Kursskript

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

4. August 2000

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

Samba – Was ist das?

Kurz gesagt läßt Samba jeden Unixrechner in der Netzwerkumgebung von Windows NT erscheinen. Außerdem lassen sich mit Samba Datei- und Druckfreigaben erstellen. Das heißt, unter Unix vorhandener Plattenplatz kann ganz normal unter Windows genutzt werden, und unter Unix vorhandene Drucker kann man als Netzwerkdrucker unter Windows ansteuern. 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.

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 Primary Domain Controller ist in einer noch nicht freigegebenen Version implementiert, ist jedoch schon in vielen Installationen im produktiven Einsatz.

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 den bekannten 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.

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, dann wird nur dieser Client in Mitleidenschaft gezogen. Eventuell stürzt dieser eine Prozeß ab, die anderen werden jedoch 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 optimal 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. Durch ein sehr flexibles Schema von Makroersetzungen ist mit Samba sehr viel mehr möglich als mit NT. 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.

2 NetBIOS

Sobald Windowsrechner Dateisysteme austauschen, sich gegenseitig in der Netzwerkumgebung sehen oder Drucker freigeben, funktioniert die Kommunikation über NetBIOS[1]. NetBIOS ist eine 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. Dabei lag der Schwerpunkt des Entwurfs auf der Einfachheit der Anwendung. Auf Skalierbarkeit und die Andwendung in Weitverkehrsnetzen wurde beim Design nicht geachtet.

Die Kommunikation mit NetBIOS wurde in drei Teilbereiche aufgeteilt, den Namensdienst, 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

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

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. Aufeinander folgende Pakete können in vertauschter Reihenfolge beim Empfänger ankommen. Es kann sogar sein, daß Pakete auf dem Weg dupliziert werden, also mehrfach ankommen. Diese Unzuverlässigkeit des Netzes wird durch den Datagrammdienst an die Benutzerprogramme weitergegeben. 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.

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.

Zwei Anwendungen wollen nun per NetBIOS miteinander kommunizieren. Dazu muß zunächst der Server seine Bereitschaft kundtun, Verbindungen entgegenzunehmen. Dazu meldet er sich im Netz mit seinem 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[2]. Eine Reservierung geschieht also, indem ein Rechner per Broadcast ankündigt, daß er unter einem bestimmten Namen erreichbar ist. Dann wartet er auf Protest. Beklagt sich niemand, schickt er eine zweite Reservierung und wartet wieder. Nach der dritten Reservierung 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.

NetBIOS kann mit unterschiedlichen Protokollen implementiert werden. NetBEUI, IPX und

[2] Mit dieser Freiheit ergeben sich viele Möglicheiten, von einem beliebigen Rechner aus ein Windows-Netz bis zur Unbenutzbarkeit zu stören. Man muß nur bestimmte, für den PDC vorgesehene Namen zum richtigen Zeitpunkt reservieren.

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 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.

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.

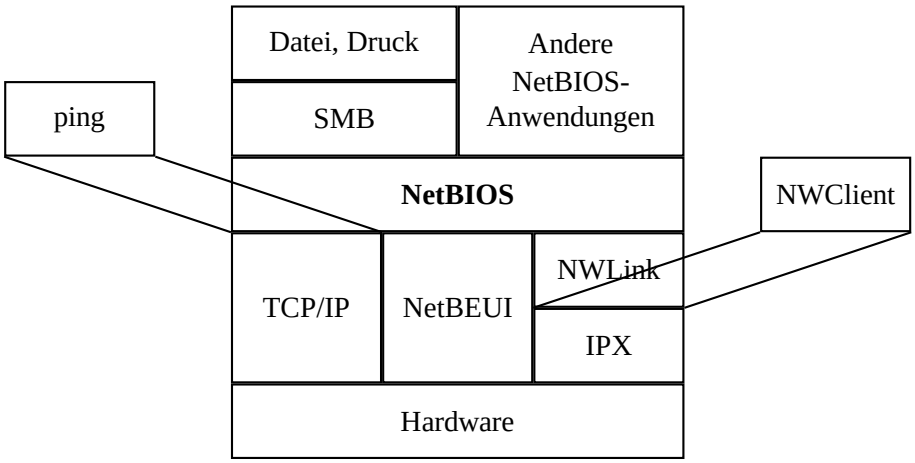

Abbildung 1: Protokollstapel

Die Protokolle ordnen sich folgendermaßen ein:

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.

3 Bestandteile von Samba

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

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 lauscht auf dem TCP-Port 139, und nimmt neue Verbindungen entgegen. Jede neue Verbindung stößt einen neuen Prozeß `smbd` 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.

Samba ist im Hauptspeicherverbrauch recht sparsam. Jeder *aktive* Client benötigt etwa 1 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 ist WINS-Server und 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 14 erklärt.

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 wird mit `smbclient` die Liste der Server im Netz erfragt, 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 Seiten von Windows ist das Kommandozeilenprogramm `nbtstat`.

Auf der Serverseite finden sich noch weitere Komponenten:

smb.conf: Die zentrale Konfigurationsdatei von Samba. Ist Samba als fester Systembestandteil installiert, findet sie sich in der Regel unter `/etc/smb.conf`. Ist Samba selbst compiliert, 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 compiliert, liegt dies Verzeichnis unter `/usr/local/samba/var`.

/etc/smbpasswd ist die Paßwortdatenbank von Samba, sofern mit verschlüsselten Paßwörtern gearbeitet wird. `/usr/local/samba/private/` ist das Standardverzeichnis für diese Datei.

4 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. `ifconfig <interface>` unter Unix, oder unter Linux speziell `ifconfig eth0` gibt sowohl die IP-Adresse der Ethernetkarte als auch die dort gesetzte Broadcastadresse als Ausgabe. 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 man 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.

Mit diesen beiden Einstellungen wird man direkt den Sambarechner in der Netzwerkumgebung sehen. Zur Vereinfachung sollten noch zwei weitere Parameter gesetzt werden, die später erklärt werden. Die vollständige `smb.conf` sieht also folgendermaßen aus:[3]

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

Mit dieser Konfiguration kann Samba gestartet werden. Unter SuSE Linux geschieht dies 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 muß letztlich nur erreicht werden, daß sowohl der `nmbd` als auch der `smbd` mit dem Parameter `-D` gestartet 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.

[3] Auf einem der Rechner sollte zusätzlich `os level = 67` und `preferred master = yes` im Abschnitt `[global]` der /etc/smb.conf gesetzt sein.

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

5 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, 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@server:/home/vlendec> nmblookup -B 192.168.1.31 server
querying server on 192.168.1.31
name_query failed to find name server
vlendec@linux:/home/vlendec>
```

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` 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 `-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.

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 Einzelnamen 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.

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.

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 sogenannte *Lokale 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<1e> Wahlen zum Lokalen Master Browser werden über diesen Namen abgewickelt. Siehe hierzu Kapitel 6.

6 Netzwerkumgebung

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 funktionieren, indem jeder Rechner, der Serverdienste anbietet, dieses regelmäßig per Broadcast im Netz mitteilt. 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 *Lokale Master Browser* (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 Lokalen Master Browser gibt. Beispielsweise kann der Explorer von Windows eine solche Wahl anstoßen. Wenn Windows 95 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.

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 Windows 95 Rechner, so hält sie sich selbst für geeigneter, den Lokalen 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, 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 Lokalen Master Browser.

Drei Parameter in der `smb.conf` bestimmen das Verhalten von Samba in der Wahl zum Lokalen 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
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 Lokalen Master Browser Fehler bereinigt wurden. Dann ist es sinnvoll, daß diese neue Software die Rolle

des LMB übernimmt. Der einfachste Weg ist, den `os level` einfach hochzusetzen. Samba hat hier einen Vorgabewert von 20.

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. Wenn man sichergehen möchte, daß Samba auf jeden Fall nach dem Start den LMB übernimmt, dann muß man den `os level` hoch genug setzen, und den Parameter `preferred master = yes` setzen. Damit wird Samba beim Start des `nmbd` auf jeden Fall eine Wahl anstoßen und sie dann unter Umständen gewinnen.

Mit den Einstellungen

```
[global]
os level = 66
preferred master = yes
```

kann man sicher sein, daß der Sambarechner immer den LMB innehat. Es sei denn, ein anderer Administrator von Samba kommt auf die Idee, einen noch höheren Wert für den `os level` zu benutzen.

Ein Primary Domain Controller kann unter Umständen erheblich gestört werden, wenn er in seinem Subnetz nicht der LMB ist.

7 NetBIOS über Subnetzgrenzen

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.

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.

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

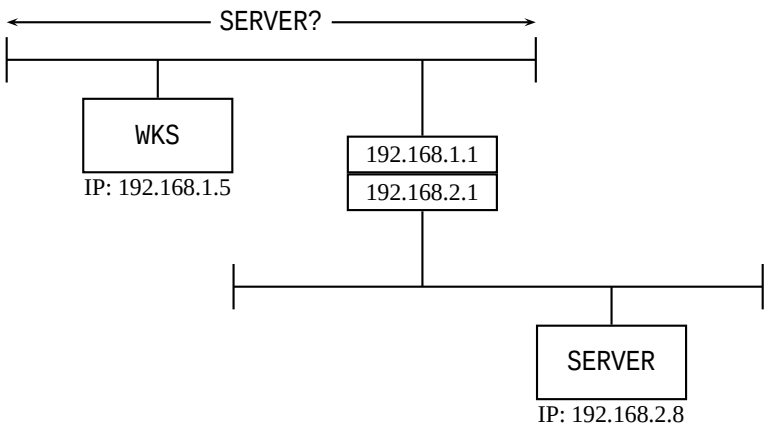

Abbildung 2: Namensanfrage per Broadcast

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 zweite Möglichkeit, das Problem zu lösen, ist eine zentrale Datei `LMHOSTS`. Dazu gibt es den WINS-Server. Ein solcher Server ist ein Rechner, bei dem sich jede 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.

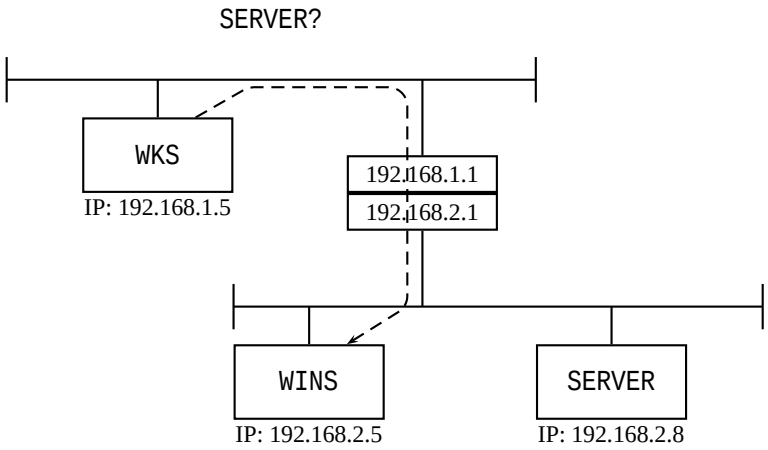

Abbildung 3: WINS-Anfrage

Samba kann ganz normal als WINS-Server konfiguriert werden, indem der Parameter `wins support = yes` gesetzt wird. Ist diese 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.

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. Namens-

reservierung per Broadcast erfolgt durch 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. Rechner im einen Namensraum können mit Rechnern, die an einem anderen WINS-Server angeschlossen sind, nicht kommunizieren. Es kann trotzdem zu Kollisionen kommen, da Windowsrechner bestimmte Namen unabhängig von WINS-Einstellungen ausschließlich per Broadcast reservieren. Unter Windows NT kann man mehrere WINS-Server einsetzen, die sich gegenseitig abstimmen. Diese WINS-Server treten gegenüber den Clients als ein einziger Server auf, unabhängig von ihrer Anzahl.

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. Damit kann man einen Samba-Server in ein Subnetz stellen. 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.

8 Netzwerkumgebung über Subnetzgrenzen

So, wie die Netzwerkumgebung in Abschnitt 6 betrachtet wurde, funktioniert sie nur in einem einzigen lokalen Netz. Die Wahl zum lokalen 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.

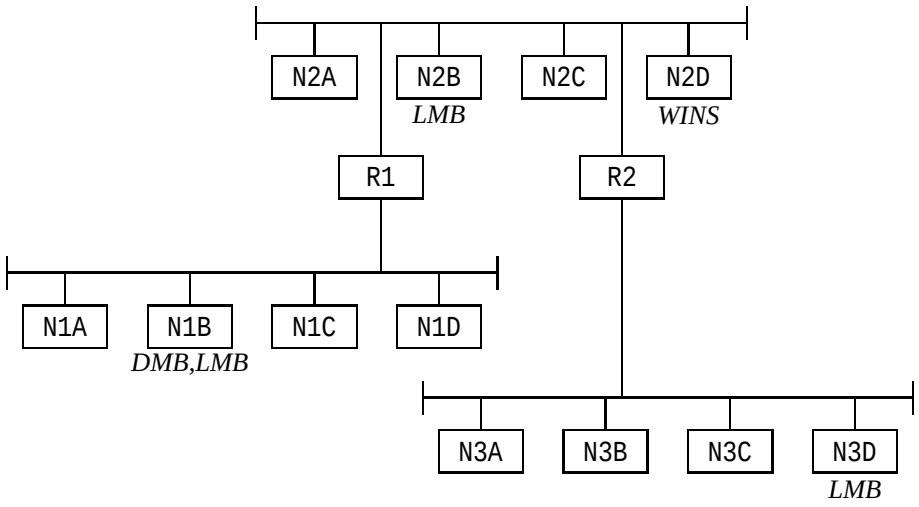

Abbildung 4: Domain Master Browser

Dieses Netz besteht aus 3 Subnetzen (1,2,3), die durch 2 Router (R1 und R2) verbunden sind. Die Router lassen keine Broadcasts durch. Alle Subnetze bestehen aus jeweils 4 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 Browseliste 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 N1C, 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 Subnetz 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 Browseliste. 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 2 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.

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 Lokalen 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.

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.

9 Einfache Freigaben

Der grundsätzliche Aufbau der Datei `smb.conf` wurde bereits auf Seite (9) erwähnt. Bis zu diesem Punkt hat sich sämtliche Konfiguration im Abschnitt `[global]` abgespielt, der globale Servereinstellungen beinhaltet. Wenn der Sambarechner nicht nur im Netz gesehen werden soll, sondern auch sinnvolle Dinge tun soll, muß man Freigaben zur Verfügung stellen. Dies tut man, indem man einfach einen neuen Abschnitt beginnt, dessen Name gerade nicht `[global]` ist. Um eine Freigabe vollständig zu machen, muß man mit dem Parameter `path` angeben, welches Verzeichnis man freigeben möchte. Eine für alle Zugriffe offene Freigabe des Verzeichnisses `/cdrom` erreicht man mit folgender `smb.conf`:

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

```
security = share
encrypt passwords = yes

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

Damit entsteht auf dem Server eine Freigabe namens `CD`, die das Verzeichnis `/cdrom` im Netz für alle zum Lesen zur Verfügung stellt.

Achtung: Es findet hier *keine* Überprüfung der Zugriffsrechte statt. Um diese Überprüfung zu ermöglichen, sollte zunächst einmal der Aufbau einer Verbindung zu einer Freigabe genauer beleuchtet werden.

10 SMB-Sitzungen

Wird am Client eine Verbindung zu einer Freigabe auf einem SMB-Server aufgebaut, so müssen mehrere Schritte durchlaufen werden.

NetBIOS-Namensauflösung

Zu einem Rechnernamen muß eine IP-Adresse herausgefunden werden. Dies wurde in den letzten Abschnitten eingehend behandelt.

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`.

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`[4] 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 sie `/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
```

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.

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,

[4] Das 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, den CIFS von seinem Urvater SMB unterscheidet. Mit Windows 2000 werden diese NetBIOS-Namen beim Verbindungsaufbau gar komplett unterschlagen. Dafür ist es aber notwendig, einen weiteren Port zu reservieren, und zwar Port 445.

bei dem im Prinzip der Client jede Aktion durch eine Anfrage anstößt, und der Server diese beantwortet[5].

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.

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 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

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

den Benutzernamen, der sich anmelden will, kennt. Es ist also nicht möglich, für einige Benutzer Klartextpaßwörter und für andere Benutzer verschlüsselte Paßwörter zu verwenden.

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.

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.

11 Zugriffsrechte

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 bezüglich der Zugriffsrechte bei einer Freigabe nichts gesagt wird, 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:

  ```
  [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
writeable = no
write list = @admins
```

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

12 Unix-Zugriffsrechte

Unter Windows NT gibt es zwei Möglichkeiten, Zugriff auf Dateien zu gewähren. Über eine Freigabe kann ein Lese- oder ein Schreibrecht vergeben werden. Im zweiten Schritt können dann über eine Rechtevergabe im Dateisystem weitere Rechte vergeben werden. Samba regelt die Zugriffskontrolle ebenfalls in zwei Schritten. Die freigabebezogenen Rechte werden über Parameter wie `valid users` und `write ok` geregelt. Die Zugriffsrechte innerhalb des Dateisystems regelt Samba nicht selbst, sondern verläßt sich hierfür auf das darunterliegende Betriebssystem Unix.

Zwischen Unix und DOS bestehen große Unterschiede. DOS und alle seine Nachfolger sind Einzelbenutzersysteme, Unix ist von Anfang an als Multiusersystem entworfen worden. Diese Unterschiede werden besonders deutlich, wenn man die Attribute betrachtet, die auf Dateien vergeben werden. DOS kennt vier Attribute:

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 vom Benutzer frei gesetzt und wieder zurückgesetzt werden. Sie bieten also keinen echten Zugriffsschutz, sondern nur eine gewisse Sicherung gegen Fehlbedienung.

Unix führt mit jeder Datei einen Satz von Zugriffsrechten mit. Diese sind aufgeteilt in 3 Gruppen von Benutzern: Der Dateibesitzer, die besitzende Gruppe und alle anderen. Jeder Gruppe können 3 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

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 besitztende 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.

13 Beispiel: Projektverzeichnisse

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`. Das Gruppenverzeichnis `/data/groups` sieht folgendermaßen aus:

```
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>
```

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 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. 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 ausgschlossen.

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.

14 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 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. Mit 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.

Im SMB-Protokoll wird zur Authentifizierung ein Challenge-Response Verfahren eingesetzt. Der Server verschickt an den Client eine Zufallszahl, die sogenannte Herausforderung. Der Client kennt das Benutzerpaßwort, und verschlüsselt die Herausforderung mit dem Paßwort als Schlüssel. Diesen verschlüsselten Wert verschickt der Client anstelle des Paßworts. Der Server kennt das Benutzerpaßwort ebenfalls, und kann den verschüsselten Wert entschlüsseln. Entsteht bei der Entschlüsselung wieder die Herausforderung, so hat der Benutzer die Herausforderung offensichtlich mit dem korrekten Paßwort verschlüsselt. Kommt etwas anderes heraus, war das Paßwort nicht richtig.

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 Verschlüselungsalgorithmus gewählt werden, der gegen einen solchen Angriff immun ist. Er kann keine Replay Attacke starten, da er bei jedem neuen Verbindungsaubau eine neue Herausforderung bekommt, die er verschlüssen muß.

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

Dieses 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,

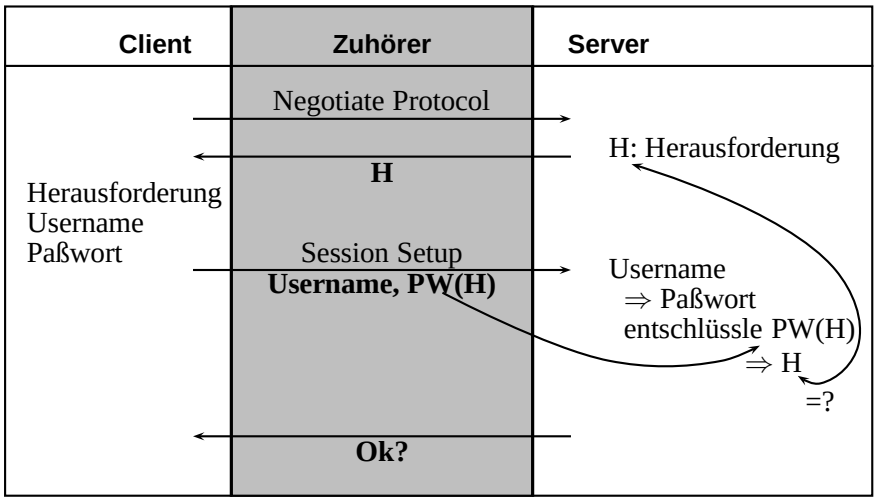

Abbildung 5: Challenge-Response Verfahren

den Wert aus der Datei `/etc/shadow`.

Eine Hashfunktion, wie sie unter Unix eingesetzt wird, hat drei Eigenschaften.

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, mit denen der Rückweg ausgeschlossen ist[6].

   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 sogenannte 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.

[6] Wie überall in der Kryptographie gilt dies auch nur so lange, bis jemand den Rückweg gefunden hat.

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 Paßwortdatenbank gepflegt werden, die Datei `smbpasswd`.

Auch in der Datei `smbpasswd` stehen keine Klartextpaßwörter. Bevor die Herausforderung mit dem Paßwort verschlüsselt wird, wird das Paßwort unter Windows ebenfalls durch eine Hashfunktion geschickt. Von dieser Hashfunktion gibt es zwei Varianten, die beide nicht mit den unter Unix verwendeten Funktionen übereinstimmen. 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, denn 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.

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[7].

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.

Sollte ein neugieriger Administrator einmal an den tatsächlichen Paßwörten seiner Benutzer interessiert sein, dann macht NT es ihm deutlich einfacher als Unix dies tut. Unix verwendet sogenannte 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.

[7] Allerdings nur für Samba 1.9, zu 2.0 hin wurde das Format geändert. Es gibt in Samba 2.0 aber ein Konvertierungsskript.

15 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 sogenannte 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 sogenannten 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`.

Mit der Option `printer =` wird festgelegt, welche Druckerwarteschlange unter Unix angesprochen werden soll. Dieser Drucker muß das Format verstehen, das vom Windowsdruckertreiber geliefert wird. Also sollte hier entweder Postscript angenommen werden, oder die Daten sollten per sogenannter 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.

16 Samba als Logon-Server

Wenn sich in einem Netz Windows 95/98 Clients befinden, kann es wünschenswert sein, daß sich die Benutzer dieser Arbeitsplätze nur mit einem Paßwort anmelden können, das zentral auf einem Server vorgehalten wird. Dazu muß der entsprechende Server spezielle Aufrufe von Clients entgegennehmen und korrekt beantworten. In der reinen Windowswelt ist dazu ein Windows NT Server notwendig, der als sogenannter Primary Domain Controller (PDC) installiert ist. Samba ist ebenfalls in der Lage, dies zu tun. Dazu ist im Abschnitt `[global]` der Parameter `domain logons = yes` zu setzen. Die Implementation, die Microsoft gewählt hat, um Domänenanmeldungen zu ermöglichen, erzwingt zusätzlich, daß der Domain Master Browser auf dem gleichen Rechner liegt wie der Logon Server. Das heißt, man benötigt für Domänenanmeldungen die folgenden Parameter:

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

Hat man diese Parameter gesetzt, kann man in den Eigenschaften des Clients für Microsoft-Netzwerke einstellen, daß der Client sich an der Domäne `samba` anmelden soll. Hat man verschlüsselte Paßwörter (Siehe Abschnitt 14) aktiviert, kann man vom Client aus sein SMB-Paßwort ändern, indem man das entsprechende Kontrollfeld in der Systemsteuerung von Windows benutzt.

17 Windows NT Domänen

Die Domänenanmeldung unter Windows 95/98 ist eine relativ einfache Sache, da es sich dabei praktisch nur um eine Überprüfung der Benutzerpaßwörter handelt. So etwas wie Benutzer kennt Windows 95 praktisch nicht, jeder Benutzer hat vollen Zugriff auf das gesamte System.

Erst mit Windows NT hat Microsoft den Schritt hin zu einem Betriebssystem gemacht, das Benutzerkonten und Zugriffsrechte verwalten kann. Damit sind sie sehr viel weiter gegangen, als Unix dies getan hatte. Um das Konzept der Windows NT Domäne zu verdeutlichen, soll hier zunächst auf das Konzept des Benutzers unter Windows und unter Unix eingegangen werden.

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 oder Yellow Pages. Die Benutzerdatenbank wird verteilt, gilt aber auf jedem Rechner rein lokal.

Unter NT sieht das sehr ähnlich aus, nur daß hier der Benutzer nicht durch eine kleine Zahl, sondern durch einen Security Identifier SID repräsentiert wird. Ein solcher SID ist mehrteilig. Der erste Teil dieses SID beinhaltet eine Kennung der Benutzerdatenbank, zu der dieser Benutzer gehört. Ein solcher SID ist 96 Bit lang und Microsoft behauptet, daß dieser Wert zufällig genug gewählt ist, daß es keine zwei Benutzerdatenbanken geben kann, die die gleiche SID haben. Der zweite Teil besteht aus einem sogenannten Relative Identifier RID, der den Benutzer innerhalb der Domäne eindeutig identifiziert. Die Kennung für die Domäne besteht aus 3 32-Bit Zahlen, die zusammen 96 Bit ergeben.

Unter Windows NT hat nun jeder Rechner eine eigene Benutzerdatenbank, genau wie unter Unix. Da aber jede Benutzerdatenbank eindeutig identifiziert ist, kann es keine zwei Benutzer mit gleicher Userid geben. Der Relative Identifier mag gleich sein, der Identifier für die Benutzerdatenbank unterscheidet sich aber auf jeden Fall.

Microsoft unterscheidet verschiedene Netzwerkmodelle. Das Peer-To-Peer Netz ist das Modell, das auch Unix zugrunde liegt. Hier hat jeder beteiligte Rechner eine eigene Benutzerdatenbank, eigene Paßwörter und eigene Rechtezuordnungen. Das Domänenmodell ist das Modell, das sich signifikant von Unix unterscheidet. Mit dem Domänenmodell wird eine Workstation in die Lage versetzt, mehr als eine Benutzerdatenbank zu benutzen. Neben der eigenen Benutzerdatenbank, die jede Workstation hat, kann sie eine Benutzerdatenbank von einem anderen Rechner importieren. In einer Windows NT Domäne gibt es einen Rechner, der seine eigene Benutzerdatenbank anderen zur Verfügung stellt, den sogenannten Primary Domain Controller. Dieser reserviert für sich spezielle NetBIOS-Namen, um sich den Workstations als Logonserver anzubieten. Eine Workstation befragt den Primary Domain Controller nach allen relevanten Daten zu den Benutzern, die sich bei ihr anmelden wollen, und die Rechte auf der Workstation wahrnehmen können.

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 Pro-

tokolle, wie beispielsweise der 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. Daher muß jede Workstation explizit in die Domäne aufgenommen werden.

Bei Samba ist es so, daß es zu jedem Benutzer, der ein Paßwort in der `/etc/smb.conf` hat, einen Benutzer im System geben muß. Der zu einer Workstation gehörende Benutzer muß den NetBIOS-Namen der Workstation, ergänzt um ein $-Zeichen, haben. Man benötigt also zwei Schritte, um eine Workstation in die Domäne aufzunehmen. Im ersten Schritt wird der Unixbenutzer angelegt. Dies geschieht in vielen Linuxsystemen mit dem Kommando `useradd -m <user>`. Der angelegte Benutzer benötigt im Unixsystem weder ein Paßwort noch ein Heimatverzeichnis. Er ist notwendig, da die Workstation in der Domäne eine eigene SID bekommt, die aus der Unix userid berechnet wird. Dann muß die Workstation ein Paßwort in der `/etc/smbpasswd` bekommen, und zwar mit dem Befehl `smbpasswd -a -m name`. Ein Beispiel sieht folgendermaßen aus:

```
root@erde: useradd -m wks\$
root@erde: smbpasswd -a -m 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.

18 Samba als Domänenmitglied

Mit dem Parameter `security` kann man den Zeitpunkt steuern, zu dem das Benutzerpaßwort geprüft wird. `security = share` legt fest, daß die Prüfung beim Tree Connect stattfindet, das heißt, wenn die Freigabe angesprochen wird. Ist `security = user` angegeben, wird das Paßwort bereits einen Schritt vorher, also beim Session Setup geprüft. Bei `security = user` wird also die Kombination von Benutzer und Paßwort geprüft bei `security = share` die Kombination Freigabe und Paßwort.

Der Parameter `security` kann noch zwei weitere Werte annehmen: `server` und `domain`. Bei beiden Einstellungen verhält sich Samba gegenüber dem Client genau wie bei `security = user`, der Benutzer muß sich unter seinem Namen beim Server authentifizieren. Die Unterschiede liegen in der Art und Weise, wie das Paßwort überprüft wird.

- `security = user`: Die Überprüfung findet anhand einer lokalen Datenbank statt. Werden Klartextpaßwörter verwendet (`encrypt passwords = no`), so wird die lokale Unix-Paßwortdatenbank in `/etc/passwd`, `/etc/shadow` oder die entsprechende NIS-Tabelle herangezogen. Bei verschlüsselten Paßwörtern mit wird die Samba-eigene Paßwortdatenbank in der Datei `smbpasswd` zur Überprüfung herangezogen.

- `security = server`: Bei dieser Einstellung bekommt der Sambaserver vom Client einen Benutzernamen und ein Paßwort präsentiert. Er versucht daraufhin, sich mit diesem Paßwort bei einem weiteren Server anzumelden. Funktioniert dies, hat der Benutzer sein Paßwort offensichtlich richtig eingegeben. Schlägt dies fehl, wird auch dem Client des Sambaservers der Fehler mitgeteilt und der Zugriff verweigert. Der Paßwortserver, der zur Überprüfung herangezogen wird, muß mit seinem NetBIOS-Namen im Parameter `password server` angegeben werden.

- `security = domain`: Auch hierbei wird die Überprüfung einem Paßwortserver überlassen. Dieser muß jedoch ein Primary Domain Controller sein, der den Sambaserver in die Domäne aufgenommen hat. Der Hauptvorteil gegenüber `security = server` besteht in einer deutlich reduzierten Last auf dem Paßwortserver und einer verschlüsselten Kommunikation zwischen Samba und Paßwortserver.

Um einen Windowsrechner dazu zu bringen, für einen Sambaserver die Paßwortüberprüfung zu übernehmen, muß man nur `security = server` und den `password server` passend setzen. Dabei übernimmt der Server ausschließlich die Überprüfung der Paßwörter. Bei verschlüsselten Paßwörtern können Benutzer nur dann in die `smbpasswd` aufgenommen werden, wenn sie in der Unix-Benutzerdatenbank existieren. Genau so verhält es sich bei `security = server`. Benutzer können auf Samba nur dann zugreifen, wenn sie als normale Unixbenutzer existieren.

`security = server` ist nicht die optimale Lösung für die Überprüfung von Paßwörtern durch einen weiteren Rechner.

Um die Vorteile der Domänenmitgliedschaft zu nutzen, ist etwas mehr Aufwand notwendig. Mitglied einer Domäne zu sein heißt, mit dem Primary Domain Controller über einen verschlüsselten Kanal kommunizieren zu können. Diese Verschlüsselung wird verwendet, um Benutzerinformationen verdeckt austauschen zu können. Als Verschlüsselungsverfahren kommt ein symmetrisches oder auch secret key Verfahren zum Einsatz. Um ein symmetrisches Verfahren anwenden zu können, müssen sich beide Partner über ein gemeinsames Geheimnis, den *secret key* einig sein. Ein solches gemeinsames Geheimnis muß regelmäßig geändert werden, um einer großen Klasse von kryptographischen Angriffen auszuweichen. Eine solche Änderung darf selbstverständlich nicht abgehört werden können, da ein Zuhörer damit die gesamte Kommunikation abhören kann. Für die Änderung eines Geheimnisses gab es bereits vor der Implementation des Domänenprotokolls ein fertiges Protokoll, das man direkt verwenden konnte: Die Möglichkeit, Benutzerpaßwörter über das Netz zu ändern, war mir einem gesicherten Protokoll implementiert. Um dieses Protokoll zur verschlüsselten Kommunikation zwischen einer Workstation oder einem Mitgliedsserver und dem Domänencontroller nutzen zu können, muß es für jedes Domänenmitglied ein Benutzerkonto geben. Genau dies wird auf dem Domänencontroller erstellt, wenn man eine Workstation oder einen Server mit dem Servermanager in die Domäne aufnimmt. Betritt man danach mit der Workstation die Domäne, wird als erstes das Paßwort des Computerkontos geändert.

Um einen Sambaserver in eine Domäne aufzunehmen, sind zwei Schritte notwendig.

- Auf dem Server muß der Sambaserver mit seinem NetBIOS-Namen in die Domäne aufgenommen werden.

- Der Sambaserver selbst muß darüber informiert werden, daß er sich in der Domäne befindet, und er muß sein Paßwort ändern. Dies geschieht mit dem Befehl

  ```
  smbpasswd -j DOM -r PDC
  ```

  Dabei steht `DOM` für die Domäne, die betreten wird. Mit `PDC` wird der NetBIOS-Name des Domänencontrollers der Domäne benannt.

Mit diesem Kommando wird das Maschinenpaßwort auf dem PDC auf einen neuen, zufälligen Wert geändert. Dieses neue Maschinenpaßwort für den Samba Server wird in einer Datei im gleichen Verzeichnis wie die Datei `smbpasswd` abgespeichert und hat folgenden Namen:

`<NT DOMAENENNAME>.<Samba Servername>.mac`

Die Endung .mac steht für *Machine ACcount* Paßwortdatei. Im obigen Beispiel würde die Datei also `DOM.SERV1.mac` heißen.

Diese Datei wird von root erstellt und ist für keinen anderen Benutzer lesbar. Sie ist der Schlüssel zu Ihrer Domänensicherheit und sollte genau so vorsichtig behandelt werden wie die Datei `/etc/shadow`.

Nach diesen beiden Schritten kann man mit `security = domain`, `password server = PDC BDC1 BDC2` und `encrypt passwords = yes` die Paßwortüberprüfung an einen der Domänencontroller delegieren. Dies sind die Primären und Backup Domänencontroller, die Samba der Reihe nach kontaktieren wird, um Benutzer zu authentifizieren. Samba wird sie in der aufgeführten Reihenfolge ansprechen. Sie können also die Reihenfolge verändern, um eine günstigere Lastverteilung zu erreichen. Eine weitere Option ist die Angabe `password server = *`. Damit sucht Samba mit den Standardmethoden[8] von Windows NT nach einem Domänencontroller und befragt die Server, die es bei dieser Anfrage herausbekommen hat.

Warum ist `security = domain` besser als `security = server`?

Der Vorteil der Domänensicherheit ist, daß Samba die Authentifizierung über einen gesicherten RPC Kanal schickt, genau wie ein Windows NT Server es tun würde. Das heißt, daß Samba nun genau wie ein Windows NT Server an einer Vertrauensstellung teilnehmen kann. Das heißt, Sie können einen Samba Server in eine Resourcendomäne aufnehmen, und Sie können die Authentifizierung via Resourcen PDC vom PDC der Benutzerdomäne vornehmen lassen.

Zusätzlich muß in der Einstellung `security = server` der Samba Dämon eine Verbindung zum Authentifizierungsserver während seiner gesamten Laufzeit offenhalten. Dies kann die Anzahl der offenen Verbindungen auf einem Windows NT Server in die Höhe treiben, so daß dieser keine Verbindungen mehr annimmt. Mit `security = domain` verbinden sich die Samba Dämonen nur so lange mit dem PDC, wie es für die Benutzerauthentifizierung notwendig ist. Danach wird die Verbindung wieder abgebaut, so daß die Verbindungen wieder anderweitig verwendbar sind.

Und nicht zuletzt bekommt der Samba Server als Teil der Antwort auf die Authentifizierungsanforderung Informationen über den Security Identifier, die Gruppenzuordnungen und andere Informationen über den Benutzer. All diese Informationen werden Samba zukünftig erlauben, in einem sogenannten Appliance Modus zu laufen. In diesem Modus wird kein manuell angelegter Unixbenutzer mehr notwendig sein. Samba wird Unix Benutzer und Gruppen aus der

[8] Windows NT findet einen Domänencontroller, indem der NetBIOS-Name `DOMAIN<1C>` gesucht wird.

Authentifizierungsantwort des PDC erzeugen. Damit wird Samba wirklich ein Plug and Play Mitglied einer Domäne.

Dieser Appliance Modus kann heute schon annähernd erreicht werden, indem bei Samba der Parameter `add user script` angegeben wird. In diesem Parameter wird ein Unixprogramm angegeben, das dynamisch einen Unixbenutzer erzeugen muß, nachdem ein Paßwortserver die Korrektheit eines Paßworts bestätigt hat. Ein Beispiel kann sein:

```
add user script = /usr/bin/useradd -m %U
```

Damit wird einfach ein Benutzer hinzugefügt, wenn er noch nicht existiert, aber der PDC das Paßwort bestätigt hat.