profile, Logon Scripts und Policies

Unter Unix gibt es für jeden Benutzer ein Heimatverzeichnis als einzigen Bereich im System, in dem der Benutzer schreiben kann. So etwas kennt Windows in dieser restriktiven Form nicht, da viele Anwendungen voraussetzen, überall im System schreiben zu können. Aus Kompatibilitätsgründen muß Windows also relativ offen sein. Dem Heimatverzeichnis am nächsten kommt unter NT das Profil des Benutzers, ein ihm zugeordneter Bereich unter `c:\winnt\profiles\<benutzername>`. Dort ist der Desktop, der benutzereigene Teil des Startmenüs, der Zweig HKEY_CURRENT_USER der Registry und vieles andere abgelegt. Also alles, was zur Arbeitsumgebung des Benutzers gehört.

Meldet sich ein Benutzer bei NT das erste Mal an, wird aus `c:\winnt\profiles\default user` eine Kopie in das benutzereigene Profil gelegt. Beim Anmelden an der nächsten Workstation wird der gleiche Vorgang wiederholt. Das heißt, jeder Benutzer hat an jeder Workstation ein anderes Profil. In einer Domänenumgebung möchte man natürlich erreichen, daß ein Benutzer sein Profil mitnehmen kann, daß er also an jedem Arbeitsplatz seine eigene Umgebung vorfindet. Windows löst dies mit den serverbasierten Profilen.

Für jeden Benutzer kann ein Pfad angegeben werden, in dem sein Profil abgelegt wird. Viele Anwendungen setzen aber voraus, daß das Profil auf einer lokalen Platte abgelegt wird. Folglich kopiert Windows beim Anmelden des Benutzers das Profil von seinem Serverpfad nach `c:\winnt\profiles` und bei jedem abmelden wieder zurück auf den Serverpfad.

Der Pfad für das serverbasierte Profil wird bei Samba mit dem Parameter `logon path` festgelegt. Der Standardpfad steht auf `logon path = \\%N\%U\profile` . Damit wird im Heimatverzeichnis des Benutzers auf dem PDC ein Verzeichnis namens `profile` angelegt und das Profil dort gespeichert. Leider kann man mit Samba nicht sauber für einzelne Benutzer festlegen, daß sie ihre Profile auf einem Server ablegen, andere Benutzer ihre Profile aber lokal auf den Workstations belassen.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

24.1 Anmeldeskripte

Meldet sich ein Benutzer an einer Domäne an, kann der Primary Domain Controller der Workstation mitteilen, daß unter den Rechten des Benutzers eine Batchdatei automatisch ausgeführt werden soll, das so genannte *Logon Script*. Samba kennt den Parameter `logon script`, mit dem der Name des Logon Schriptes festgelegt wird. Standarmäßig gibt es kein Logon Script. Wird eines festgelegt, bezieht es sich immer auf eine `.bat`-Datei in der festgelegten Freigabe `[netlogon]`. Eine vollständige `smb.conf` für einen PDC sähe so aus:

```
[global]
workgroup = samba
encrypt passwords = yes
domain master = yes
domain logons = yes

logon script = logon.bat

[homes]
writeable = yes
valid users = %S

 [netlogon]
path = /data/netlogon
```

Ein einfaches Logon Script in `/data/netlogon/logon.bat` kann so aussehen:

```
@echo off^M
net use h: \\pdc\homes^M
```

Die `^M`-Zeichen am Zeilenende bezeichnen die DOS-Zeilenendekonvention, bei der an jedem Zeilenende zuerst ein Carriage Return und dann ein Linefeed kommt. Unix kennt nur den Linefeed als Zeilenende. Der Carriage Return ist hier entscheidend, da ansonsten Windows diese Batchdatei nicht ausführen wird. Wenn ein Logon Script unter Unix editiert wird, bekommt man den Carriage Return im Editor normalerweise durch die Kombination Control-V Control-M. Moderne Editoren wie der vim oder der Emacs erkennen eine existierende Datei mit DOS-Zeilenendekonvention automatisch und speichern sie auch entsprechend wieder ab.

25 Samba als Domänenmitglied

Wenn man mehr als einen Sambaserver betreibt oder einen echten Windows-Server betreibt, benötigt man genau so wie mit einer echten Windows-Domäne eine zentrale Benutzerverwaltung.

Die zentrale Verwaltung der Paßwörter ist ein erster Schritt. Um dies zu erreichen, muß man mit Samba eine Windows NT-Domäne betreten. Dazu setzt man in der `smb.conf` folgende Parameter:

```
[global]
  workgroup = windows
  security = domain
  password server = *
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
encrypt passwords = yes
name resolve order = wins bcast
```

Im Kapitel 14 wurde beschrieben, wie eine SMB-Sitzung aufgebaut wird. Dort wurde auf den Unterschied zwischen `security = share` und `security = user` eingegangen. `security = domain` verhält sich aus der Sicht eines Clients genau wie `security = user`, es wird vom Benutzer im Session Setup ein Benutzername, eine Domäne und ein Kennwort verlangt. Ist das Kennwort nicht korrekt, so wird der Benutzer zurückgewiesen. Der Parameter `security = domain` bewirkt nun, daß das Paßwort nicht wie bei `security = user` in der lokalen `smbpasswd` nachgesehen wird, sondern an einen PDC weitergeleitet wird. Dieser entscheidet dann, ob das Paßwort korrekt ist oder nicht. Bestätigt der PDC das Paßwort, akzeptiert Samba den Benutzer. Kann der PDC die Benutzeridentität nicht bestätigen, macht Samba einen zweiten Versuch anhand der lokalen `smbpasswd`. Damit kann man es erreichen, daß für Administratoren der Zugriff auf den Sambaserver noch möglich ist, falls einmal kein Domänencontroller verfügbar sein sollte.

Zusätzlich zu `security = domain` gibt es noch `security = server`. Diese Einstellung ist jedoch nicht mehr zu empfehlen, dazu mehr am Ende des Kapitels.

Für den Parameter `password server` gibt es zwei Möglichkeiten. Entweder man setzt ihn auf * wie im Beispiel geschehen. Dann sucht sich Samba mit NT-konformen Mitteln selbst den PDC oder einen BDC, um Benutzer zu authentifizieren. Man kann aber auch eine Liste von NetBIOS-Namen angeben, mit denen Samba arbeiten soll. In beiden Fällen ist es wichtig, daß die Namensauflösung einwandfrei funktioniert. Samba muß in der Lage sein, einen Domänencontroller für die Authentifizierung zu finden. Dies ist eine der wenigen Stellen, bei denen Samba als NetBIOS-Client arbeitet. Daher ist es hier möglicherweise nötig, die `name resolve order` korrekt zu setzen. Insbesondere ist dies wichtig, wenn die Namen der PDCs im DNS bereits vergeben sind und vielleicht auf andere Maschinen zeigen als die entsprechenden NetBIOS-Namen.

Um für bestimmte Benutzer nicht auf den PDC angewiesen zu sein, versucht Samba bei einem erfolglosen Versuch der Domänenanmeldung zusätzlich, den Benutzer in der `smbpasswd` zu finden. Damit kann der Server mit möglicherweise nicht aktuellen Paßwörtern funktionsfähig gehalten werden, auch wenn der PDC einmal ausfallen sollte.

Samba muß, um Paßwörter an den PDC weiterzuleiten, genau wie eine NT-Workstation ein Computerkonto auf dem PDC besitzen und die Domäne betreten. Das Computerkonto kann auf dem PDC mit dem Servermanager oder mit dem Kommando

```
net computer <name> /add
```

auf der Kommandozeile erledigt werden. Danach kann Samba mit dem Aufruf

```
smbclient -j DOMAIN -r PDCNAME
```

die Domäne betreten. Seit Samba 2.2.2 ist es zusätzlich möglich, das Computerkonto wie von einer NT Workstation aus beim Betreten der Domäne automatisch erstellen zu lassen. Dies geschieht, indem man dem Aufruf von `smbpasswd -j` noch einen berechtigten Benutzer mitgibt:

```
smbclient -j DOMAIN -r PDCNAME -U Administrator
```

`smbclient` erfragt das Paßwort des Domänenadministrators. Nach Eingabe des Paßwortes wird das Computerkonto auf dem PDC erstellt und das entsprechende Paßwort korrekt gesetzt.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Ist Samba Domänenclient, so wird das Paßwort zum Computerkonto in der Datei `secrets.tdb` abgespeichert. Diese übernimmt damit die Aufgabe der Datei `DOMAIN.MACHINE.mac`, die es bis Samba 2.0 gab. Geht diese Datei verloren, muß die Domäne neu betreten werden. Dies kann durch Entfernen und wieder Hinzufügen zur Domäne mit dem Servermanager und nachfolgendes `smbpasswd -j` oder durch ein automatisches Erstellen des Computerkontos geschehen.

Mit der Domänenmitgliedschaft wird der Sambaserver nur von der Paßwortverwaltung befreit. Um die Benutzer und Gruppen muß er sich weiterhin selbst kümmern. Eine gewisse Erleichterung kann dabei das `add user script` bringen, das bei Samba als PDC dafür gesorgt hat, die Computerkonten in der `/etc/passwd` automatisch zu erstellen. Ist Samba Domänenmitglied, so wird bei einer Benutzeranfrage auf den Server zunächst der PDC nach der Richtigkeit des Paßwortes befragt. Bestätigt dieser das Paßwort und will der Benutzer dann auf das Dateisystem des Sambaservers zugreifen, so wird eine Unix UID benötigt, Samba schaut in die `/etc/passwd`. Findet Samba dort trotz erfolgreicher Anmeldung am PDC keinen Benutzer, so wird das `add user script` mit entsprechenden Argumenten aufgerufen, um den Benutzer zu erstellen.

Damit muß man sich teilweise nicht mehr um die Verwaltung der Benutzer auf dem Sambaserver kümmern. Teilweise deswegen, da von dem neu anzulegenden Benutzer ausschließlich der Name bekannt ist. Es fehlt jegliche Information darüber, in welchen Gruppen sich der Benutzer in der Domäne befindet. Diese Einschränkung macht eine Rechteverwaltung auf dem Sambaserver sehr schwierig bis unmöglich.

Unter Unix gibt es mehrere Möglichkeiten, über Rechnergrenzen hinweg die Benutzerdatenbanken zu synchronisieren. Das reicht vom unsicheren NIS bis hin zur skriptgesteuerten Verteilung der Dateien `/etc/passwd` und `/etc/group` über rsync und ssh. Setzt man für die Datei- und Druckdienste komplett auf Unix mit Samba, kann man so eine zentrale Verwaltung der Server erreichen. Die `smbpasswd` muß dabei in die Verteilung der Benutzerdatenbanken nicht mit einbezogen werden, da hierfür eine Domäne aufgebaut werden kann. Einer der Server wird zum Domänencontroller erklärt, die anderen Server sind ganz normale Mitgliedsserver, die die Paßwörter vom Domänencontroller überprüfen lassen.

25.1 Hintergrund: `security = server`

Vor Samba 2.0 gab es für die zentrale Verwaltung von Paßwörtern nur die Möglichkeit, `security = server` zu setzen. Damit konnte ein Sambaserver sehr einfach die Anmeldung von einem weiteren Server oder einer NT Workstation beziehen. Samba 2.0 und 2.2 beherrschen diese Möglichkeit immer noch, man sollte jedoch strikt von ihrer Nutzung abraten, da sie erhebliche Probleme mit sich bringt. `security = server` nutzt nicht das Domänencontrollerprotokoll, sondern leitet den Benutzernamen und das Paßwort an einen Server weiter. Dies ist im Prinzip nicht schlecht, birgt aber ein subtiles Problem. Setzt man keine verschlüsselten Paßwörter ein, verschicken viele Clients die Paßwörter in Großbuchstaben. Verlangt der Paßwortserver nun verschlüsselte Paßwörter, muß Samba raten. Dies kostet Last und Zeit. Setzt man auf dem Sambaserver verschlüsselte Paßwörter ein, handelt man sich ein noch subtileres Problem ein. Um das zu verstehen, sollte man sich das Kapitel 20 auf jeden Fall genau angesehen haben.

In der Antwort zur Anfrage Negotiate Protocol liefert der Server dem Client eine Herausforderung. Im Session Setup muß der Client die mit dem Paßwort verschlüsselte Herausforderung liefern. Will Samba dies nun gegenüber einem Paßwortserver machen, so muß er zunächst einen Negotiate Protocol absetzen, um vom Paßwortserver eine Herausforderung zu erhalten. Diese Herausforderung liefert er seinem Client direkt weiter, damit dieser sie dann mit dem Paßwort verschlüsseln kann. Da es pro Verbindung vom Client zum Server nur einen Negotiate Protocol Request gibt, gilt die Herausforderung für die gesamte Verbindung. Viele Clients setzen aber mehrere Session Setups über die gleiche Verbin-

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

dung ab. Damit der Sambaserver zwischen Client und Paßwortserver immer mit der gleichen Herausforderung arbeiten kann (der Client sieht nur diese eine Herausforderung), muß er zum Paßwortserver ständig eine Verbindung offen halten. Bräche diese Verbindung ab, bekäme der Sambaserver vom Paßwortserver eine neue Herausforderung mitgeteilt. Der Sambaserver hat leider keine Möglichkeit, den Client dazu zu zwingen, eine neue Herausforderung zu verlangen. Die einzige Möglichkeit ist, die Verbindung zum Client abzubrechen, um einen neuen Negotiate Protocol zu verlangen. Damit gibt es zwei Probleme:

- Pro Clientsystem muß der Sambaserver ständig eine Verbindung zum Paßwortserver offenhalten. Damit werden auf dem Paßwortserver erhebliche Resourcen gebunden.
- Das Netz wird außerordentlich instabil, sollte sich der Paßwortserver entscheiden, diese vielen nicht besonders aktiven Verbindungen abzubrechen. Clients werden sich am Sambaserver erneut anmelden müssen.

Das Domänencontrollerprotokoll löst diese beiden Probleme, indem es dem Sambaserver erlaubt, sich eine eigene Herausforderung pro Client auszudenken und diese bei der Netzwerkanmeldung beim PDC mitzuschicken. Um kein Sicherheitsproblem aufkommen zu lassen, muß diese Netzwerkanmeldung vom Sambserver zum PDC verschlüsselt sein, daher das Computerkonto, dessen Paßwort als Schlüssel für die symmetrische Verschlüsselung zwischen Sambaserver und PDC verwendet wird.

26 winbind

Wenn man Samba als Domänenmitglied betreibt, hat man die größte Hürde zu einer problemlosen Integration bereits genommen: Die Paßwortverwaltung. Jeder Benutzer kann sein Paßwort in der Domäne ändern, und die Änderung wird sofort auf allen Domänenmitgliedern sichtbar. Ein Problem bleibt jedoch bestehen. Man muß auf den Sambaservern, die Domänenmiglieder sind, die Benutzer nachpflegen. Wird ein neuer Benutzer in der Domäne angelegt, oder werden Gruppenmitgliedschaften geändert, muß dies manuell auf den Sambaservern eingetragen werden. Ist auch der Primary Domain Cotroller ein Sambaserver, kann man sich mit dem Network Information System NIS behelfen, aber wenn die Benutzerdatenbank auf einem echten NT-Server liegt, ist dieser Weg verschlossen.

Eine wirklich zwischen Windows NT und Unix vereinheitlichte Benutzerdatenbank bietet seit Samba 2.2.2 der Dämon *winbind*. Er ermöglicht die volle Einbindung eines Unixsystems in eine NT-Domäne. Voraussetzung dafür ist die Unterstützung der `nsswitch`-Module. Momentan bieten dies Linux und Solaris. Die anderen von Samba unterstützten Unixsysteme bleiben außen vor.

26.1 nsswitch-Module

Viele Programme unter Unix müssen auf Datenbanken im Verzeichnis `/etc` zugreifen. Beispielsweise muß `ls -l` den Dateibesitzer, der in der Datei nur in numerischer Form vorliegt, in einen Benutzernamen übersetzen. Dazu muß die numerische User-ID in der Datei `/etc/passwd` gesucht werden. Daß diese Übersetzung tatsächlich stattfindet, kann man leicht folgendermaßen überprüfen. Man muß nur testweise die Leserechte für **others** von der Datei `/etc/passwd` mit `chmod o-r /etc/paswd` wegnehmen und `ls -l` aufrufen. Die Dateibesitzer werden nur noch numerisch angezeigt[17]

[17] Sollte dies nicht spontan funktionieren, kann der `nscd` schuld sein. Siehe hierzu Seite 69.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Sauber geschriebene Programme greifen nicht direkt auf die Dateien in /etc zu, sondern durch Bibliotheksaufrufe wie beispielsweise getpwuid(2). Seit der glibc-Version 2 werden diese Bibliotheksaufrufe in dynamisch geladenen Modulen implementiert. Gesteuert werden diese Module über die Datei /etc/nsswitch.conf. Für jede der Dateien in /etc, die von allgemeinem Interesse ist, gibt es eine Zeile in der /etc/nsswitch.conf. Beispielsweise wird der Zugriff auf die Datei /etc/passwd über die Zeile

```
passwd: compat
```

oder

```
passwd: files nis
```

gesteuert. Durch die erste Version wird ein Kompatibilitätsmodul zum Zugriff herangezogen, die zweite Variante legt explizit fest, daß zuerst in der lokalen Datei /etc/passwd nach Benutzern gesucht werden soll. Schlägt dies fehl, wird das NIS befragt.

Wie funktioniert diese Steuerung? Die Option compat bewirkt, daß zur Laufzeit eines Programms die dynamische Bibliothek /lib/libnss_**compat**.so.2 geladen wird und die Anfrage beantworten muß. files und nis beziehen sich entsprechend auf die Dateien /lib/libnss_**files**.so.2 und /lib/libnss_**nis**.so.2.

winbind besteht aus zwei Teilen: Einer Datei libnss_winbind.so und einem Dämon winbind. In den Samba-Quellen findet sich der winbind unter source/nsswitch. Dort wird auch die Datei libnss_winbind.so abgelegt. Zur Installation muß sie manuell nach /lib kopiert werden:

```
cp source/nsswitch/libnss_winbind.so /lib/libnss_winbind.so.2
```

Der winbindd selbst wird automatisch mit im sbin-Unterverzeichnis von Samba installiert.

26.2 Konfiguration von Winbind

Um Winbind zu aktivieren, müssen in der Datei /etc/nsswitch.conf die beiden Zeilen für passwd und group durch die Angabe winbind ergänzt werden, etwa so:

```
# /etc/nsswitch.conf
passwd: files winbind
group: files winbind
```

Damit werden Benutzer und Gruppen zuerst in den lokalen Dateien gesucht. Danach wird der winbindd befragt, der im Hintergrund laufen muß.

Für die Konfiguration des winbindd muß Samba ein normales Domänenmitglied sein. Siehe hierzu Kapitel 25. Der winbindd benötigt in der /etc/smb.conf einige zusätzliche Parameter. Eine vollständige Beispielkonfiguration ist die folgende:

```
; Samba als Domaenenmitglied
workgroup = windows
security = domain
password server = *
encrypt passwords = yes
```

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

```
; Winbind-Konfiguration
winbind separator = +
winbind uid = 10000-20000
winbind gid = 10000-20000
template shell = /bin/bash
template homedir = /home/%D/%u
```

Die Parameter bedeuten im einzelnen:

winbind separator: Unter Windows wird ein vollständiger Benutzername mit Domäne in der Form **DOMAENE\benutzername** angegeben. Unter Unix hat dies Nachteile, da der Backslash \ für die Shell eine Sonderbedeutung hat. Daher kann man den Trenner zwischen Domäne und Benutzername separat konfigurieren. Als unkritisch erweist sich das +-Zeichen. Ein Benutzer wird somit als **DOMAENE+benutzername** angegeben.

winbind uid: Der `winbindd` muß dynamisch für Domänenbenutzer numerische User-IDs vergeben. Um dies tun zu können, wird ihm mit dem Parameter `winbind uid` eine Menge von User-IDs übergeben, die nicht in der `/etc/passwd` oder im NIS verwendet werden. Wird auf einen Benutzernamen das erste Mal zugegriffen, wählt der `winbindd` für diesen Benutzer aus seinem Pool die nächste freie User-ID aus und speichert diese Zuordnung fest in der Datei `winbindd_idmap.tdb`.

winbind gid: Für Group-IDs gilt das gleiche wie für User-IDs.

template shell: Der Primary Domain Controller kennt das Konzept der Login-Shell nicht. Diese muß zentral für alle Winbind-Benutzer in der `smb.conf` vergeben werden.

template homedir: Auch ein Heimatverzeichnis wird in der SAM von Windows nicht abgespeichert und muß in der `smb.conf` vorgegeben werden. Hierbei sollte man auf jeden Fall die Domäne des Benutzers in den Pfad mit aufnehmen, um Namenskollisionen bei Vertrauensstellungen zu behandeln. Der Benutzer **schmidt** in der Domäne **GOE** sollte ein anderes Heimatverzeichnis bekommen als der Benutzer **schmidt** in der Domäne **HD**. Die Heimatverzeichnisse werden selbstverständlich nicht automatisch erzeugt, sondern müssen manuell angelegt und den Benutzern übergeben werden. Auf einem reinen Fileserver mit gemeinsamen Dateien ist es jedoch möglicherweise nicht notwendig, für jeden Benutzer eigene Heimatverzeichnisse anzulegen.

Mit diesen Einstellungen kann man den `winbindd` zusätzlich zu `smbd` und `nmbd` starten, die ebenfalls laufen müssen.

26.3 Winbindd abfragen: wbinfo

Laufen `winbindd`, `smbd` und `nmbd` in der Domäne, kann man das ganze testen. Das Utility zum Testen heißt `wbinfo`. Der wichtigste Test ist der Aufruf `wbinfo -t`. Damit wird die Verbindung zum Domänencontroller geprüft, `winbindd` sucht den PDC und meldet sich mit dem Workstationkonto an. Das Programm `wbinfo` muß die Ausgabe `Secret is good` geben. Gibt `wbinfo` diese Ausgabe, so ist der `winbindd` gültiges Domänenmitglied und kann Informationen vom PDC abrufen.

`wbinfo` kennt noch eine Reihe weiterer Parameter, mit denen die Benutzerdatenbank der Domäne abgefragt werden kann. Die Ausgabe des Aufrufs `wbinfo` ohne Parameter gibt einen Hilfetext mit den verfügbaren Optionen aus.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

Schlägt `wbinfo -t` fehl, so muß die Domänenmitgliedschaft überprüft werden. Hierzu siehe Kapitel 25.

Verläuft der Test mit `wbinfo -t` erfolgreich, so kann man sich sämtliche verfügbaren Benutzer mit `getent passwd` und die Gruppen mit `getent group` auflisten lassen.

26.4 nscd

Unter Linux ist der Name Service Caching Daemon `nscd` ist ein Programm, mit dem sämtliche Abfragen durch den nsswitch-Mechanismus gecached werden können. Der `nscd` macht Sinn, wenn diese Anfragen sehr lange dauern. Dies kann der Fall sein, wenn die Dateien sehr groß sind, etwa wenn hunderte von Usern im System angelegt sind. Ein anderer Verzögerungsgrund ist die Abfrage von Benutzerdaten über ein Netzwerk beim Einsatz von NIS.

Ein Nachteil des `nscd` kann sein, daß er Änderungen in der Benutzerdatenbank nicht schnell genug mitbekommt. Insbesondere beim Testen des `winbind` kann dies sehr hinderlich sein. Wer beispielsweise folgendes schon einmal erlebt hat, hat ein Problem mit dem `nscd`:

```
delphin:~ # useradd -m vl
delphin:~ # passwd vl
passwd: Unknown user vl
delphin:~ #
```

In diesem Falle sollte man den `nscd` schleunigst killen und dafür sorgen, daß er beim nächsten booten nicht wiederkommt.

27 smbcontrol

Bis zur Version 2.0 hatte man relativ wenig Möglichkeiten, in das laufende Samba einzugreifen. Man konnte mit dem Signal `USR1` den Debuglevel um einen Punkt erhöhen und mit `USR2` um einen Punkt erniedrigen. Darüber hinaus blieb häufig nur die Möglichkeit, einzelne Sambaprozesse oder sogar das ganze Samba herunterzufahren, wenn man Konfigurationsänderungen vorgenommen hatte. Mit Samba 2.2 gibt es für diese Anwendungen ein neues Werkzeug: `smbcontrol`. `smbcontrol` bietet die Möglichkeit, einzelne Dinge anzustoßen. `smbcontrol` verschickt hierzu Nachrichten an einzelne Sambaprozesse, oder an alle `smbd`s.

Man kann jetzt im Gegensatz zu Samba 2.0 den Debuglevel einzelner Prozesse direkt setzen. Dies geschieht wie in folgendem Beispiel:

```
root@server:~ > smbcontrol smbd debuglevel
Current debug level of PID 4423 is 0
Current debug level of PID 17392 is 0
Current debug level of PID 22272 is 0
root@server:~ > smbcontrol 17392 debug 1
root@server:~ > smbcontrol smbd debuglevel
Current debug level of PID 4423 is 0
Current debug level of PID 17392 is 1
Current debug level of PID 22272 is 0
```

An diesem Beispiel ist deutlich, wie `smbcontrol` zu benutzen ist. Als ersten Parameter verlangt `smbcontrol` das Ziel der Nachricht, die es verschicken soll. Zweiter Parameter ist die Nachricht, die

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)

verschickt werden soll. Daran schließen sich dann weitere Parameter an, die möglicherweise zu der Nachricht gehören.

Die Nachrichten im Einzelnen:

debug: Mit dieser Nachricht wird der Debuglevel anhand des weiteren Parameters gesetzt.

debuglevel: `smbcontrol` liest hiermit den aktuellen Debuglevel von Prozessen aus.

force-election: Mit dieser Nachricht wird eine Wahl zum Local Master Browser erzwungen. Diese Nachricht kann nur an den `nmbd` geschickt werden. Der `smbd` hat mit der Wahl nichts zu tun.

ping: Mit dieser Nachricht können Prozesse einfach zum Antworten bewegt werden. **ping** erwartet einen Parameter, der die Anzahl der Pings zum Ziel festlegt.

profile: Diese Nachricht ist für Entwickler gedacht. Um das Profiling zu nutzen, muß Samba mit der `configure`-Option `--with-profiling-data` compiliert werden. Dann kann mit dieser Nachricht der interne Profiling-Code gesteuert werden. Damit können Entwickler die Teile des Codes bestimmen, in denen am meisten Zeit verbraucht wird.

profilelevel: Diese Nachricht ist ebenfalls für Entwickler gedacht.

close-share: Der `smbd` kann mit dieser Nachricht dazu bewegt werden, alle Verbindungen zu einer bestimmten Freigabe zu beenden, ohne die restlichen Verbindungen zu stören. Dies kann insbesondere dann sinnvoll sein, wenn Änderungen an den Zugriffsrechten einer Freigabe vorgenommen wurden.

printer-notify: Wenn Sie von Windows NT Clients aus mit Druckern verbunden sind, nutzen Sie möglicherweise das neue Drucksystem von Samba. In diesem Fall sind Druckereinstellungen auf dem Server gespeichert. Ändern sich diese Einstellungen, können Sie mit dieser Nachricht die momentan verbundenen Clients über diese Änderungen informieren.

© 1999, 2000, 2001, Volker Lendecke (http://www.sernet.de/vl/)