Gerald Carter



SAMS

Teach Yourself Samba in 24 Hours

SECOND EDITION

SAMS

800 East 96th St., Indianapolis, Indiana, 46240 USA

Sams Teach Yourself Samba in 24 Hours, Second Edition

Copyright © 2002 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-32269-2

Library of Congress Catalog Card Number: 2001093500

Printed in the United States of America

First Printing: December 2001

04 03 02 01 4 3 2 1

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Associate Publisher
Jeff Koch

Acquisitions Editor
Katie Purdum

Development Editor
Mark Renfrow

Managing Editor
Matt Purcell

Copy Editor
Michael Kopp,
Publication Services, Inc.

Production Editor
Theodore Young, Jr.,
Publication Services, Inc.

Indexer
Richard Bronson,
Publication Services, Inc.

Proofreader
Publication Services, Inc.

Technical Editor
John Traenkenschuh

Team Coordinator
Denni Bannister

Interior Designer
Gary Adair

Cover Designer
Aren Howell

Page Layout
James Torbit,
Publication Services, Inc.

Contents at a Glance

	Introduction	1
Part I	**Starting Up**	**5**
1	Introduction to Samba	7
2	Windows Networking Concepts	15
3	Obtaining the Latest Release of Samba	35
4	Starting Your Feet to Dance	45
Part II	**Basic Configuration**	**63**
5	Exploring Samba's `smb.conf` File	65
6	SWAT and Other GUI Administration Tools	81
7	Security Levels and Passwords	95
8	Samba—The File Server	115
9	Samba—The Print Server	137
Part III	**The Social Life of Samba**	**161**
10	Microsoft's DOS Network Client and Windows 95/98/ME	163
11	Windows NT 4.0/2000	181
12	CIFS Clients for Unix	193
13	Troubleshooting Techniques	215
Part IV	**Going Production**	**243**
14	Replacing a Windows NT File and Print Server—Lock, Stock & Barrel	245
15	Server-Side Automation	275
16	Managing User Accounts and Single Sign-On	289
17	Security Tips	311
18	WINS and NetBIOS Name Services	327
19	Local Subnet Browsing	339
20	Cross Subnet Browsing	351
21	Domain Control for Windows 95/98/ME	363
22	Domain Control for Windows NT 4.0/2000	377
23	Capacity Planning and System Tuning	395
24	Exploring the Samba Community and Looking Down the Road	409
	Index	417

Contents

Introduction 1

Part I Starting Up 5

Hour 1 Introduction to Samba 7

A Brief Samba History Lesson 7
A Brief Description 8
What Was Life Like Before Samba? 10
A Brief Look at Supported Platforms 12
The GNU General Public License (GPL) 12
Summary 13
Q&A 14

Hour 2 Windows Networking Concepts 15

NetBIOS Overview 16
What Is a "Name Service"? 17
NetBIOS Names and Registration 18
Name Resolution 22
Session Service 24
Datagram Service 25
CIFS (SMB) 25
SMB over NetBIOS 25
Connection-Oriented Protocol 26
Protocol Overview 27
Windows Networking Models 29
Workgroups 29
Domains 30
Summary 31
Q&A 32
New Terms 32

Hour 3 Obtaining the Latest Release of Samba 35

Download Sites and Methods 36
Before Upgrading an Existing Server 37
Backing Up Important Files 37
Compiling Samba 39
The Filesystem Hierarchy Standard (`--with-fhs`) 41
What Goes Where When I Type `make install`? 42
Binary Distribution Methods 43
Summary 44
Q&A 44

HOUR 4 Starting Your Feet to Dance **45**

Which Binaries? 45
Samba's Configuration File: `smb.conf` 46
Configuring the [`global`] Section of `smb.conf` 47
Setting Up the Shared Group Directory 48
Creating the Directory on Disk 49
Adding the Group Share to `smb.conf` 50
Verifying `smb.conf` 51
Starting `smbd` and `nmbd` 52
Starting from `inetd` 52
Starting from `xinetd` 53
Running as a Daemon 54
Command-Line Arguments 55
Testing the Installation 56
A Note Regarding Windows Clients and Our Server 58
Other Tools Included with Samba 59
Summary 60
Q&A 61
New Terms 61

Part II Basic Configuration **63**

HOUR 5 Exploring Samba's `smb.conf` File **65**

Layout 65
The [`global`] Section 67
Variables 67
Environment Variables 69
Parameters 70
Common [`global`] Parameters 70
NetBIOS Naming Parameters 71
Logging 72
Internationalization 76
State Information 79
Summary 79
Q&A 80

HOUR 6 SWAT and Other GUI Administration Tools **81**

SWAT 81
The `inetd` Server 82
The `xinetd` Server 83
Logging In 83
Managing the [`global`] Section 85
Managing File Shares 86

Managing Printer Shares 88
Obtaining Status Information 89
Viewing the Complete `smb.conf` File 90
Changing Your Password 90
Security Concerns 90
Webmin 91
Summary 93
Q&A 94

HOUR 7 Security Levels and Passwords 95

Security Levels and the `security` Parameter 95
`security = user` 96
`security = share` 97
`security = server` 99
`security = domain` 101
Usernames 101
`username level` 102
`username map` 103
Passwords 104
`password level` 105
LAN Manager and Windows NT Password Encryption 105
Null Passwords in `smbpasswd` 108
Guest Access 111
Summary 113
Q&A 113
New Terms 114

HOUR 8 Samba—The File Server 115

Beginning Configurations and Accounts 116
A Basic, Group Accessible Share 116
Differences Between File Systems on Windows and Unix 119
A Restricted Access Service 121
File Share Access Control Lists 124
DOS Attributes and Timestamps 125
A Publicly Accessible Share 126
Modifying Unix Permissions from Windows Clients 128
Share Mode, Deny Modes, and Oplocks 130
User Home Directories 132
Symbolic Links 134
File System Quotas 135
Summary 135
Q&A 136

HOUR 9 Samba—The Print Server 137

Prerequisites 138
Samba's Printing Support 138
Configuring Our First Printer 140
Adding Print Drivers to the Server 144
Using the Add Printer Wizard to Create and Delete Printers on the Server 151
Using Local Print Drivers on Clients 152
The [printers] Share 153
Securing Printer Shares....156
Troubleshooting Printer Shares 157
Summary 158
Q&A 159

Part III The Social Life of Samba 161

HOUR 10 Microsoft's DOS Network Client and Windows 95/98/ME 163

The Network Redirector 164
Microsoft Network Client Version 3.0 for MS-DOS 165
Obtaining the Software 165
Installing the Client 166
Making the Network Boot Disk 171
Windows 9x/ME 173
Configuring the Client....174
Logging In to the Network 178
Summary 179
Q&A 180
New Terms 180

HOUR 11 Windows NT 4.0/2000 181

Windows NT 181
Installing the Network Adapter 182
Installing the TCP/IP Protocol 183
The Workstation Service 184
Setting the NetBIOS Machine Name and Workgroup Name 185
Configuring the Windows 2000 Client 186
NetBIOS Names and Workgroups....188
Conflicting Credentials 189
Summary 190
Q&A 191

HOUR 12 CIFS Clients for Unix 193

The Linux Kernel and `smbfs` 194
Allowing Normal Users to Mount `smbfs` Filesystems 200
`smbclient` 201
Using `smbclient` to `get` and `put` Files 204
Backing Up a Directory....206
Printing a File 207
The `smbprint` Script 208
Printing from Unix to Windows 209
Printing to Windows with BSD-Style Printing 210
Printing to Windows with System V-Style Printing 210
Resolving Hostnames Using the NetBIOS Name Service 211
Summary 213
Q&A 213

HOUR 13 Troubleshooting Techniques 215

Egypt, Samba, and Bugs....216
Available Tools at Our Disposal 217
Documentation....217
People 217
The Test Environment....218
Level 1: Beginning the Climb up the Pyramid....219
Pinging the Server and Client 219
Comparing Network Broadcast Addresses 221
Verifying the `smb.conf` Settings 223
Level 2: Local Server and Client Software 224
`smbd` 224
`nmbd` 226
Windows' NetBIOS Interface 227
Level 3: Remote Access to Shares 228
Name Resolution 228
Enumerating Shares from the Windows Client....229
Connecting to a Share Remotely....230
Level 4: Network Browsing 231
Level 5: A Wealth of Information—Log Files and Packets 231
Samba's Log Files 231
Monitoring Network Traffic 234
Level 6: Samba Internals 238
Summary 239
Q&A 240
New Terms 241

Part IV Going Production 243

HOUR 14 Replacing a Windows NT File and Printer Server—Lock, Stock & Barrel 245

The Existing Network 246
Samba and ACLs 248
Configuring the Linux Server 249
`--with-acl-support` 255
The Replacement Process 255
Step 1: Joining the Domain… 256
Step 2: Creating the New Shares and Transferring the Data 262
Step 3: Testing the Server 265
Managing Shares on a Samba Member Server 267
Microsoft's Distributed File System 270
Summary 272
Q&A 273
New Terms 274

HOUR 15 Server-Side Automation 275

What Is Server-Side Automation? 275
`preexec` and `postexec` Scripts 276
The `preexec` and `postexec` Parameters 276
The `root preexec` and `root postexec` Parameters 279
The `message command` Parameter 281
The `include` Parameter 282
Samba's Virtual File System for Shares 286
Summary 287
Q&A 287
New Terms 287

HOUR 16 Managing User Accounts and Single Sign-On 289

Password Synchronization Between `/etc/passwd` and `smbpasswd` 290
Samba `--with-pam` 293
PAM in 5 Minutes 294
PAM Support in `smbd` 296
Unix Services in a Windows Domain—`pam_winbind` 299
Single Sign-On Samba and Unix—`pam_smbpass` 304
Recording User Connections to the Unix Login Records (`utmp`) 307
Samba's New `--with-ldapsam`, `--with-tdbsam`, and `--with-nisplussam` Options 308
Summary 309
Q&A 309

HOUR 17 Security Tips **311**

Authentication 312
Configuration File Permissions 314
Distributing `smbpasswd` to Multiple Servers 316
Virus and Internet Worm Prevention 318
Securing Access to SWAT Via SSL 319
Using Samba and SSL Together 320
Firewalls 321
Samba's Very Own Bouncer 323
Summary 324
Q&A 325

HOUR 18 WINS and NetBIOS Name Services **327**

WINS 328
The `wins server` Parameter 330
The `wins support` Parameter 331
The `wins proxy` Parameter 331
The `dns proxy` Parameter 332
The `lmhosts` File 333
The `name resolve order` Parameter 334
Name Resolution Client or Server? 335
WINS and Windows 2000 335
Summary 336
Q&A 336
New Terms 337

HOUR 19 Local Subnet Browsing **339**

Introduction to Browsing 340
Samba Browsing Parameters 343
Browsing with OS/2 Clients 344
Viewing a List of Shares 345
The `auto services` (a.k.a. `preload`) and `load printers` Parameters 345
Browsing Examples 346
Browsing Problems 348
Summary 348
Q&A 349
New Terms 350

HOUR 20 Cross Subnet Browsing **351**

Browsing Across Subnets 351
The `domain master` Parameter 354
Designing Network Browsing 354
Troubleshooting Remote Browsing 358

Additional Browsing Parameters 359
The `remote announce` Parameter 359
The `remote browse sync` Parameter....360
The `enhanced browsing` Parameter....360
Summary....360
Q&A 361

HOUR 21 Domain Control for Windows 95/98/ME 363

Setting Up the Samba Domain Controller 364
Setting Up a Windows 9x Client 367
Successfully Logging In to the Domain....368
Troubleshooting Windows 9x/ME Domain Logons....368
User Profiles 370
The Windows Registry 101 370
What Else Exists in a User Profile? 371
Making User Profiles Work for You 372
Windows System Policies 373
Summary....375
Q&A 375

HOUR 22 Domain Control for Windows NT 4.0/2000 377

Current Features 378
How to Configure a Samba PDC 379
Machine Trust Accounts 380
Creating a Machine Trust Account on a Samba PDC 381
Joining the Domain 384
Time to Reboot the Client 386
`Domain Admins` and `Domain Guests` 387
`logon home`, `logon drive`, and `logon path` 387
User Profiles and System Policies 389
Implementing Backup Domain Controller (BDC) for a Samba-Controlled Domain 390
Windows 2000 Domains....393
Summary....394
Q&A 394
New Terms 394

HOUR 23 Capacity Planning and System Tuning 395

How Big is Big Enough? 395
How Much CPU Power is Enough? 399
Tuning Your Server....399
Windows NT/2000/XP Clients, MS-RPC, and the `deadtime` Parameter 400
Defining a Ceiling 401
Oplocks and Performance 402
The `socket options` Parameter 403

High Availability and Fail Over 404
Summary 407
Q&A 408

HOUR 24 Exploring the Samba Community and Looking Down the Road 409

Who Is the Samba Team? 409
Who Guides Samba Development? 410
Future Plans for Samba 412
Samba 3.0 413
Beyond 3.0 414
How Can I Help? 414
Summary 415
Q&A 415

Index 417

About the Author

Gerald Carter has been a member of the Samba Team, a group of people worldwide who develops and documents Samba, since 1998 and is actively involved in distributed network authentication solutions. His journey with Samba began three years prior while employed as a network administrator for Auburn University. Gerald has published articles with various web-based magazines such as Linuxworld and has authored instructional course for companies such as Linuxcare. Today Gerald spends much of his time teaching at conferences, writing, and developing Samba.

Gerald, known by his friends as Jerry, first became interested in computers in 1983 with a Commodore 64 and a copy of "Zork I: The Great Underground Empire," which he now carries on his Palm III (Zork I, not the C64). In 1997, he received his Master's degree in Computer Science from Auburn University, where one day he hopes to finish pursuing his PhD, also in Computer Science.

His hobbies include running, hiking, and playing music. Gerald presently resides near Lake Martin in Dadeville, Alabama, with his wife Kristi. If you would like to contact him, Jerry's e-mail address is `jerry@samba.org` and his web site is `http://www.plainjoe.org/`.

About the Tech Editor

John Traenkenschuh is a former educator, Safety Editor for a magazine, certified Motorcycle Safety Principal Instructor, and mechanic who now dabbles in computers, networks, and Computer and Network security to the exclusion of all other hobbies, such as fishing and wood-working.

He has been a security analyst at two Fortune 100 companies and is a Certified Checkpoint Security Engineer. He now divides his work life between Web Server administration and research projects delving into the cost-effectiveness of Open-Source alternatives, such as Samba. He and his wife enjoy time spent with their two teenage daughters, two cats, and one little hyperactive dog named Molly. Most of the time.

Dedication

To God (1 Corinthians 10:31) and to my wife, Kristi. With each day, the smiles grow and the laughter multiplies.

Acknowledgments

After surviving through two editions of this book, I know now more than ever that no book is ever written in isolation. My first thanks is to my Lord and Savior, Jesus Christ, for the opportunity to begin and complete this project. I only hope that I have been a good steward of what He has entrusted to me. Once again, my wife helped me pull this off by being the most amazing person I have ever known. Thank you so much Kristi. Thanks to Katie Purdum for convincing me to do the second edition, and also a generous round of applause is due to all of the people at Sams who took care of all the details that drive me crazy. Finally, thanks to all my friends on the Samba Team. Samba would not be the same without you all.

Tell Us What You Think!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an associate publisher for Sams Publishing, I welcome your comments. You can e-mail or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that, due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author name as well as your name and phone or fax number. I will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: `feedback@samspublishing.com`

Mail:

Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Foreword

Much has changed in the Samba world since I wrote the foreword for the first "Teach Yourself Samba in 24 Hours" book. Two years have passed, and two major versions of Samba have been released. Gerald is the author of significant parts of the code in these versions and of much of the included documentation. Millions of people depend upon him, as one of the primary members of the Samba Team, to create timely, bug free code releases, and they have much to thank him for! Much has also changed in this edition of "Teach Yourself Samba in 24 Hours," with a new focus on security, which is very important in the hostile world of computer networking.

Samba has become one of the most significant parts of every Unix system. It allows Windows clients to use Unix file and print resources transparently, without their users even knowing they're talking to a non-Windows server. The new versions of Samba can also provide Windows domain authentication from Unix servers, giving single sign-on capability. This flexibility is hard to achieve without understanding the internals of Windows networking. This book provides that knowledge and, for that reason, belongs not only on the shelf of every Unix administrator who wants to work with Samba, but also on the shelf of every Windows NT, Windows 2000, or Windows XP administrator.

Gerald is an expert in every aspect of Samba and an experienced network administrator. Having come from a system administration background, he not only writes code and fixes bugs in Samba, but he also has to make Samba work and keep it working in production environments. More importantly for the author of a "Teach Yourself" book, he knows how to teach, having provided USENIX tutorials on Samba installation and administration for many years (even if he "tells too many bad southern jokes" and "isn't as funny as he thinks he is" *). This book will teach Samba administrators to become Samba experts and will answer many of the hard questions that even experienced Samba administrators struggle with.

Gerald is a good friend and a colleague I can depend upon. I am honored to be asked to write this foreword and wholeheartedly recommend the book you have in your hand. Even though I write much of the code in Samba, when I want to set up a Samba server I reach for Gerald's book.

Jeremy Allison,
Co-Author of Samba
San Jose, CA
October 2001

(*) Actual quotes from audience responses to Gerald's tutorials :-).

Introduction

Welcome to the world of Samba! We will spend the next 24 hours together learning about a tool that can help integrate Unix servers with Windows clients and vice versa. The second edition of this book contains 24 updated and rewritten lessons on what I consider to be the most important information for configuring and maintaining Samba installations.

Each lesson includes sample configurations, diagrams to illustrate concepts, and hands-on examples. There is even a section at the end of each chapter to summarize things we have learned and to provide answers to common questions that you may ask.

After working through all the lessons, this book will find a place on your shelf close to your desk or computer within an arm's reach, just for those times when you need to verify a Samba feature or parameter. In short, I believe that this book will be an invaluable investment for you. These chapters contain information and examples that I have used daily when previously working as a systems administrator and now during Samba development.

What Is Samba?

Webster's Dictionary defines Samba as "a Brazilian dance of African origin characterized by a dip and a spring upward with a bending of the knee at each beat of the music." That is of course not the Samba I am going to talk about. In fact, at no time in this book will you ever be asked to dance (unless of course you feel like it when something works!).

Samba is an implementation of a Common Internet File System (CIFS, also known as SMB) protocol server that can be run on almost every variant of Unix in existence. Microsoft clients will use this protocol to access files and printers located on your Unix box just as if it were a native Windows server.

Samba is an open source project, just like the Linux kernel (a Unix-like operating system for PCs). The source code, written in C, is always available to you to explore, test, or change. And it's free!

The implication of these items is that Samba is being installed in more and more server rooms in order to provide file and print services to Microsoft Windows clients without installing a Windows NT/2000 Server or any other CIFS server.

Offices of all sizes can benefit from Samba. My mother even uses Samba! (No joke!) Her small office network of three computers uses a bare bones PC running Linux and Samba to offer home directories, group disk shares, and shared printers. It is a simple, cost-effective solution for her to use and for me to support (we currently live about three hours apart).

Why Teach Yourself Samba?

There are many reasons to learn about Samba, whether you are a full-time network administrator or just have a couple of PCs at your home:

- Samba enables you to share files and printers between Unix servers and Microsoft Windows clients.
- Samba provides a way to authenticate user logons from PC clients.
- The combination of Samba and one of the free Unix compatible operating systems, such as Linux or FreeBSD, is an efficient, stable, cost-effective replacement for some PC servers.
- There's a growing job market for Samba administrators.

Who Should Use This Book?

This book is designed for people who have a general working knowledge of Unix-based systems. Although you are not expected to be an expert, you should have a working knowledge of things like `ps`, `grep`, and `kill`. It is also helpful to know how to use the `make` utility and an ANSI C compiler, such as `gcc`. It will also be to your benefit to know how to configure new hardware or software on a Windows based PC.

Conventions Used in This Book

Features in this book include the following:



Notes provide you with comments and asides about the topic at hand.



Tips offer shortcuts and hints on getting the task done.



Cautions explain roadblocks you might encounter when you work with Perl and tell you how to avoid them.

At the end of each chapter, you'll find handy Summary and Q&A sections. Many times, you'll also find a New Terms section.

In addition, you'll find various typographic conventions throughout this book:

- Commands, variables, directories, and files appear in text in a special `monospaced font`.
- Commands and such that you type appear in **boldface type**.
- Placeholders in syntax descriptions appear in a *`monospaced italic`* typeface. This indicates that you will replace the placeholder with the actual filename, parameter, or other element that it represents.



PART I
Starting Up

Hour

1 Introduction to Samba

2 Windows Networking Concepts

3 Obtaining the Latest Release of Samba

4 Starting Your Feet to Dance



HOUR 1

Introduction to Samba

Over the past years, I have spent many hours answering questions and responding to general e-mail. Some of these are very specific questions; others lean towards a more generic sense. However, most begin with "Can I...?" or "Is it possible...?" This book is dedicated to answering some of the more common questions about Samba, Windows Networking, and how to integrate Unix and Windows hosts on a network. This hour in particular will help you to understand where Samba came from, where it has been, and how it can be useful to you.

A Brief Samba History Lesson

Samba is an Open Source Software (OSS) project, first developed in 1991 by Andrew Tridgell, then a Ph.D. student in the computer science laboratory at the Australian National University in Canberra, Australia. He had been using PC-NFS to connect to files on Sun workstations. On the arrival of a beta copy of eXcursion from Digital, he began testing this new client. To some disappointment, however, the only servers that supported the eXcursion client were on VMS and Ultrix systems. Being curious, as most computer science graduate students are, Andrew began to think about implementing

the file-sharing protocol on platforms other than Digital workstations. At the time, he had never heard of the terms NetBIOS or SMB. In fact, this was his first venture into network socket programming. Somewhere the mumbled phrase "How hard could it be?" was heard, and a short time later Andrew had a (somewhat) working connection to the Sun using the eXcursion client.

The initial implementation of his server had a lot of hard-coded, magic values, which simply replicated responses made by the Ultrix server. Andrew was first introduced to the NetBIOS protocol when speaking to a person at Digital. However, it was not until two years after his first implementation that he saw specifications for the SMB protocol and learned what all the magic values represented.

Andrew released his first implementation in January 1992. During the next two years, he used an X terminal mostly and had no need to develop his pet project further. During this time, Andrew was also introduced to Linux by a chap named Dan Shearer. When a community began to grow around his SMB server, development of Samba resumed, and the rest, as they say, is history.

One commonly asked question is "What does Samba mean?" The answer is fairly simple. Andrew's original software had been named "SMBserver," but because of legal issues the name had to be changed. Just for fun, Andrew began searching through the file /usr/dict/words on his local workstation. One of the words located by

```
$ grep '^[sS].*[mM].*[bB]' /usr/dict/words
```

was "Samba".

A Brief Description

Today, Samba is available for download from various official mirrors listed at `http://samba.org`. It is also included in all modern Linux distributions as well as being packaged for many commercial Unix systems. You can download and compile the source code for yourself, or you may choose to use precompiled binaries for your platform.

A simple description of Samba is a collection of client and server programs that enable Windows clients to access files and printers on a Unix server and vice versa. More specifically, Samba is a free implementation of the client and server portion of the *Server Message Block* (SMB) protocol. While it is primarily developed for Unix-based hosts, Samba has also been ported to other platforms such as the Amiga and VMS.

All Windows clients, as well as other PC platforms such as OS/2, use the SMB protocol, recently renamed to *Common Internet File System* (CIFS), to access remote file systems and printers. These are referred to as "shares" or "services" in Windowspeak. For many

sites, Samba's ability to export Unix file systems and printers to Windows clients is enough to sell the idea and might be all that is ever used at a site.

However, Samba has many other capabilities that can be icing on the cake, and many new ones have been added to the 2.2 release. Here's a list of some of the things Samba can do that we will explore in later hours:

- Samba can act as a NetBIOS name server (Hour 18).
- Samba can participate fully in NetBIOS browsing and browse master elections with Windows NT 4.0 domains and workgroups (Hours 19 and 20).
- Samba contains CIFS clients that allow Unix boxes to access shares or printers on other CIFS servers, PCs, or other Samba servers (Hour 12).
- Samba provides a command line utility for remote administration of Windows NT servers and Samba servers.
- Samba 2.2 has official support for acting as a Domain Controller for Windows NT 4.0-style domains supporting Windows 9x/ME/NT/2000 clients (Hours 21 and 22).
- Samba 2.2 possesses built-in support for Access Control Lists (ACLs) on printers and file shares (Hours 8 and 9).
- Samba 2.2 supports manipulating file system ACLs from the Windows NT/2000 Security tab in the Windows Explorer. In contrast to the ACLs on printers and file shares, this option is available only on certain platforms that support POSIX ACLs (Hour 14).
- Samba 2.2 has improved capabilities for integrating with PAM's (Pluggable Account Management) account and session management features (Hour 16).
- Samba 2.2 supports acting as an MS-Dfs root server (Hour 14).
- Samba 2.2 fully supports both Windows NT RPC-based printing and Windows 9x-style LanMan printing. Unlike version 2.0, Samba 2.2 supports the "point and print" features of Windows NT, including downloading of printer drivers to clients upon demand (Hour 9).

A Samba server can fit into existing networks in many ways. Here are some common examples:

- Samba can replace a Windows NT File/Print server for licensing cost reasons (Hour 14).
- Samba 2.2 can provide a unified logon system for Unix/Linux clients using Windows NT domain user and group accounts. This is accomplished through the Winbind PAM and NSS modules (Hours 14 and 16).

- Samba 2.2 can act as a Primary Domain Controller (PDC) for a Windows NT 4.0 domain (Hours 21 and 22).
- Samba Provides a gateway for synchronizing Unix and Windows NT passwords (Hour 16).
- Samba acts as a "home directory" server so that Unix home directories and Windows home directories exist in a common space (Hour 8).
- Samba can act as a print gateway between PCs and Unix networked printers (Hour 9).

These are only a few examples. In most circumstances, you will be limited only by your own imagination and creativity.

What Was Life Like Before Samba?

You might be familiar with some of the solutions used in the past to allow Windows clients to access remote files or printers. At a site where I once worked, the only remote access to files was through FTP, and to do remote printing one saved one's file to a floppy disk and physically carried it to the remote machine—a "protocol" that was affectionately known as "Sneakernet." Although that sort of setup had its advantages (it was simple), today's users need remote access that allows for work of a more collaborative nature.

During the late 1980s, several companies began to develop client software to allow PCs to access Unix file systems exported via NFS (Network File System). This became known as PC-NFS, and many sites embraced it as a way to integrate PCs with existing Unix-based infrastructures.



NFS was first developed by Sun Microsystems. It has since become the de facto means of providing remote access to files among Unix hosts connected together in a local network.

Existing NFS servers required no software modifications to serve PC-NFS clients. Authentication of clients was performed by a daemon, usually called `pcnfsd` or `rpc.pcnfsd`, running on one server. The clients would send an authentication request to the PC-NFS authentication server. If the authentication was successful, the server would then send back the user's Unix uid, which could be used in all subsequent NFS requests.

Two issues arose with PC-NFS software. First, the limitations in the NFS protocol itself to handle things such as file locking caused problems with many PC applications.

Windows clients often rely on advanced locking support from file servers; this expectation is quite different from those of Unix NFS clients. Samba implements full Windows NT-style semantics for CIFS file locking, share modes, and even opportunistic locking (*oplocks*).



These locking terms will be discussed is depth during Hour 8, "Samba: The File Server."

The second problem with PC-NFS clients was the lack of native support within PC operating systems (i.e., Windows). Microsoft has historically chosen the NetBIOS and SMB protocols upon which to build its network model. As a result, all modern Windows systems ship with an CIFS client as part of the OS. No additional third-party software is needed to connect to a Samba server, other than the operating system itself. To connect to an NFS server, extra client software was needed and thus an extra cost was incurred. In previous times, when hard disks were smaller, this extra bulk was an issue. The prohibitive cost nature of third party Windows-based NFS clients is still an issue for most network administrators.

Using Samba eliminates these problems. Native CIFS client support has several further advantages, one being that users are assured that the necessary client to connect to a Samba server will work correctly with new versions of the operating system. Also, on upgrading the operating system, users receive the latest network client software, whereas the PC-NFS client had to be upgraded separately, which was sometimes problematic. Perhaps one of the biggest advantages is financial. Licensing fees for NFS clients can add up quickly, especially for sites with large networks. Let's be honest—is there anyone who really likes tracking client licensing?

Today, many networks have already integrated Windows servers into their infrastructures (if they have not replaced other operating systems altogether). The traditional solution to the challenge of expanding demand for services this time is simply to add more Windows NT Servers. This is not always a cost-effective path. One solution—using Samba and one of the free PC Unix operating systems such as Linux—enables administrators to leverage the advantage of low-cost, commodity hardware while still providing the stability and services necessary for their PC clients.

A Brief Look at Supported Platforms

Samba is distributed with full source code (written in C) and is available under the GNU General Public License. If you are unfamiliar with the GPL, read the section later in this hour that covers the details.

Because it is distributed with source, it can be compiled on just about any existing Unix variant including (but not limited to):

- Solaris
- Ultrix
- Linux
- Irix
- HP-UX
- OSF1
- AIX
- NetBSD, FreeBSD, and OpenBSD
- SCO

In addition to the Unix main distribution, Samba has been successfully ported to the following operating systems:

- Amiga
- VMS
- OS/2
- MVS
- Stratus-VOS
- MPE/iX

The GNU General Public License (GPL)

Many OSS projects are released under the GNU GPL developed by the Free Software Foundation. In this sense, the word "free" refers to liberty and not price. The spirit of the GPL is to allow software to be distributed and requires that

- Any modifications to the software must be released under the same license as the original software (see section 2 of the GPL version 2).
- The source code, if not distributed with the binaries, must be available by request (see section 3 of the GPL version 2).

Without stating the specific sections, the following excerpt from the GPL Preamble explains the intent of the license:

> When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.
>
> To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software or if you modify it.

The idea behind copying and distributing free software is fairly self-explanatory. The sections about modifying software licensed under the GPL might need a little more explaining.

> 2b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

Section 2, part b of version 2 of the GPL states that any work derived from any other software licensed under the GPL must also be licensed under the GPL. By doing this, the license ensures that software released under its protection will always be free even if new changes are incorporated into existing code or the software is absorbed into another project. Once a piece of software has been released under the GPL, it is almost impossible to release it at a later time under a different license (proprietary or free).

The FSF has more information describing the philosophy behind free software and the GPL on its Web page at `http://ww.fsf.org/philosophy/`. You can also find out more information about Open Source and various Open Source licenses at `http://www.opensource.org/`.

Summary

Samba has quite a few capabilities—two of which stand out in particular:

- Samba is free.
- Samba allows Unix boxes to serve as files and printers to PC clients and vice versa.

In the following hours, you will learn how to install and configure all of Samba's specific capabilities.

Q&A

Q Can Samba fully replace my Windows NT server that is not a Primary Domain Controller (PDC)?

A Samba can completely serve files and printers to Windows, just as a Windows NT server would.

Q Can Samba replace my Windows NT PDC?

A Not completely. Samba's domain control capabilities for a Windows 9x client are solid and complete, and so these clients would probably never know the difference. The domain control support for Windows NT/2000 clients is still being developed. Currently, enough has been implemented to allow a Windows NT client to join a Samba-controlled domain, but there is more to domain control than that. The most conspicuous absence is the lack of support for Windows NT trust relationships and the SAM replication protocol used between NT PDCs and Backup Domain Controllers (BDCs). Samba's domain control capabilities are discussed more in Hours 21 and 22.

Q I didn't see my particular server's OS in the list of platforms supported by Samba. How can I find out whether Samba will compile on my server?

A If I have not listed your particular version of Unix (or other platform), you can do two things. First, simply try to compile Samba to see whether anything doesn't configure or compile. The second option is to search the `samba` mailing list archives, located at `http://lists.samba.org/`, for more information.



HOUR 2

Windows Networking Concepts

Before beginning to configure the inner workings of Samba, in this hour we examine some of the Windows networking concepts that, by necessity, Samba implements. The following sections explain the fundamental ideas behind the networking model that is supported by Microsoft operating systems, from Windows for Workgroups to Windows XP. This hour presents a broad overview of the networking protocols used by clients from Microsoft and other vendors such as Apple and IBM.

Things have changed somewhat in Windows 2000, but Microsoft goes to great lengths to make their software backwards-compatible. This means that although a homogeneous Windows 2000 and XP network can take advantage of newer technologies such as Kerberos and LDAP, these clients can use the same networking protocols used by non–Windows 2000/XP hosts.

The reason I am presenting this in the beginning of the book is because it is important to understand what Samba is attempting to do. Knowing what should be happening helps to pinpoint the problem when things break. It is the foundation of our ability to troubleshoot the server later.

NetBIOS Overview

If you have worked with Intel-based machines for any length of time, the initial BIOS screen that appears when the machine first boots is probably a familiar sight. BIOS stands for "basic input/output system." During the mid-1980s, the BIOS was extended to support rapidly emerging networks. The result was an application programming interface (API) called NetBIOS (Network basic input/output system). The generally accepted definition of the NetBIOS API at that time was the *IBM PC Network Technical Reference Manual,* published in September 1984.

Shortly thereafter, in 1985, IBM developed a network protocol to encapsulate and extend the NetBIOS API. The resulting protocol was named NetBEUI, for "NetBIOS extended user interface." NetBEUI is optimized for small LANs but is not routable beyond a single broadcast network and is therefore really usable only in small environments. NetBIOS can also run over IPX. This allowed Novell networks and Microsoft networks to coexist. In 1987, the Internet Engineering Task Force (IETF) standardized the interface over TCP and UDP in Request for Comments (RFC) 1001 and RFC 1002. NetBIOS over TCP/IP is commonly referred to as NBT.



Samba utilizes only NBT. Therefore, to connect to a Samba server, all clients must have a TCP/IP protocol stack installed in addition to the client software.



All of the Windows NT remote administration tools, such as the Server Manager and User Manager for Domains, were indirectly built on top of CIFS/SMB and NetBIOS. By providing support for the NetBIOS transport layer regardless of the underlying network layer (TCP/IP, NetBEUI, or IPX), Microsoft was attempting to gain ground in office networks by allowing its software to coexist with that of its main competitor at the time, Novell. Otherwise, many people would have been hesitant to adopt Microsoft technology if that meant that the entire network infrastructure would have to be replaced.

RFCs 1001 and 1002 define three services that are to be provided by NetBIOS over TCP/IP:

- Name service
- Session service
- Datagram service

What Is a "Name Service"?

A name service helps bridge the gap between how computers know and locate other machines and how humans know and locate them. Figure 2.1 shows that people have a tendency to remember names better than numbers, while computers require some type of numeric address. A name service provides the service of mapping names to numbers.

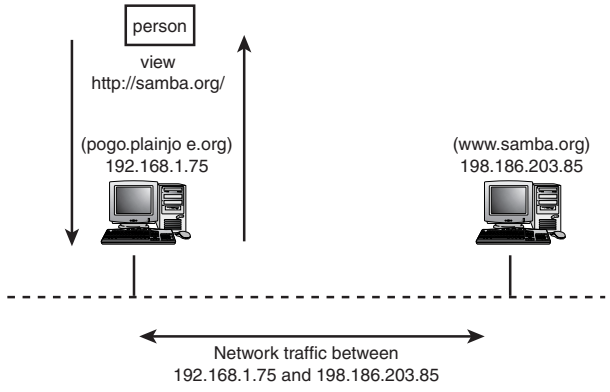


FIGURE 2.1
Understanding the differences between how humans and computers locate a host on a network.

If you are familiar with TCP/IP networking, you know that each host on a network has a hostname, such as `pogo.plainjoe.org`, and an associated IP address, such as 192.168.2.80. When you type

```
$ ping pogo.plainjoe.org
```

from another machine on the network, the machine that is issuing the ping request has to resolve the hostname `pogo.plainjoe.org` to a network address before it can send any packets to the remote computer. The most common means of resolving an Internet hostname to an IP address is either to look up the name in a local hosts file, such as `/etc/hosts`, or to query the Domain Name Service (DNS).



For more information on TCP/IP networking and DNS, refer to *DNS and Bind* by Paul Albitz and Cricket Liu (O'Reilly Publishing, 2001).

The NetBIOS name service, defined in RFCs 1001 and 1002, provides a service to NetBIOS clients similar to the one DNS provides to IP hosts—a means of associating a name with a computer. It will be easier to understand the details of the NetBIOS name service once we have explained what composes a NetBIOS name.

NetBIOS Names and Registration

NetBIOS names consist of 15 alphanumeric characters. Valid characters include the upper- and lowercase letters A to Z and a to z, the digits 0 to 9, and the following symbols:

```
! @ # $ % ^ & ( ) - ' { } . ~
```

A 16th byte containing a number from 0x00 to 0xFF is appended to the 15 characters of the name. This "resource byte" represents the type of the name and defines the service provided by the owner of the name. It may help you to think of a NetBIOS resource type as being equivalent to a TCP or a UDP port number and the actual NetBIOS name as representing the IP address. For example, a telnet client expects a telnet server to use TCP port 23. In the same way, an SMB client expects a file server to own a registered name containing a <0x20> resource byte (file share access point).

When a NetBIOS client such as CAESAR in Figure 2.2 boots onto the network, it must first claim its name among other NetBIOS hosts. Figure 2.2 illustrates how this takes place.

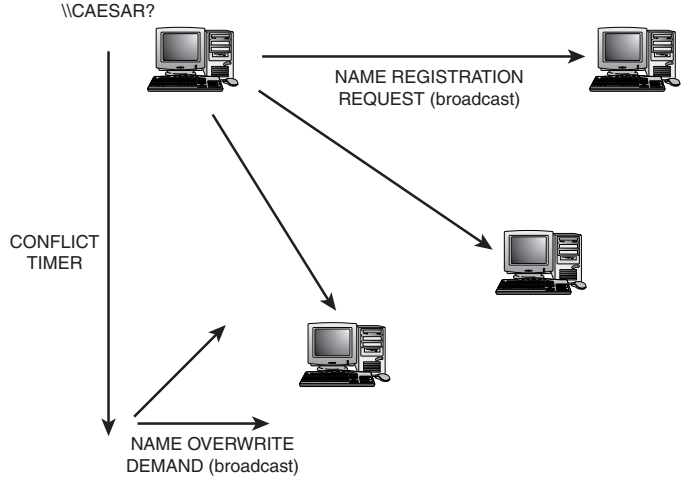


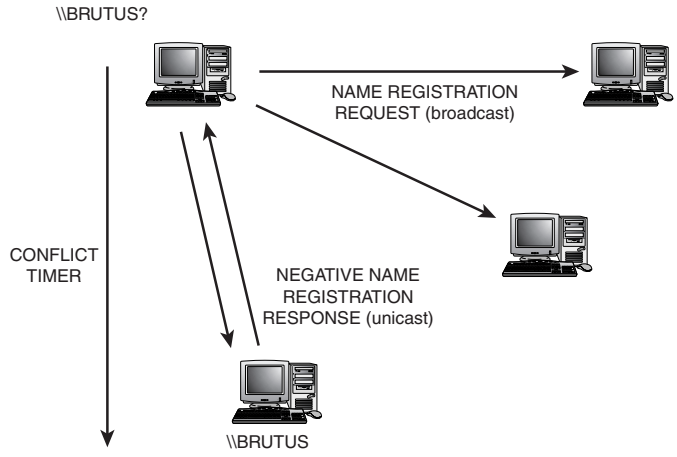
FIGURE 2.2
Using broadcasts to claim a NetBIOS name.

If no host responds that the requested name is unavailable after the amount of time defined by the value of `CONFLICT_TIMER`, CAESAR is able to claim the name as its own. The act of claiming a name is referred to as *name registration.*

Once a name is claimed, it must also be defended if another host attempts to claim it. Figure 2.3 shows a failed name registration attempt. Another host defends the name BRUTUS by responding that it already owns the name. Therefore, the requesting host is

denied the name. The most common result of this error is that the client with no NetBIOS name disables its NetBIOS interface and informs the user of the problem.

FIGURE 2.3
A rejected name registration attempt



NetBIOS names can be owned exclusively or can be shared with other hosts in some cases. User and machine names, such as JERRY<03> and MYMACHINE<0x00>, are registered and owned by a single host. An example of a shared NetBIOS name is the name of the workgroup or domain to which a host belongs.

Tables 2.1 and 2.2 list all the current NetBIOS resource tags with a short explanation of each.

TABLE 2.1 Unique NetBIOS Resource Types

Resource Byte	*Description*
`<00>`	The workstation service name, commonly referred to as the NetBIOS name
`<03>`	Messenger service name used for sending and receiving messages
`<06>`	RAS server service
`<1B>`	Domain master browser name used by a machine to contact a domain's primary domain controller (see Hour 22, "Domain Control for Windows NT 4.0/2000.")
`<1D>`	Name used by a client to access the local master browser (see Hour 19, "Local Subnet Browsing.")
`<1F>`	NetDDE service
`<20>`	Server service name to provide file-sharing access points
`<21>`	RAS client
`<BE>`	Network monitor agent
`<BF>`	Network monitor utility

TABLE 2.2 Group NetBIOS Resource Types

Resource Byte	*Description*
<1C>	A domain group that contains a list of domain controllers that have registered themselves with the NetBIOS name service (see Hour 22.)
<1E>	Normal group name used in the election of browse masters (see Hour 19.)
<20>	Internet group, used for administrative purposes
MSBROWSE	Appended to the domain name to announce the domain to other master browsers

A practical use for these resource tags is given in Listing 2.1. This output from the `nbtstat.exe` command was taken from a Windows 95 OSR2 machine, and the machine being examined was a Windows NT 4.0 workstation.

LISTING 2.1 Output from the `nbtstat.exe` Command

```
C:\WINDOWS> nbtstat -a pogo
       NetBIOS Remote Machine Name Table

   Name               Type         Status
---------------------------------------------------
POGO           <00>  UNIQUE      Registered
NARNIA         <00>  GROUP       Registered
POGO           <03>  UNIQUE      Registered
JERRY          <03>  UNIQUE      Registered
POGO           <20>  UNIQUE      Registered
NARNIA         <1E>  GROUP       Registered
NARNIA         <1D>  UNIQUE      Registered
..__MSBROWSE__.<01>  GROUP       Registered
   MAC Address = 00-60-97-40-CD-18
```

The value in the Type column combined with the resource byte helps determine what the name represents. For example, you can determine that the remote machine's name is POGO because it is a unique name with the <00> tag. The information in Table 2.1 justifies this conclusion. You also can determine from the group entries that, for browsing purposes, the machine belongs to the NARNIA workgroup. The <1E> tag is used to select

browse masters. The browsing process is discussed in more detail in the section on Windows networking models later in this hour and in Hours 19, "Local Subnet Browsing," and 20, "Cross Subnet Browsing."

NetBIOS names exist in a flat name space. In the discussion so far, the name space has been limited to those hosts that can receive the broadcast packets that are used to register a name. We will see in Hour 18, "WINS and NetBIOS Name Services," how it is possible to extend this name space beyond a single subnet by using a WINS server.

It is possible to isolate one name space from another on the same IP subnet by using a NetBIOS scope. A NetBIOS scope is a string of characters whose length plus the length of the NetBIOS name cannot exceed 256 characters. Scope does not provide any hierarchical organization to the NetBIOS name space, though. It simply isolates names in one scope from those in another. Unless you have a very good reason to set a NetBIOS scope, it is generally a good idea to simply leave it blank.

In Windows ME, you can set the scope ID by starting the Network control panel and setting the value in the WINS Configuration tab under the TCP/IP properties for your network adapter card. Figure 2.4 shows the scope ID field set to `SAMS_24`.

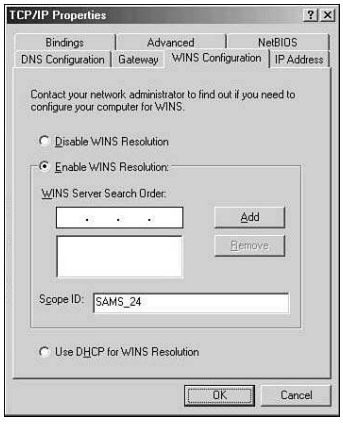


FIGURE 2.4
Setting the NetBIOS scope ID in the Windows ME TCP/IP network properties.

Name Resolution

As we have seen, the NetBIOS name service provides a method for registering client machine names. This would not be much use without a means of matching names to machines when sending traffic from one computer to another. This capability is referred to as resolving a NetBIOS name, and it plays an important part in everything from logging in to a Windows NT domain to simply browsing the Network Neighborhood.

We will continue to limit our discussion to broadcast packets for the moment, although this is not the only means available to a NetBIOS client of locating other hosts. Figure 2.5 depicts the normal process of locating a host. At the risk of oversimplifying things, the entire process is really no more complicated than someone answering a pay phone in a coffee shop and then yelling, "Hey! Is anyone named SPARTACUS here? You've got a phone call!" The calling host asks, "Is anyone here named SPARTACUS?" The owner of the name responds, "Yes, I am SPARTACUS." After this, the two hosts can begin their conversation.

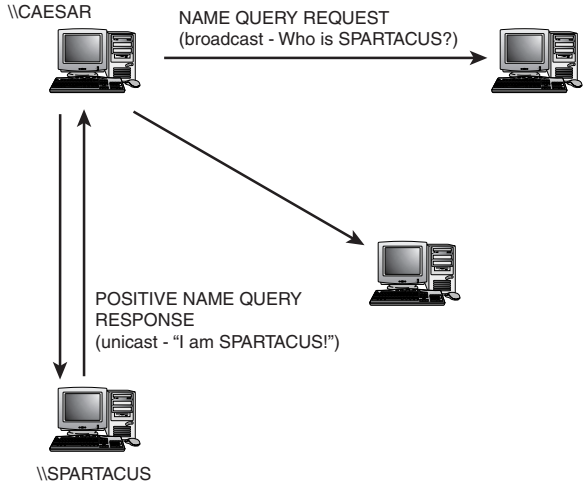


FIGURE 2.5
Using broadcasts to resolve a NetBIOS name.

Although I have presented only broadcasts as a means of registering and resolving NetBIOS name, two methods are actually available: broadcast and point-to-point. You should understand that broadcast registration (or resolution) means that the request packet is sent to all hosts on the same logical subnet with the same scope ID. Routers can be configured to forward broadcasts, but this is generally not a good idea because of the increase in traffic on the network.

Point-to-point name registration (or resolution) directs requests to a NetBIOS name server (NBNS). NBNS hosts are also described in RFCs 1001 and 1002. The NetBIOS RFCs allow the NBNS to accept varying degrees of responsibility for the management and validation of names. Microsoft's implementation of the NBNS, the Windows Internet Name Service (WINS), acts as an agent to allow clients to register and to resolve names to an IP address. Samba is also able to function as a WINS server (see Hour 18, "WINS and NetBIOS Name Services").

Broadcast and point-to-point registration and resolution create the following taxonomy:

- b-node—Nodes that use strictly broadcasts for registering and resolving names. The hosts CAESAR and BRUTUS in Figures 2.2 and 2.3 are examples of b-node clients.
- p-node—Nodes that unicast name registration and resolution requests directly to the NBNS.
- m-node—Nodes that broadcast for name registration. If the registration is successful, the NBNS is informed via unicast packets. Clients attempt to resolve names using broadcast first and unicast to the NBNS only on failure. In practice, this type of node is rarely used.
- h-node— So-called hybrid nodes were added by Microsoft after RFCs 1001 and 1002 were written and are the logical inverse of m-nodes. h-nodes use the NBNS for all name registration and resolution and degrade to broadcasts only if the NBNS is unavailable or the requests fail.

In practice, most NetBIOS clients are either b-nodes, which do not use a WINS server, or h-nodes, which use a WINS server. You can view the current node type on a Windows 2000 host using the `ipconfig.exe` command, as shown in Listing 2.2.

LISTING 2.2 Viewing the NetBIOS Node Type for a Windows 2000 Client

```
c:\WINNT> ipconfig /all
Windows 2000 IP Configuration

        Host Name . . . . . . . . . . . . : POGO-2000
        Primary DNS Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Broadcast
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : localdomain
[remaining output deleted]
```

Session Service

The NetBIOS session service provides a means of supporting reliable connections and the exchange of messages between two hosts. An astute reader will notice that this seems to be redundant when using TCP/IP, which already provides a connection-oriented protocol (TCP). This is a correct observation. However, remember that NetBIOS is an abstraction layer than can be run over multiple network-layer protocols.



The similar features provided by NetBIOS and TCP/IP are one reason that Microsoft has chosen to use TCP/IP and DNS to replace NetBIOS in the Windows 2000 operating system. However, NetBIOS is still necessary to communicate with non–Windows 2000 hosts.

The relationship between the NetBIOS name service and session service can be compared to a phone conversation. For any given connection, the sender has a *calling name,* and the receiver has a *called name.* I mention these terms only because they sometimes appear in error messages from Windows clients and some of the utilities included with Samba.

Suppose you have a new friend named Brutus and want to give him a call. What is the first step you will take? Most likely you will need to look up Brutus's phone number some where, such as in the phone book or a personal address book. But assume that Brutus has just gotten his phone connected and is not listed in the phone book yet. What then? The most common solution is to ask someone who knows the information, such as an operator or another person who knows Brutus.

Once the correct phone number has been located, you can place the call, and a circuit connecting the two of you is established to carry the voice data. This process of resolving Brutus's name to a phone number is similar to the capabilities provided by the NetBIOS name service. The NetBIOS session service is what carries the data between the called host (Brutus) and the calling host (yourself).

If you call the wrong phone number and ask for Brutus, the voice on the other end of the line will usually reply, "There is no one here by that name," or something to that effect, and then hang up (some more politely than others). This is what happens when you try to connect to a host using the wrong NetBIOS name, hence the error message, "Not listening on called name."

Datagram Service

The datagram service is a complement of the session service. It provides a connectionless service for sending packets to a specific host (unicast) or group of hosts within a workgroup (multicast) or to all hosts in a logical subnet or NetBIOS scope (broadcast). The datagram service allows for such things as locating the current local browse master for a workgroup by sending a request to the workgroup name (e.g., `WORKGROUP<1c>`).

If the NetBIOS session service can be compared to a telephone connection, the datagram service should be compared to the postal service. If you mail seven letters in one day to seven different people, there is no way of knowing who will receive his or her letter first. If you send one letter a day for seven days to a single person, there is no guarantee that the letters will be received in the order in which you mailed them. Although there is a high probability that each letter will be delivered, there is no guarantee that one will not get lost. The same is true of the connectionless datagram service.

CIFS (SMB)

CIFS stands for Common Internet File System and is often used interchangeably with the older abbreviation SMB (Server Message Block) because CIFS is simply the next incarnation of the SMB protocol.

SMB was initially defined in a joint document by Microsoft and Intel in 1987 named "Microsoft Network/OpenNET File Sharing Protocol." Since then, the protocol has gone through many changes. Recently the Storage Network Industry Association (SNIA) has put together a working group to document the most recent updates to the protocol. However, these documents tend to be advisory in nature, and the real documentation is simply "how NT does it."



You can find out more about the SNIA at `http://www.snia.org`.

SMB over NetBIOS

The SMB protocol runs on top of NetBIOS (as in Figure 2.6). As discussed in the previous section, NetBIOS can run on top of NetBEUI, IPX/SPX, and TCP/IP. Samba implements SMB only over TCP/IP.

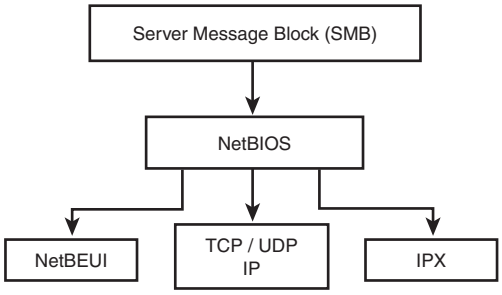


FIGURE 2.6
Hierarchy of SMB over NetBIOS and possible network-layer protocols.

Connection-Oriented Protocol

SMB is a connection-oriented protocol, meaning that all SMB packets occur within the context of virtual circuit (VC) between the client and server and are delivered in the order that they are sent. This VC is provided by the NetBIOS session service.

Consider the following four components of an SMB connection environment, shown in Figure 2.7:

- A virtual circuit (VC) between the client and the server
- A session UID (UID)
- A resource tree connection ID (TID)
- File identifiers (FIDs)

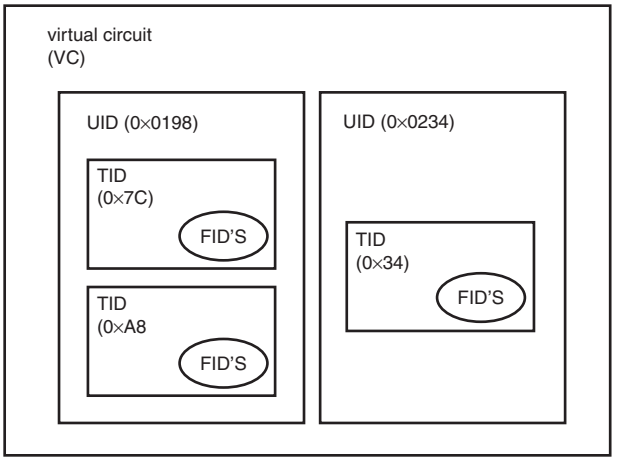


FIGURE 2.7
SMB connection environment.

If any one part of the connection environment becomes invalid, everything contained within that portion of the environment is invalidated as well. An SMB server maintains no information about past connections. If a connection terminates, everything must be rebuilt from scratch. The session UID that a client obtained from a previous connection will not work with a new connection. All tree connection IDs (TIDs) and file IDs (FIDs) must be reopened as well. Most clients silently attempt to reconnect the terminated session so that the mapped network drives and printers will be available.

Figure 2.7 shows the environment for a fictitious connection. If the UID (0x0198) terminated for some reason, both of the tree connections (TIDs), 0x7C and 0xA8, would become invalid because they are contained within the UID environment. Now suppose that the entire virtual circuit terminated. In this case, all the UIDs, TIDs, and FIDs would become invalid because they all occur within the context of the virtual circuit.

Protocol Overview

When you execute the command

```
C:\WINNT> net use h: \\SERVER\share1
```

three steps occur before you can access any files on the network share (assuming that you are successfully authenticated). Figure 2.8 depicts these three stages in creating an SMB connection between a client and a specific share on a server.

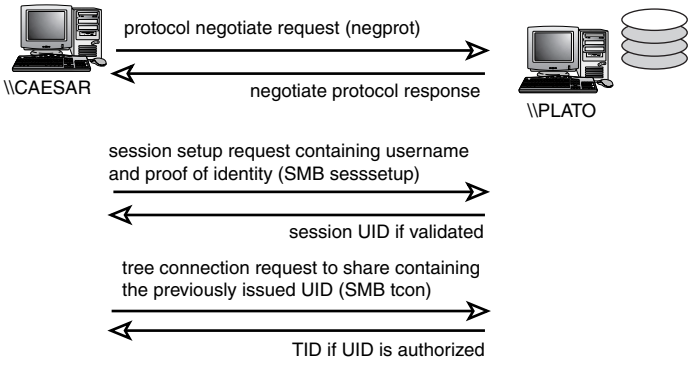


FIGURE 2.8
The three steps in establishing an SMB connection to a file share on a server operating in user-level security.

1 `negprot`. The first step in connecting to an SMB share is to negotiate the SMB protocol dialect to use. In the request packet, the client (CAESAR) sends a text listing of all the SMB dialects that it understands. Each SMB dialect (or protocol level) supports certain functionality. The server (PLATO) then selects the most advanced protocol that it knows and responds to the client, specifying the protocol number from the list. At this point, the client and server have agreed that SMB commands can be used for the remainder of the conversation.



For more information on the features provided by different SMB protocol dialects, refer to the latest CIFS specifications at the SNIA Web site.

2 `SMBsessetup`. The next step is to create a session connection between the client and server. To do this, the client issues a "session setup request," which includes a username and some proof of validity, such as a password. The server attempts to validate the requesting user. If successful, the server then returns a session UID to the client. This UID is unique for each session and has no relation to the server's internal representation of users. For example, the session UID is not the same as the user's Unix UID or an NT user's security identifier (SID). As you will learn in Hour 7, "Security Levels and Passwords," this description is not entirely accurate. The SMB protocol can operate under different modes of security that affect steps 2 and 3. However, for the moment, this is an adequate description.

3 `SMBtcon`. The final step before access to files on a remote share is allowed is for the client to make a successful tree connection to the shared resource. The client sends to the server a "tree connect" request, which includes the UID previously issued by the server. At this stage the server verifies that the authenticated user is authorized to access the requested resource. If the user has sufficient privileges to access the share, the client is issued a tree connection ID (TID). The TID is used in all requests to access files contained in the resource to which the TID refers.



Authentication is the process of proving you are who you say you are. The burden of proof is on the client. It is the job of the server to make sure that once authenticated, a client does not have access to anything that it is not permitted to use. This access control is referred to as *`authorization`*.

When these steps are complete, the user can perform operations on the share, such as opening a file or scanning a directory.

Windows Networking Models

Before personal computers, the network model was based on a large central server and terminals to allow users access to the system. These terminals had no autonomous computing power of their own and hence were named *dumb terminals*. All they provided the user was an interactive view of the server.

With the introduction of PCs in the 1980s, people began to store their files on the local hard drive on their PCs. This, however, posed a problem for sharing files, a task that was trivial when everyone logged in to the same machine (a mainframe) from a terminal. People wanted to be able to store their files locally so that they would be accessible during a server outage (over which they had no control) but still give other users access to the files from their own computers. This PC-centered distributed model is named *peer networking,* or sometimes peer-to-peer networking, because all machines are as likely to be servers as clients and could operate in both modes.

Workgroups

The idea of a workgroup goes hand in hand with the concept of peer networking. A workgroup is a unit of people who share responsibilities to achieve a common goal. Each one has to pull his or her own weight. A computer workgroup is similar, and can be used in two contexts.

The first use of a workgroup is an administrative group of machines in which each member is responsible for authenticating connection requests from clients to its local resources. Remember the `SMBsessetup` request and response of the SMB protocol overview. That is the stage at which the client sends a username and some proof of identity. Who possesses the information to determine whether or not the login name and password are valid?

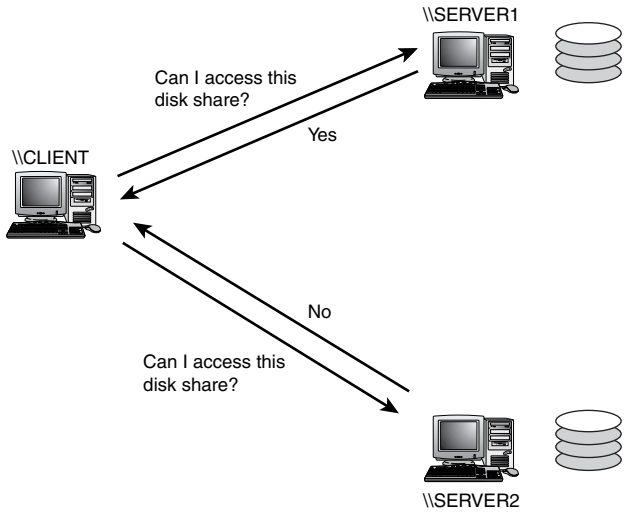
In a workgroup, each machine has a separate, local account database. Therefore, all validation is done by the machine receiving the `SMBsessetup` request. Remember that this is called peer networking because all machines are essentially equal. Each machine must pull its own weight. Each PC has the capability to serve files and printers and to validate access requests.



It may be easier to ignore Windows 9x/ME hosts as servers for the moment and to consider only Windows NT/2000 machines. We examine how a Windows 9x/ME server operates in more detail in Hour 7.

Figure 2.9 illustrates the workgroup authentication model. The client, shown on bottom, attempts to access the disk share on SERVER1. SERVER1 alone is responsible for validating the request against its local account database. When the client attempts to access the printer share on SERVER2, that server is responsible for validating the connection. The outcome is entirely separate from the outcome of the connection to SERVER1. Each server has its own account database, which is unrelated to the other server's database.

FIGURE 2.9
Accessing resources in a workgroup



The second use of a workgroup is related to network browsing and is covered more extensively in Hours 19 and 20. The motivation for network browsing is the manner in which resources appear and disappear from the network as hosts start and stop. Unlike a central computing model, such as the mainframe and terminal solution where everything is located on one machine, the distributed model makes it much more difficult to survey the large number of hosts that can come on and off the network at the whim of the PC's owner. Browsing allows users to view the current servers and resources available dynamically.

Domains

A domain is similar to a workgroup with respect to network browsing, but very different in relation to authentication and user accounts. In a domain, there is a central authentication server that maintains the user and group accounts. Regardless of the machine on which resources are located, resources in the domain are accessed by validation against the domain controller. This is still peer networking because all machines maintain the

capability to serve files and printers and perform the necessary validation. The difference is that the validation is performed against a remote account database located on the domain controller rather than a local database. Domains grew out of the need to reduce the number of passwords that are necessary when every machine has its own local account database. The domain solution provides users with one account that can allow access to all resources if desired.

FIGURE 2.10
Accessing a resource in a domain.

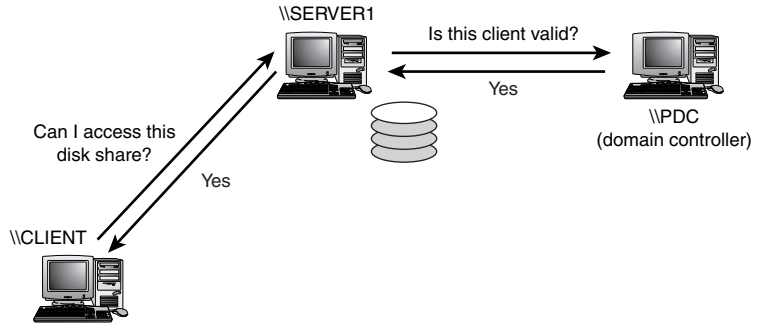


Figure 2.10 shows a connection to a server that is a member of a domain. First, the client sends a connection request containing the user information to SERVER1, asking for access to a certain disk share. SERVER1 then sends a validation request to the domain controller (DC). The validation request contains the user information originally sent by the client. If the DC successfully validates the user, it sends a positive response to SERVER1, which then sends a positive connection response back to the client. Assuming that the access control mechanisms (such as permission lists) allow it, a client can connect to any server in the domain using a single username and password. In Figure 2.9, the client needed a separate username and password to connect to each server.

Summary

The NetBIOS and CIFS/SMB protocols contain many implementation details not covered here. These are the main things to remember from this hour:

- NetBIOS offers three services to clients: a name service, a session service, and a datagram service.
- CIFS is a connection-oriented service that uses the NetBIOS session service.
- A CIFS connection requires three steps: negotiating the protocol level, establishing a session connection, and establishing a connection to a specific share.
- A workgroup name and a domain name are the same with regard to network browsing but different with regard to authentication models (see Figures 2.9 and 2.10).

We explore CIFS authentication models in more depth in Hour 7.

Q&A

Q What TCP and UDP ports does NetBIOS over TCP/IP use?

A The NBT name service uses port 137/udp, the NBT session service uses port 139/tcp, and the NBT datagram service uses port 138/udp. In Hour 17, "Security Tips," we explore port use in more detail as we look at how to protect your Samba and Windows servers using a firewall.

Q Where can I find the NetBIOS RFCs?

A RFCs 1001 and 1002 can be downloaded from the IETF's Web site (`http://www.ietf.org`).

Q Where can I find out more about the CIFS protocol?

A Microsoft has emphasized the CIFS protocol for obvious reasons. However, this should not be confused with an attempt to standardize the protocol. While many vendors besides Microsoft use CIFS in their server products, developers spend more time looking at how a Windows NT/2000 server acts than what the specifications say it should be doing. Nevertheless, there are a few locations to look for more detailed information. Microsoft maintains a collection of older CIFS/SMB documents at `ftp://ftp.microsoft.com/developr/drg/CIFS/`, and the SNIA also has documentation at `http://www.snia.org/`.

New Terms

broadcast name registration and resolution Hosts use packets destined for the local network's broadcast address (for example, `aaa.bbb.ccc.255` for a standard class C subnet) to register and resolve names.

called name The NetBIOS name of the server to contact in a connection request.

calling name The NetBIOS name of the client in a NetBIOS connection request.

domain A collection of machines on which the ultimate authentication of connection requests to any shared resource is performed by a domain controller.

domain controller A machine responsible for service authentication requests by servers in a domain.

NetBEUI NetBIOS extended user interface. NetBEUI is a network protocol designed by IBM to encapsulate and extend the functionality of the NetBIOS API.

NetBIOS Network basic input/output system, an API developed to allow programmers to write network applications.

NetBIOS scope A character string used to segment the NetBIOS name space. Hosts in one NetBIOS scope are unable to see hosts in another NetBIOS scope.

point-to-point name registration and resolution Hosts use packets destined for a single host (unicast) that acts as a central name server to register and resolve names.

workgroup A collection of machines in which each machine validates client connection requests to its own shared resources.



HOUR 3

Obtaining the Latest Release of Samba

Very often, while wading through mail on the various Samba mailing lists or newsgroups, I come across a message like this:

> "I'm running Samba version 1.9.16p4 and I can't get ____ to work."

Invariably the answer is,

> "Upgrade to the latest version, and if it still doesn't work, post your question again."

It is important to realize the rate at which code develops and changes, especially in an open source software project such as Samba. The reported problem might be a known bug that has already been fixed.

Perhaps you are installing Samba for the first time or maybe taking over for someone who previously maintained your servers. Whatever the reason, sooner or later you will need to obtain a copy of the latest Samba source code and compile it yourself. In fact, you might even find that it's something you look forward to.

In this hour we provide the information necessary to download the latest Samba source code distribution and compile the various tools yourself. We also take a look at the various precompiled binary packages that are available in case you don't feel like compiling source on your own.

Download Sites and Methods

The first place to look for the latest Samba source code release is one of the official mirror sites listed at `http://samba.org`. Once you have chosen a mirror site, select the download link from the page header. The resulting page will have a link for downloading a file named `samba-latest.tar.gz`, the most recent stable release of Samba.

If you prefer to browse and see what is available, links to directories are available on the same download page for use with Web browser or via anonymous FTP. Again, you should see a file named `samba-latest.tar.gz` in the directory listing. If not, simply look for the file `samba-#####.tar.gz` with the highest version number.

After downloading the distribution, change to a temporary directory where you won't overwrite anything and extract the source files. The directory `/usr/local/src` is often used for the purpose of compiling source code. To extract the files, you need a working version of GNU `gzip` and `tar`. If the file you download is named `samba-latest.tar.gz`, for example, the following command extracts the files for you:

```
$ gzip -dc samba-latest.tar.gz | tar xvf -
```



If `gzip` and `tar` are not installed on your system, several sites, such as `http://rpmfind.net/` for Linux, `http://www.sunfreeware.com/` for Solaris and http://eigen.ee.ualberta.ca/ for HP-UX, offer these tools in precompiled form. You can also obtain the source code packages at `http://www.gnu.org/`.

Although the directory tree might change from time to time, three directories are common to all versions thus far:

- `docs/`—This directory contains various documents, such as `man` pages, HTML files, and ASCII `HOWTO` files.
- `examples/`—This directory contains various examples for many operating systems describing different setup possibilities, primarily sample `smb.conf` files.
- `source/`—This directory contains the Samba source code tree for the distribution.

Before Upgrading an Existing Server

Before you begin compiling and installing the latest version of Samba, it is a good idea to make sure that all the important configuration files from an existing installation are safe. If you are installing a Samba server from scratch, you can skip this section and continue directly to the section "Compiling Samba." However, the same set of files that should be protected during an upgrade should be protected against a server crash. The information in this section can help you to formulate a good backup plan for your server's configuration data.

If you are unsure what version of Samba is currently installed on your server, the easiest means of getting this information is to use the `-V` switch for either the `smbd` or `nmbd` binary. These two programs together compose Samba's server implementation. The following command shows that the version of Samba installed on the server is 2.2.0. Note that `/usr/local/samba/bin/` is just one of the places where `smbd` can be installed. The `/usr/sbin/` and `/usr/local/sbin/` directories are also common locations.

```
$ /usr/local/samba/bin/smbd -V
Version 2.2.0
```


Backing Up Important Files

Samba's configuration data can be broken up into two categories: (1) configuration files such as `smb.conf` and (2) user account information such as the `smbpasswd` file or a username map.

Of course, it is always a good idea to keep a set of known working binaries from the previous installation in case the upgrade needs to be rolled back. If you are using some type of software-packaging system, such as RPMs, then just make sure that you keep a copy of a known working package. Otherwise, you need to manually back up the Samba programs and tools. We cover the Samba binaries in detail later in this chapter, so it might be a good idea to read ahead in order to familiarize yourself with these.

The majority of Samba's configuration information is maintained in a text file generally named `smb.conf`. In addition to `smb.conf`, other smaller pieces of data should be protected as well. Table 3.1 lists the most important configuration files, their purpose, and their possible locations.

TABLE 3.1 Samba Configuration Files

Filename	*Description*	*Possible Locations*
`smb.conf`	This text file contains the majority of Samba's configuration data.	`/etc/`, `/etc/samba/`, `/etc/samba.d/`, `/usr/local/samba/lib/`
`MACHINE.SID`	The workstation identifier (SID). This file is created by `smbd` at startup if it does not exist, which can cause problems if Samba is part of a Windows NT domain (or controlling one). In Samba 2.2, this information was moved to the `secrets.tdb` file.	`/etc/`, `/etc/samba/private/`, `/etc/samba.d/private`, `/usr/local/samba/private`
`secrets.tdb`	This file contains the machine trust account password when Samba is configured as a domain member server. in Samba 2.2, it also contains the information formally stored `MACHINE.SID`.	`/etc/samba/private`, `/etc/samba.d/private`, `/usr/local/samba/private`
`ntdrivers.tdb`	This database contains all the configuration data to support "Point and Print" features for Windows clients. In Samba 2.2.2, this file was replaced by three files: `ntforms.tdb`, `ntprinters.tdb`, & `ntdrivers.tdb`.	`/var/lock/samba`, `/usr/local/samba/var/locks`

Although the configuration files in Table 3.1 exist on all Samba servers, certain files containing user account information may not. The existence of these files depends on whether or not certain features of Samba have been enabled. Table 3.2 lists the Samba-specific user account files and the `smb.conf` parameters that must be enabled for the files to be used.

TABLE 3.2 Samba-Specific User Account Information

smb.conf *Parameter*	*Filename*	*Description*
`username map = <filename>`	The actual filename is given by the value of the username map parameter.	A mapping of Windows usernames to Unix usernames.
`encrypt passwords = yes` `smb passwd file = <filename>`	The default name is `smbpasswd`. Alternative names can be defined using the `smbpasswd` file parameter	The `smbpasswd` file contains the LM and NT password hashes used to support SMB password encryption.

All of these files are covered in greater detail in later hours. Tables 3.1 and 3.2 are provided simply to help you avoid trouble early on; these necessary files should not be deleted.

Compiling Samba

After obtaining the source code distribution of Samba, you must decide what nondefault features, if any, should be enabled when you compile the software.

Beginning with release 2.0, Samba began to use the GNU `autoconf` system for determining the capabilities of the host system on which it is compiled. This means that current Samba distributions can be built by executing the following three commands within the extracted `source/` subdirectory.

```
root# ./configure
root# make
root# make install
```

However, many options that may suit your particular needs are available for the `configure` script. Table 3.3 briefly describes the most common nondefault options. For a complete listing of all available options, the `configure` script supports a `--help` option, as shown here.

```
root# ./configure --help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
  --cache-file=FILE       cache test results in FILE

[remaining output deleted...]
```

TABLE 3.3 Common Nondefault Samba Configuration Options

Option	*Description*	*Hour*
`--enable-debug`	Enable debugging information when building Samba programs.	13
`--prefix=<dir>`	Define the top-level directory to use when installing the Samba files. The default is `/usr/local/samba/`.	3
`--with-acl-support`	Include support for managing file-system ACLs, such as the ones used by the Windows NT Explorer Security Tab on File Properties, via the SMB command. This requires the server's OS to support ACLs on the file system.	14
`--with-fhs`	Enable support for Samba binary placement that is compliant with the Filesystem Hierarchy Standard (FHS). A fully FHS-compliant package also requires that other flags be set.	3
`--with-pam`	Enable authentication, account, and session management via PAM.	16
`--with-pam_smbpass`	Enable building the `pam_smbpass.so` library.	16
`--with-msdfs`	Enable support for acting as a Microsoft DFS root server.	14
`--with-quota`	Enable support for interacting with server file-system quotas.	8

continues

TABLE 3.3 Continued

Option	*Description*	*Hour*
`--with-smbmount`	Enable building the user-space tools for mounting and unmounting `smbfs` file systems. This option is available only on Linux systems.	12
`--with-ssl`	Enable support for using SSL to encrypt SMB sessions.	17
`--with-syslog`	Enable writing log information via the `syslog` facility rather than the default individual log files.	4
`--with-utmp`	Enable logging user connections via the `utmp` facility.	16
`--with-vfs`	Enable support for pluggable per-share VFS modules.	24

Instead of the original installation instructions (`./configure`, `make`, `make install`), to enable logging via `syslog` and support for interacting with file-system quotas, use these instructions:

```
root# ./configure --with-syslog --with-quotas
root# make
root# make install
```



It may be helpful to read this entire chapter before beginning to compile Samba for the first time. Also expect to rebuild the Samba binaries as you learn more about the compile time options to the autoconf script.

The Filesystem Hierarchy Standard (`--with-fhs`)

A few comments should be made regarding the `--with-fhs` option. Most binary Samba packages now enable this option in the distribution. By default, all Samba programs and related configuration files are placed in the directory `/usr/local/samba/`. The `--with-fhs` option simply creates an FHS-compliant directory structure below `/usr/local/samba/`.

To place the various Samba files in a truly FHS compliant fashion, a few more options need to be defined, as shown here:

```
root# ./configure --prefix=/usr \
        --with-fhs \
        --libdir=/etc/samba \
        --with-configdir=/etc/samba \
        --localstatedir=/var \
        --with-lockdir=/var/lock/samba \
        --with-codepagedir=/usr/share/samba/codepages \
        --with-swatdir=/usr/share/swat \
```



The examples covered in this book assume that Samba has been installed in `/usr/local/samba` using the default locations for all Samba files.

What Goes Where When I Type `make install`?

The directories where `make install` puts files is not as straightforward as one might think due to the flexibility of Samba's `configure` script. Depending on which options were enabled, `make install` creates the Samba directory tree, if necessary, and copies over the binaries and other relevant files. In the default `/usr/local/samba/` directory layout, the following subdirectories exist:

- `bin/`—This the location for storing the `smbd` and `nmbd` binaries and other utilities included with Samba.
- `lib/`—This directory contains the `smb.conf` and `lmhosts` files and the `codepage` support files in a `codepages/` subdirectory.
- `var/`—This directory is empty until Samba is first run. At that time, the `smbd` and `nmbd` daemons create the lock files, shared memory files, browse list information databases, possibly the WINS databases, and Samba's log files. Under Samba 2.x, it also contains the `smbd.pid` and `nmbd.pid` files, along with the process ID of the currently running parent daemons.
- `private/`—This is the default directory for storing the `MACHINE.SID` file mentioned in Table 3.1, the `smbpasswd` file used to support SMB password encryption, and the machine trust account information used when operating as a member server in a Windows NT domain.
- `man/`—The Samba `man` pages are located in various subdirectories here. If you want the pages in your `man` page search path, you can either move the files to an existing `man` page location or add them to your `MANPATH` environment variable.
- `swat/`—This directory contains the files for the GUI `smb.conf` editor SWAT, which is discussed more in Hour 6, "SWAT and Other GUI Administration Tools."

Binary Distribution Methods

If, for some reason, you choose not to or are unable to compile the Samba source code yourself (say, for a lack of a C compiler on your system), you can download only the binaries. In Hour 1, "Introduction to Samba," I discussed the basics of the GPL. One stipulation of the license is that the source code must be available to those who want it, but this does not mean that it has to be distributed hand-in-hand with binary releases.

During my first job as a network administrator, I was flying by the seat of my pants. I had been given the responsibility of building a student-accessible PC lab. One of the things I purchased for the lab was a Sparc Ultra 1 running Solaris 2.5.1. Imagine my surprise when, as I was getting ready to install software, I realized that Sun didn't ship a C compiler with Solaris 2!

Binary distributions can be very helpful, depending on your needs. If you do not plan to modify the compile-time defaults or do not have a particularly unusual site, downloading the binaries can probably save you some time, and current packaging tools make software upgrades much easier to track. As many as half of the downloads of the latest Samba 2.2 release from the main Web and FTP server at `samba.org` were of some form of a precompiled package.

Obtaining a Samba binary release is very similar to obtaining a source code release. The first place to look is the Samba home page (`http://samba.org`); select the FTP or HTTP site closest to you. If you use HTTP to download the files, follow the download link from the Samba mirror home page and look for information on downloading binary packages. If you use FTP, look for a directory named `/pub/samba/bin-pkgs/` or `/pub/samba/Binary_Packages/`, and then select your operating system.

Binary packages are not available for all platforms on which Samba can be compiled, nor are they always available for the latest source code release. The reason is that volunteers compile and upload the packages. If possible, the binaries and associated files, such as sample `smb.conf` files, are archived using the tools native for the OS. For example, Red Hat binaries are stored using RPM, and Solaris binaries are distributed in the `pkgtool` format. The details of the package distribution tools for various operating systems are beyond the scope of this book. If you need more information on the binary distribution tool for your operating system (for example, `pkgadd` on Solaris 2.x, `rpm` on Red Hat Linux, or `dpkg` on Debian GNU/Linux), refer to the `man` pages for the installation tool used on your system.

Another source of precompiled Samba packages is your OS vendor. Previously, this could be said only about Linux vendors. However, this is now also true for commercial Unix vendors, such as SGI, IBM, and HP.

Now that you have the current version of Samba and the binaries are ready to go, Hour 4, "Starting Your Feet to Dance," helps you get a sample installation up and running.

Summary

You have two options for installing Samba, either download the source code and compile it yourself, or obtain a binary-only release if it is available for your platform. Samba 2.2 has many new configuration options for compiling from source. When you upgrade an existing Samba installation, it is important to make sure that certain files, listed in Tables 3.1 and 3.2, are maintained across the upgrade.

In its simplest form, compiling Samba is a process of

```
root# ./configure
root# make
root# make install
```

Using a default installation, all of the Samba binaries and related files are stored in `/usr/local/samba/`. However, vendors may choose do place things in different locations for their binary packages.

Q&A

Q I downloaded the source distribution for Samba 2.2.x. I extracted the files and typed `make`, but I get this error message: "Fatal error: No arguments to build."

A You must run the `configure` script before executing `make`.



HOUR 4

Starting Your Feet to Dance

Experience is often the best teacher. To begin our discussion of configuring Samba, we look at a simple server that provides a single file share to members of a Unix group.

It is important to be able to picture the types of environments and circumstances where Samba can offer the most advantages. In future hours we cover all of the configuration file options in depth; this hour covers only enough to get a working server up and running. Don't think that the work done here will be thrown away at the end of the hour. Future hours build on what we accomplish here.

Which Binaries?

Remember from Hour 3, "Obtaining the Latest Release of Samba," that the server portion of Samba is composed of two main binaries. The `smbd` daemon implements the CIFS server and handles the file and print service

requests from clients, while `nmbd` provides the underlying NetBIOS services required by `smbd`. Both of these daemons read a common file for configuration information. Traditionally, this file has been named `smb.conf`. However, you will soon see that it is possible to override this name using a command-line switch when you start the Samba daemons.



I refer to Samba's configuration as `smb.conf` throughout this book. This name has no special significance other than tradition, but it is used by all precompiled Samba distributions.

Samba's Configuration File: `smb.conf`

Samba's configuration file contains plain ASCII text, which makes it easy to edit. Its layout resembles older Windows `*.INI` files. There are a few basic things to remember about its format.

- Comments begin with either a hashmark (#) or a semicolon (;) and continue until the end of the line.
- The backslash (\) character can be used to indicate line continuation.
- Sections begin with a section header denoted by a name enclosed in square brackets ([]) and continue until the next section.
- Configuration settings are of the form <*parameter*> = <*value*>.

Three built-in section names are supported by Samba, in addition to any services that you may have defined.

- `[global]`—Contains settings that determine Samba's overall behavior.
- `[homes]`—A default share for providing a home directory for all CIFS users. This section will be covered in Hour 8, "Samba—The File Server."
- `[printers]`—A default share for exporting all printers on the host via CIFS. This section will be covered in Hour 9, "Samba—The Print Server."



The `[homes]` and `[printers]` shares are not available if they have not been defined in `smb.conf`.

Following is an example configuration file for a Samba server that exports no directories or printers.

```
#
# smb.conf global settings
#
[global]
   ; define the servers netbios name
   netbios name = POGO
   ; define the workgroup membership for browsing purposes
   workgroup = STY-SAMBA
```

While this may appear to be so simple as to be useless, a short configuration file like this can sometimes be very helpful when you are attempting to stabilize network browsing in your environment.



Network browsing is covered in Hours 19 and 20.

Configuring the `[global]` Section of `smb.conf`

The problem stated at the beginning of this hour is to create a Samba server that provides a single file share for members of a group. Before actually configuring the required network share, first we must configure some initial global settings to get Samba up and running.



CIFS shared resources, whether they are directories or printers, are often referred to as *shares.* This is the equivalent of an NFS-exported directory or remote printer made available via `lpr`. Sometimes a CIFS share is also called a *service.* I use the terms interchangeably.

To begin, in the previous section we answered the following two questions in the sample `smb.conf` file:

- What will be the NetBIOS name of the Samba server?
- Of which workgroup will the Samba server be a member?

In this section, the server's NetBIOS name is POGO and the workgroup of which it will be a member is named STY-SAMBA. If a `netbios name` is not defined, Samba will use the IP hostname of the server by default. If a `workgroup` is not specified, Samba will use the compile-time default value, which is normally "WORKGROUP". For the moment, these two values in the `smb.conf` can be considered arbitrary. Later hours provide more information to help you determine appropriate settings for your site.

Our working `smb.conf` is listed again here (without the comments):

```
[global]
   netbios name = POGO
   workgroup = STY-SAMBA
```

Once we have decided on the server's host and workgroup names, the next choice is which security level to use. We will use user-level security here because it is the most intuitive model for Unix systems (and because we have not discussed the other models yet). User-level security, presented in Hour 2 in the section on the CIFS protocol, requires both a username and some proof of identity for authentication, just as standard Unix services do.

The following line should be added to our working `smb.conf`:

```
   security = user
```



User-level security is the default setting for 2.0 and later releases of Samba. Versions prior to 2.0 used `security = share` by default.

We will use the default installation location, `/usr/local/samba/`, for the Samba files and programs. The file `smb.conf` should be placed in the `lib/` directory below the installation root directory (that is, in `/usr/local/samba/lib/`). Use your favorite text editor to create the file and add the settings discussed so far.

Setting Up the Shared Group Directory

It is always a good idea to make sure that you can describe what you are doing in a natural language before attempting to implement it in a programming language or configuration file syntax. This helps to ensure that you have an understanding of what you are trying to accomplish.

Here is a quick description of the goal of our Samba server:

> All members of the Unix group "staff" should be able to read and write files stored in the Samba share `[staff-files]`.

This is a trivial feature to implement using Unix file permissions. Our main concern is to make sure that Samba also supports the required functionality.

Creating the Directory on Disk

Before defining the new file service in `smb.conf`, we set up the directory on the disk that will be shared. Let's assume that the following entries exist in `/etc/passwd` and `/etc/group` (or whatever file or network directory your system uses for user and group accounts).

```
root# cat /etc/passwd
...
fred:x:1000:1001:Bill's account:/home/bill:/bin/bash
velma:x:1000:1000:Jane's account:/home/jane:/bin/bash
scooby:x:1000:1001:Scooby-Dooby-Doo:/home/scooby:/bin/bash

root# cat /etc/group
...
staff:x:1000:scooby
users:x:1001:
```

According to our description of the goal, `velma` and `scooby` should have read-write access to the `[staff-files]` share because they are both members of the `staff` group. However, `fred` is to be left out in the cold because he belongs only to the `users` group.

First we create the directory on disk. The `/export/` directory is commonly used for shared file systems on Unix hosts, so we use that as our root and create a subdirectory named `staff` with the following command:

```
root# mkdir -p /export/staff
```



The `mkdir -p <dir>` creates the entire directory path, including the upper directories, if they do not exist.

The next step is to set the correct group ownership of the directory and the necessary permissions.

```
root# chgrp staff /export/staff
root# chmod 770 /export/staff
```

Adding the Group Share to `smb.conf`

Now we are ready to define the Samba file service in `smb.conf`. First we must create the new section named `[staff-files]` and define the path to the directory to be shared by adding the following lines to our existing `smb.conf`:

```
# new file share for the staff group
[staff-files]
   path = /export/staff
```

By default, Samba does not allow write access to file shares, so we must also disable the `read only` parameter.

```
   read only = no
```

Samba does not allow more liberal access than the file system itself provides. This means that only members of the `staff` group will be able to connect to this new share. If you are curious about why this is the case, consider what would happen if a user who is not a member of this group attempted to change to the `/export/staff/` directory. The user would be denied access because the read (`r`) and execute (`x`) bits have not been set in the permission set for world access. The same thing occurs when a user attempts to connect to the `[staff-files]` share if he or she does not have the appropriate group membership.

At this point we have a working file share that meets our original requirements. Pretty simple, huh? However, there are two problems that can arise as the share is used over time.

- What happens when a new file or directory is created and is owned by a group other than `staff`?
- What if permissions are assigned to newly created files or directories that do not allow write access to members of the staff group?

To solve the first problem, we can specify that all file access in the share be done using the `staff` group. The `force group` parameter accepts a single Unix group name and ensures that all newly created files are owned by that group. By default, Samba uses the primary group of the connected user. All that is necessary is to add the line

```
   force group = staff
```

to the `[staff-files]` service in the `smb.conf` file. The order of unique parameters within a section does not matter, so we can simply add the new line to the end of the `[staff-files]` section.

It does us no good for all the files and directories to be owned by `staff` if the group permissions do not allow us the desired read-write access. Two new parameters must be added, one for files and one for directories, to ensure that the group bits are set correctly.

```
force create mode = 0060
force directory mode = 0070
```

Both the `force create mode` and `force directory mode` take an octal value as a permission mode, which determines which bits must be set in the final permission set. Our settings here ensure that the group `r` and `w` bits are set on files and that the group `r`, `w`, and `x` bits are set on directories.

What if we also want to enforce that no permissions bits for world access are ever set? Samba possesses a pair of parameters that complement the `force create mode` option. The `create mask` and `directory mask` settings are used to mask bits from the final permission mode. Any bit not set in the mask will be removed from the result mode. For our purposes, the `rwx` bits for world should be masked out, while the `rwx` user and group bits may or may not be set, depending on the file or directory. The following two lines should be added to prevent items from being created with a mode that allows access by someone other than the owner or a member of the group that owns the object.

```
create mask = 0770
directory mask = 0770
```

This time we truly do have a working file share that meets our initial requirements as well as those for long-term use.

Verifying `smb.conf`

Editing `smb.conf` by hand can often result in mistakes and misspellings. Samba developers therefore include a tool that verifies the syntax of a configuration file and points out any errors. By default, `testparm` is located in the `bin/` subdirectory of your Samba installation root. This utility checks `smb.conf` and prints out any errors that it finds and all the default values that are not overridden. It can be very helpful to make sure that Samba is seeing what you think it should see.

I have purposely misspelled `netbios` in the `[global]` section of the following `smb.conf` file.

```
[global]
  netbis name = POGO
  workgroup = STY-SAMBA
  security = user
```

The file is then checked using the `testparm` tool. The `-s` switch is used to point `testparm` to a particular configuration file, as opposed to the one stored in the default location of `/usr/local/samba/lib/`.

```
root# /usr/local/samba/bin/testparm -s smb.conf
Load smb config files from smb.conf
Unknown parameter encountered: "netbis name"
Ignoring unknown parameter "netbis name"
Processing section "[staff-files]"
<...remaining output deleted...>
```

Notice that the output reports an unknown parameter: `netbis name`. Once the error is corrected and `testparm` reports no other errors, we are ready to start and test our new Samba server.

Starting `smbd` and `nmbd`

After obtaining the Samba binaries (by compiling the source code or downloading precompiled binaries) and completing the `smb.conf` file, the next step is to start the Samba daemons, `smbd` and `nmbd`. There are two methods of launching the processes. Which you choose depends on how many connections you expect to have to the Samba server, how frequent they will be, and how many resources you presently have to spare on the server.

Each client connection has its own `smbd` daemon; however, the `smbd` process responsible for a client can maintain many connections to shares. For more information on how to gauge the amount of resources potentially required by a Samba server, refer to Hour 23, "Capacity Planning and System Tuning."

The two means of starting the Samba server processes are through the `inetd` metadaemon or as stand-alone daemons. My experience has been that running `smbd` and `nmbd` as stand-alone daemons generally provides faster service on initial connections. However, if the server sits idle for long periods of time without any SMB connections, you can choose to run `smbd` and `nmbd` from `inetd.conf` to save a little on the overall system memory usage. You need to decide which is best for you because you must choose only one method.



Running the Samba daemons as stand-alone daemons is the recommended method unless you have a particular need to launch `smbd` and `nmbd` from `inetd`.

Starting from `inetd`

To run Samba from the `inetd` daemon, you must edit two files. We assume that the Unix server is not using NIS/NIS+, LDAP, or a similar service to distribute system files.

First, add the following entries in /etc/services. Make sure that there are no other entries for TCP port 139 and UDP port 137.

```
netbios-ssn 139/tcp
netbios-ns 137/udp
```

Next, add the following entries to /etc/inetd.conf:

```
netbios-ssn stream tcp nowait root /usr/local/samba/bin/smbd smbd
netbios-ns dgram udp wait root /usr/local/samba/bin/nmbd nmbd
```

After saving the changes, the inetd daemon must be instructed to reread its configuration file. Usually, this is accomplished by sending a HUP signal to the inetd process. The following set of commands works for most systems.

```
root# ps -ef | grep inetd
root 656 1 0 Jun05 ? 00:00:00 inetd

root# kill -HUP 656
```

Starting from xinetd

The xinetd metadaemon is becoming a popular replacement for the traditional inetd server due to its more secure means of providing services. It is now shipped, at least as an optional component, in many modern Linux distributions. More information on xinetd can be found at http://www.synack.net/xinetd/.

To start the Samba services from the xinetd daemon, make sure that the netbios-ssn and netbios-ns entries exist in the system's /etc/services file, as described in the previous section on starting Samba with inetd.

The xinetd daemon can use one of two configuration file formats. The first is a single file, normally named /etc/xinetd.conf, containing all the entries for installed services. In the second format, each service's configuration information is placed in a separate file in a directory named /etc/xinet.d/. In either model, the xinetd entry for smbd appears as follows:

```
service netbios-ssn
{
   socket_type = stream
   protocol = tcp
   wait = no
   user = root
   server = /usr/local/samba/bin/smbd
   disable = no
}
```

The entry for nmbd is as follows:

```
service netbios-ns
{
   socket_type = dgram
   protocol = udp
   wait = yes
   user = root
   server = /usr/local/samba/bin/nmbd
   disable = no
}
```

If you choose to use the /etc/xinet.d/<*service file*> format, the service definitions for smbd and nmbd remain the same. However, the service files for smbd and nmbd should be named /etc/xinetd.d/netbios-ssn and /etc/xinetd.d/netbios-ns, respectively.

Finally, if you are using a single xinetd.conf file, it will be necessary to force the xinetd daemon to reread its configuration information by sending it either a USR1 signal for a soft reset or a USR2 signal for a hard reset.

Running as a Daemon

The most commonly recommended method of starting the Samba server processes is to use the -D switch when launching the daemons from a command shell, for example,

```
root# smbd -D; nmbd -D
```

In this mode, smbd and nmbd detach from the current terminal and remain running in the background. Whenever a new connection request is made, smbd forks a copy of itself to handle the session.



Whenever smbd or nmbd is run from a shell prompt, the -D argument is assumed in the absence of any conflicting options (such as -V to print the version number and exit).

Modern Unix variants generally use one of two models to start processes at boot time. The most common of these is System V init scripts, which start and stop processes based on the run level of the system. The samba.init script contained in the packaging/Example/ subdirectory should work on most systems.

If you are installing Samba on a system that uses BSD-style startup files such as /etc/rc.S, /etc/rc.M, and /etc/rc.local, simply add the following lines to the rc.local script.

```
SAMBA=/usr/local/samba
```

```
if [ -x $SAMBA/bin/smbd -a -f $SAMBA/lib/smb.conf ]; then
   echo "Starting Samba..."
   /usr/local/samba/bin/smbd -D
   /usr/local/samba/bin/nmbd -D
fi
```



For a more complete description of the Unix booting process and run levels, refer to Evi Nemeth's "Unix System Administration Handbook" (Prentice Hall) or "Essential System Administration" by Aeleen Frisch (O'Reilly).

Command-Line Arguments

It frequently is helpful to temporarily (or permanently) override a compile-time default setting using the available command-line switches for `smbd` and `nmbd`. This section briefly covers some of the common options available, but the best reference for current options are the `smbd(8)` and `nmbd(8)` man pages.

Normally, these are installed in `/usr/local/samba/man/`. To view the man pages, locate the files `smbd.8` and `nmbd.8`, and use the `nroff` command to display them:

```
nroff -man smbd.8 | more
```

The location of these man pages can be added to the `MANPATH` environment variable so that it is searched by the standard `man` command. The following command should work for all Bourne-compatible shells such as `bash`.

```
export MANPATH=$MANPATH:/usr/local/samba/man
```

Now the `smbd(8)` man pages can be viewed by typing

```
$ man smbd
```

Table 4.1 lists some common options for `smbd` and `nmbd`. For a complete list, please read the `smbd` or `nmbd` man pages.

TABLE 4.1 Common `smbd` and `nmbd` Command-Line Arguments

Option	*Description*
`-D`	Runs as a daemon, meaning that the process runs in the background, servicing requests. This is the preferred way to run `smbd` for a file server that does more than reply to infrequent requests. This option is assumed when `smbd` or `nmbd` is executed from a shell prompt.
`-d` *`debuglevel`*	Specifies the debug level (also called the log level) at which the process should run. The debug level is an integer between 0 and 10.

continues

TABLE 4.1 Continued

Option	*Description*
`-l log file`	Path to the file where the process should write log entries.
`-s config file`	Path to the configuration file that the process should use.

Testing the Installation

So far we have

- compiled binaries for Samba,
- correctly set up the directory that will be shared,
- entered settings for a complete `smb.conf` configuration file and verified it using `testparm`, and
- started `smbd` and `nmbd` as daemons.

The time has come to start the Samba daemons and verify that everything works correctly. During these tests, I highlight some common errors, how to detect them, and how to correct them.

Samba includes a utility named `smbclient` that provides an FTP-like interface for accessing CIFS servers. In fact, the original use for `smbclient` was as a testing tool for Samba that enabled users to check the server without locating extra PCs. Although it cannot guarantee that there are no glitches when the PCs are connected, it does locate obvious errors. Just as `smbclient` can be used to test the server's CIFS configuration, the `nmblookup` utility can be used to verify the correctness of Samba's NetBIOS services.

First, `smbd` and `nmbd` should be started like this:

```
root# /usr/local/samba/bin/smbd -D
root# /usr/local/samba/bin/nmbd -D
```

Next, verify that the daemons are running by executing

```
$ ps -ef | grep mbd
root 21247 1 0 Jun08 ? 00:00:00 /usr/local/samba/bin/smbd -D
root 21249 1 0 Jun08 ? 00:00:03 /usr/local/samba/bin/nmbd -D
```

The first test is to ensure that `nmbd` is functioning correctly. The following command should return the IP address of the Samba server:

```
$ bin/nmblookup -B POGO __SAMBA__
INFO: Debug class all level = 1 (pid 10011 from pid 10011)
querying __SAMBA__ on 192.168.1.75
192.168.1.75 __SAMBA__<00>
```

If you receive an error message like this

```
name_query failed to find name __SAMBA__
```

then verify again that `nmbd` is actually running and that the name used in the lookup is the same as the value of the `netbios name` parameter in `smb.conf`.

Next, use `smbclient` to display the list of shares on your server. This verifies that `smbd` is running and functioning correctly. The `-L` option instructs `smbclient` to enumerate the shares on the server rather than actually connecting to one. The `-N` switch instructs `smbclient` to use an anonymous login rather than the login name of the current user.

```
$ smbclient -L pogo -N
added interface ip=192.168.1.75 bcast=192.168.1.255 nmask=255.255.255.0
Anonymous login successful
Domain=[STY-SAMBA] OS=[Unix] Server=[Samba 2.2.2]

 Sharename   Type   Comment
 ---------   ----   -------
 staff-files  Disk
 IPC$         IPC   IPC Service (Samba 2.2.2)
 ADMIN$       Disk  IPC Service (Samba 2.2.2)

 Server     Comment
 ---------  -------
 POGO       Samba 2.2.2

 Workgroup  Master
 ---------  -------
 STY-SAMBA  POGO
```

Two additional shares appear in the listing, both with the comment `IPC Service`. These are special shares provided by Samba to support certain operations from CIFS clients. For the moment, ignore the information regarding the workgroup and master.

If `smbclient` returns an error message indicating "Connection refused," verify again that the `smbd` daemon is running. If you only receive the error message "Connection to *<hostname>* failed," then make sure you are using the correct NetBIOS name.

Once you are confident that `smbd` is operating properly, you can test the `[staff-files]` share. The general `smbclient` format for connecting to a share is

```
$ smbclient //<server>/<share> -U <username>
```

In the absence of the `-U` switch, `smbclient` uses the login name of the current user.

To connect to the `[staff-files]` share as the user `velma`, you must run `smbclient` with the following parameters:

```
$ smbclient //pogo/staff-files -U velma
added interface ip=192.168.1.75 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[STY-SAMBA] OS=[Unix] Server=[Samba 2.2.2]
smb: \>
```

If Samba successfully authenticates the user and grants the connection, you will be greeted by an `smb: \>` prompt. However, if you are met with the error message `ERRnosuchshare`, make sure that

- the share name is typed correctly
- the file system permissions allow you to access the file share as that connected user.

Assuming that you have successfully connected with `smbclient` to the new file share on your Samba host, you can upload a file to make sure that the permissions and group ownership are set correctly. In the following example, we connect as the user `scooby` because `staff` is one of his secondary groups. By doing this, we can test the behavior of the `force group` parameter. This set of commands uploads a copy of the local `/etc/hosts` file and verifies its ownership and permissions.

```
$ smbclient //pogo/staff-files -U scooby
added interface ip=192.168.1.75 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[STY-SAMBA] OS=[Unix] Server=[Samba 2.2.2]
smb: \>

smb: \> lcd /etc
the local directory is now /etc

smb: \> put hosts
putting file hosts as \hosts (27.1 kb/s) (average 27.1 kb/s)

smb: \> quit

$ ls -l /usr/export/staff/hosts
-rwxrw---- 1 scooby staff 333 Jun 10 00:16 /usr/export/staff/hosts
```

As expected, the newly uploaded `hosts` file is owned by the `staff` group and has the read and write bits set for the group mode.

A Note Regarding Windows Clients and Our Server

We have tested our server using `smbclient`. Very few users I know use `smbclient` on a daily basis. If this is so, then why did we not use a Windows client to test our server?

Microsoft clients have several quirks that complicate Samba configurations. It is not that Samba is unable to properly communicate with Windows. It is just that we have not been exposed to enough of CIFS and Samba's parameters to properly address those issues yet. All of the details surrounding Windows configurations and Samba servers will be covered in Hours 10 and 11. For now, it is enough to know that our Server works correctly with `smbclient`.

Other Tools Included with Samba

In addition to the programs already mentioned, the Samba distribution includes several other utilities described in Tables 4.2 through 4.4. By default, these tools are installed in `/usr/local/samba/bin/`.

TABLE 4.2 Administrative Tools Included with Samba

Name	*Description*
`smbcontrol`	A program that allows an administrator to send and receive messages to `smbd` and `nmbd` processes, such as querying or setting debug levels and instructing `nmbd` to force a browsing election.
`smbpasswd`	A tool for managing user and password entries in Samba's `smbpasswd(5)` file. This tool is covered in Hour 7, "Security Levels and Passwords."
`swat`	Samba's Web administration tool for managing the `smb.conf` file, connections, and users.
`make_codepage`, `make_unicodemap`	Tools for compiling or decompiling `codepage` files and Unicode map files for use with the internationalization features of Samba.

TABLE 4.3 Diagnostic Tools Included with Samba

Name	*Description*
`nmblookup`	A utility for querying NetBIOS name services over TCP/IP.
`smbstatus`	A tool that reports the current status of connections and locked files.
`testparm`	A simple syntax checker for Samba's configuration file, which also displays all the default parameter values seen by `smbd` and `nmbd`.
`Testprns`	A simple utility to verify whether a printer name is valid for use as a service name, that is, whether the printer name can be found in the specified `printcap name`.

TABLE 4.4 Client Tools Included with Samba

Name	*Description*
`rpcclient`	A command-line tool for executing MSRPC functions on Windows NT and Samba servers.
`smbclient`	An FTP-like tool for accessing CIFS shares on remote servers.

The `smbstatus` tool can be so helpful that I would like to take a few moments to introduce it to you before concluding this hour.

The `smbd` daemon for a connection runs as `root` until it is necessary to perform some operation on behalf of the connected user. At that time, the process `uid` becomes the user's Unix `uid` and then switches back to `root`. This can result in some difficulty in determining which `smbd` process belongs to which user. The `smbstatus` utility displays information about connected users and currently locked files.

The following output shows that I'm connected to my home directory and currently have the file `296204.doc` open. The meaning of the lock descriptions is covered in Hour 8, "Samba—The File Server."

```
$ smbstatus

Samba version 2.2.2
Service uid   gid   pid  machine
----------------------------------------------
jerry   jerry users 1465 pogo-2000 (192.168.215.129) Sun Jun 10 22:39:48 2001

Locked files:
Pid  DenyMode  R/W  Oplock          Name
--------------------------------------------------
1465 DENY_NONE RDWR EXCLUSIVE+BATCH /home/jerry/ch4/296204.doc Sun Jun 10
23:18:07 2001
```

Summary

Samba can effectively act as a file and printer server for PCs in a group environment. Samba relies on the underlying operating system to control access through means such as file permissions. All of Samba's configuration directives for `smbd` and `nmbd` are stored in a text file, commonly called `smb.conf`, which is shared by both daemons.

The `smb.conf` file is composed of sections that begin with a section header name in square brackets. The `[global]` section stores parameters that define Samba's overall behavior; administrator-defined file shares are contained in separate, individual sections.

Q&A

Q **Is it possible for Samba to share file systems that have been mounted using NFS?**

A Yes. However, this can be problematic if the NFS server that provides the file system fails, causing the Samba server to hang. It is always safer to use Samba to share a local file system.

Q **How many simultaneous connections can a Samba server support?**

A In theory, there is no limit. In practice, the limit is determined by the server's hardware, specifically the total amount of available RAM and the CPU power. It might also depend on the amount of activity from the `smbd` processes. More on this will be discussed in Hour 23.

New Terms

share A resource, such as a directory or a printer, that has been made available via the SMB protocol to remote machines across a network. Sometimes referred to as a service.



PART II
Basic Configuration

Hour

5 Exploring Samba's `smb.conf` File

6 SWAT and Other GUI Administration Tools

7 Security Levels and Passwords

8 Samba—The File Server

9 Samba—The Print Server



HOUR 5

Exploring Samba's `smb.conf` File

At the heart of Samba lies its `smb.comp` file. This configuration file is used by both `smbd` and `nmbd` as well as many of the other utilities included in the Samba suite. Despite the fact that `smb.conf` has more parameters than Godzilla has teeth, grouping these by functionality provides a simple means of presenting the file in a smaller, lizard like size.

This hour will provide our first in-depth look at `smb.conf`. We'll explore the general layout of the file, some global parameters that control the overall behavior of Samba, and some variables that are available for use at run time.

Layout

A standard `smb.conf` file is an ASCII text file logically divided by section headings, which are denoted by enclosing brackets (`[ ]`). Each section contains parameters and corresponding values that are separated by the `=` character. Listing 5.1 illustrates a sample `smb.conf` divided into two sections—one for the relevant global settings and one for the file share named `[my_share]`.

LISTING 5.1 A Sample `smb.conf` Containing Two Sections

```
# Samba smb.conf file
# written for STY Samba 2ed

# global parameter
[global]
   netbios name = POGO
   workgroup = STY-SAMBA
   server string = Example Samba Server [%v]

# a simple file share
[my_share]
   path = /export/u1

# end of smb.conf
```

The section names are not case sensitive and therefore a section named `[my_share]` is the same as one named `[My_Share]` or `[MY_SHARE]`. Parameter names, in addition to not being case sensitive like section names, are whitespace insensitive. The `netbios name` parameter in Listing 5.1 could have been written as `netbiosname` or `NetBIOS Name`.

Values that appear on the right side of the equal sign (=) do not follow such a consistent set of rules. The case (or whitespace) sensitivity of a value depends upon the rules for the particular parameter. For example, the values of the `netbios name` and `workgroup` setting are not case sensitive, but values that contain usernames or directory paths often are. For the remainder of this book, we will follow the convention of only making the distinction when a value is case sensitive.

Samba's `smb.conf` provides two more built-in section names besides [`global`], which were already alluded to in Hour 4, and three reserved share names. These other sections are

- `[homes]`—This special built-in section is a simple means of providing CIFS access to user home directories defined in the local system password file (for example, `/etc/passwd`). This share is covered in Hour 8, "Samba—The File Server."
- `[printers]`— This special built-in section provides a means of sharing all printers in the lpd printcap file to CIFS clients. The `[printers]` service is explained in more detail in Hour 9, "Samba—The Print Server."
- `[print$]`—This is a normal file share that is used by Samba as the location for storing printer driver files that Windows clients can download and install as needed. The `[print$]` share is also covered in Hour 9.
- `[IPC$]` and `[ADMIN$]`—These two internal shares, created by Samba at runtime, are used for supporting such CIFS features as such as named pipes and remote administration commands. These can never be defined in `smb.conf`.

The remainder of this hour is dedicated to the [global] section and related parameters.

The [global] Section

The [global] section contains parameters that relate to the overall functionality of the server. The netbios name and workgroup parameters, which were introduced in Hour 4, "Starting Your Feet to Dance," are examples of [global] parameters.

Returning again to Listing 5.1, we can explain the meaning of the server string parameter. When a user on a Windows client views the properties of a Samba server, the test string assigned as the value of for server string will be displayed as the server *comment*. Figure 5.1 shows the server properties dialog window displayed on a Windows 2000 Professional Client. Notice how the displayed comment also matches the server string parameter value in our smb.conf file.

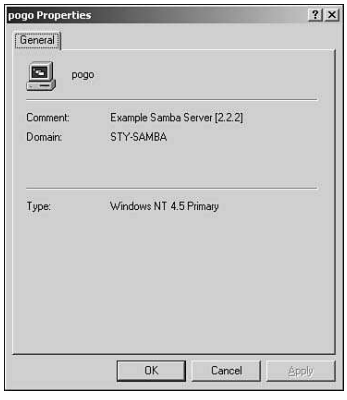


FIGURE 5.1
Properties for a Samba server displayed by a Windows 2000 client.

The difference in the comment shown in Figure 5.1 and the setting in Listing 5.1 is the expansion of the %v variable into the Samba version number of 2.2.2. This leads us nicely into our next topic of smb.conf run-time variables.

Variables

Several variables are available for use in smb.conf. These macros, denoted by a % beginning character, are replaced during the parsing on the configuration file at run time. For example, when user jdoe sends a session setup request, Samba parses smb.conf and replaces all occurrences of %U with jdoe. Table 5.1 contains a listing of the complete set of smb.conf variables that are available to you.

TABLE 5.1 The `smb.conf` Variables

Variable	*Description*
`%a`	The architecture of the remote machine. Not guaranteed to be 100% reliable, but generally good enough in practice. Currently supported values are `Samba`, `WfWg`, `Win95` (for Windows 95/98/ME), `WinNT` (for Windows NT 3.5x/4.0), `Win2k`, and `OS2`.
`%d`	The process ID of the current server process.
`%D`	The domain name of the authenticated user.
`%g`	The primary group of username `%u`.
`%G`	The primary group of username `%U`.
`%h`	The Internet hostname on which Samba is running.
`%H`	The home directory for the username `%u`.
`%I`	The IP address of the client machine in dotted decimal form.
`%L`	The NetBIOS name of the server, which is used by the client in the NetBIOS session request.
`%m`	The NetBIOS name of the client machine.
`%M`	The Internet hostname of the client machine.
`%N`	The name of your NIS home directory server as specified in the auto.home map. If you have not compiled Samba with AUTOMOUNT support, this is the same as `%L`.
`%p`	The path to the user's home directory as specified in `auto.home`. The NIS map entry is assumed to be colon separated and is divided as `%N:%p`.
`%P`	The root directory of the current service.
`%R`	The protocol selected during the protocol negotiation phase of the connection setup. Valid values are CORE, COREPLUS, LANMAN1, LANMAN2, or NT1.
`%S`	The name of the current service.
`%T`	The current date and time.
`%u`	The username of the current service.
`%U`	The username the client requested in the session setup. This is not necessarily the same as the one that was used.
`%v`	Samba version number.

These variables can be used in the place of any parameter value where the expanded value would be sensible. We have already seen one example using the version number (`%v`). The following `[global]` parameter entry will instruct Samba to log information to a file that is appended by the NetBIOS name of the connection client.

```
log file = /var/log/log.%m
```

This is a common means of monitoring certain clients or debugging problematic ones. Under default circumstances, smbd will thread connection information from all clients into a single file making it difficult read. We will use the smb.conf file's variables a great deal throughout the remainder of this book so it would be a good idea to mark this table for later reference.

Environment Variables

In addition to the built-in variables described in the previous section, Samba supports the use of shell environment variables in smb.conf. The variables can be used as a parameter value just as the built-in ones can. However, environment variables are accessed using the syntax %$(<variable>). If there was a environment setting named HOST which was set to the hostname of the local machine on which Samba was running, this information could be accessed by referring to %$(HOST).

The source environment parameter allows an administrator to define an alternative set of environment variables at run time as opposed to using the ones defined when the smbd or nmbd daemon was initially started. These variables can either be read from an existing file or generated and read from the standard output of a program.

The first example shown here causes smbd to read the file named /etc/samba.env and set the variables found inside.

```
source environment = /etc/samba.env
```

The contents of this file could appear as follows

```
GROUPDIR=/etc/samba/group/
MYVARIABLE=myvalue
```

The second form of the source environment parameter reads the variables from the standard output of a program. The distinguishing syntax between defining a file to be read and one to be executed is the '|' character. This next example instructs smbd to execute the program /etc/samba.env and read the variables from its output. Notice that the only difference between this and the previous version of the parameter is that the '|' character immediately follows the equal sign.

```
source environment =|/etc/samba.env
```

The question now is when are environment variables useful in smb.conf. The reason that this parameter was first developed was to allow a single smb.conf file to be replicated to multiple servers and allow the server to take on its own personality based on Samba's process environment. We won't go into the details of that now because we have just begun our examination of Samba's configuration file.

Parameters

For the 2.2 release, Samba developers have brought the total number of global parameters to approximately 160 and the total number of service parameters to 120. For version 2.0, there were around 130 global parameters and 100 service parameters. The current `smb.conf` man page for version 2.2 is 130 pages in length! Needless to say, quite a few options are available for configuring your server.

In order to make this large meal digestible, we'll group parameters based upon functionality. Even so, not all parameters will be covered here. For a complete listing of all current parameters, as always, refer to the `smb.conf` man page distributed with your release.

The values for parameters, with a few exceptions, fall into three categories:

- Strings of characters, such as usernames, or a descriptive string, such as was the case in Listing 5.1, with the `server string` parameter value. String values are almost always case sensitive.
- Boolean values, which can be represented as yes/no, true/false, or 1/0. Boolean values are not case sensitive so `YES`, `Yes`, and `yes` are all equivalent as far as Samba is concerned.
- The catchall categories of numeric values, such as creation masks for files or discrete sets of values as seen with the different printing styles of BSD, CUPS, or LPRNG.

Parameters lines in `smb.conf` follow the form name = value such as:

```
netbios name = POGO
```

Only the first = sign is used in the parsing of the parameter and value. The value begins at the first nonwhitespace character following the equals sign and continues until it locates the first carriage return that is not preceded by a \ character. This next line is equivalent to the previous netbios name setting (remember that parameter names are not case sensitive or whitespace sensitive).

```
NetBIOS    Name     =       POGO
```

Common `[global]` Parameters

The remainder of this hour will examine some of the more common `[global] smb.conf` options.

NetBIOS Naming Parameters

We have already covered two of the most common NetBIOS naming parameters in the previous hour—`netbios name` and `workgroup`. These respectively define the unique NetBIOS name assigned to the Samba host and the NetBIOS group to which it belongs.



Remember that the workgroup parameter has a double meaning. One use is to specify the workgroup name for network browsing and the other is to configure the domain name to which the server belongs. The second definition will be covered in detail in Hour 14, "Replacing a Windows NT File and Print Server—Lock, Stock & Barrel."

The `netbios aliases` Parameter

In Hour 2, "Windows Networking Concepts," we learned that the source client in a NetBIOS connection is referred to as the calling name and that the destination host was the called name in the connection. A NetBIOS server answers only requests that match its called name. The `netbios aliases` parameter allows Samba to reply to multiple names very much in the same way that I respond whether or not someone calls me Jerry or Gerald. However, if a person called me Joe, I would not likely respond because I would not know that I was being addressed.

The existence of the `netbios aliases` parameter means it is possible to see the same server under multiple names within a workgroup when browsing through the Network Neighborhood. I have commonly used this option when consolidating several CIFS servers into a single host.

Each server name could provide different services while still residing on the same physical machine. The default is to only reply to the server's primary `netbios name`. Suppose that our Samba server CAESAR was expanding his empire and was to take over the duties of the older Samba servers named Italy and Greece. However, users still expected to be able to access certain file shares on both Greece and Italy. An `smb.conf` configuration that would allow all three names to appear in the network neighborhood would look like this:

```
[global]
   netbios name = CAESAR
   workgroup = ROME
   netbios aliases = ITALY GREECE
```

Figure 5.2 displays the effects of this `smb.conf` when viewed by a Windows 2000 client.

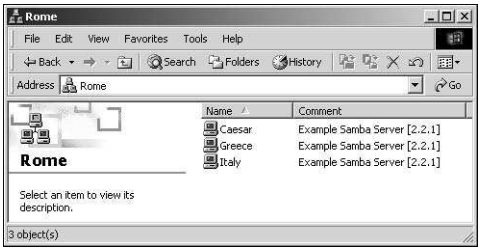


FIGURE 5.2
Using the `netbios aliases` *parameter to force a Samba server to appear with multiple names.*

The `netbios scope` Parameter

The `netbios scope` parameter allows you to define a scope ID that will be appended to Samba's `netbios name` exactly as it is done in Windows clients.

However, the same warning given in Hour 2 still applies. You should never set a `netbios scope` unless you can explain very clearly why it is necessary. Over all the Samba servers and Windows networks I have administered, I have never seen a legitimate need to use a `netbios scope`.



For more information on Samba's `netbios scope` parameter, refer to the `smb.conf` man page.

Logging

Log files can be used for all types of administrative duties, such as monitoring the security of a system or simply the day-to-day usage of it. Samba offers several options for how much information to log and where to store it.

The `log file` Parameter

The `log file` global parameter provides a means of overriding the compile time default location and name of the `smbd` log (`log.smbd` by default). The `nmbd` daemon will also utilize this parameter. However, `nmbd` will use it only to determine the directory where it should write its log file (`log.nmbd`), not to determine the actual name of the file.

There are some caveats to this parameter. You need to be aware of the order in which things occur.

1. If a file is specified using the `-l` switch at startup, `smbd` writes the initial log entries to the filename on the command line. If no location is given at startup, `smbd` logs initial information in the file set at compile time.

2 After the configuration file is parsed once and the `log file` parameter is encountered, all future log entries are written to the file specified by the parameter's value.

This means that because `smbd` cannot be aware of the `log file` location as defined in the `smb.conf` file, it writes some initial information in the log file that it is aware of at startup.

Using the `smb.conf` variables defined in Table 5.1, this example would create a separate log file for each CIFS client machine that connected (or attempted to connect) to the server.

```
log file = /var/log/log.%m
```

The `log level` Parameter (a.k.a. debug level)

The `log level` parameter enables you to set the maximum level of debug entries to be logged. Although Hour 4 mentioned that valid debug levels exist between 0 and 10, it was not clear what information was logged at each level. In fact, this is one area where improvement is being discussed. Historically, there has been no enforcement of placing certain types of messages at a particular level. This was left up to the discretion of the developer who was writing the code. Table 5.2 contains some general rules that can be seen in Samba's logging system.

TABLE 5.2 Information Recorded at the Various Log Levels

Level	*Description*
0	Critical failures, such as failure to open a log file, dropping a connection, or receiving an unknown CIFS command
1	Connection and session information
2–4	System administration debugging information
5–9	Moderate developer debugging data
10	Full developer debugging information

The parameter has a default integer value of 0 and can be set at run time using the `smbcontrol` tool as shown here

```
root# ps -ef | grep smbd
root     32019     1  0 Jul12 ?        00:00:00 bin/smbd -D
root     14533 32019  0 Jul12 ?        00:00:02 bin/smbd -D

root# smbcontrol 14533 debug 10

root# ./smbcontrol 14533 debuglevel
Current debug level of PID 14533 is 10
```

The `log level` parameter sets the logging level for both `smbd` and `nmbd`. Often system administrators prefer to set a debug level of 1 (shown here) in order to save information on who connected to the server, when they connected, and from where.

```
log level = 1
```

The `debug timestamp` (a.k.a. `timestamp log`), `debug pid`, and `debug uid` Parameters

Several global parameters exist to preface a log entry with data that identifies the specifics of the process and the user that wrote the information. Both `smbd` and `nmbd` will record a log header prior to writing the actual log entry. Here is an example of what a log header and log message look like.

```
[2001/07/10 15:07:40, 1] lib/debug.c:debug_message(247)
  doing parameter host msdfs = no
  doing parameter large readwrite = no
```

If the `debug pid` and `debug uid` parameters are also enabled, the entry header will contain the process identification of the `smbd` or `nmbd` process that wrote the entry as well as the user identification information for the current state of the process. This entry was written by the `smbd` process 14533 using the effective and real uid/gid of `0/0`.

```
[2001/07/13 02:04:19, 1, pid=14533, effective(0, 0), real(0, 0)]
lib/debug.c:debug_message(247)
```

Using its default settings, `smbd` and `nmbd` will timestamp all log entries but will not include the additional `pid` and `uid` information. In order to produce the previous entry timestamp, the following settings were added to the `[global]` section of the server's `smb.conf`:

```
[global]
      <...>
      debug timestamp = yes
      debug pid = yes
      debug uid = yes
```



When working with high log levels (for example, 10) that record large amounts of information, it is often helpful to disable the `debug timestamp` parameter so that the log entries appear more contiguous.

The max log size Parameter

Higher logging levels can result in extremely large log files. The max log size global parameter accepts an integer value to specify the maximum size (Kb) of a log file before it should be renamed with an .old extension and replaced with a clean log. If a previous backup log exists with the same name (logfile.old), it is overwritten.

The maximum size set by default is 5MB. This can be set to any value. The following entry would set the maximum log file size to 20MB:

```
max log size = 20000
```

The max log size parameter is still honored using the example log file given in the previous section, which created a separate file for each connected client machine. In this case, each individual file is subject to the maximum size (not the combined size of all logs).

The syslog Parameter

The system logger daemon (syslogd) provides support for centralized logging on modern Unix systems. In order for any of the smb.conf syslog-related parameters to have an affect, the --with-syslog option must have been enabled when running the configure script:

```
./configure --with-syslog
```

The syslog parameter takes an integer value and maps Samba debug levels to syslog log levels. Table 5.3 lists the mappings. Only Samba debug messages with a level less than the integer specified are sent to the syslog daemon. Therefore, the default is to send only 0-level debug messages to syslog. A syslog setting of

```
syslog = 3
```

would cause all log entries from levels 0, 1 and 2 to be sent to the syslog facility. Table 5.3 lists how the various debug levels map to the syslog error levels.

TABLE 5.3 Samba Debug Level to syslog Level Mapping

Samba Debug Level	*Samba syslog Level*
0	LOG_ERR
1	LOG_WARNING
2	LOG_NOTICE
3	LOG_INFO
>3	LOG_DEBUG

See the `syslogd` and `syslog.conf` `man` pages on your system for more information of the `syslog` logging facility.

The `syslog only` Parameter

This Boolean global parameter determines whether messages are sent only to the `syslogd` daemon or whether they should additionally be sent to Samba's own log files. This parameter is used in conjunction with the `syslog` parameter and also requires that the `--with-syslog` flag was enabled for the configure script. The default is to log debug entries to the standard `smbd` and `nmbd` logs in addition to the `syslog` files. If `syslog only` is enabled as shown here,

```
syslog only = yes
```

Samba will be forced to send log entries to the `syslogd` daemon exclusively.

Internationalization

Samba's support for non-English clients is another area where it is showing its age. The Samba 2.2.x server applications do not support Unicode strings when sending requests and replies across the network. This means that both `smbd` and `nmbd` rely upon certain `smb.conf` parameters for information about what DOS code page the CIFS client is using.

All DOS code pages support ASCII (i.e. English) characters and therefore will support English clients. The implication is that it is possible to support Windows clients using English code page and one other language. It is not currently possible to support two non-English clients, such as Greek and Chinese, from the Samba server.



Unicode support is scheduled for completion in the 3.0 release of Samba. Once this happens, all of the code-page and character-set parameters described here will be depreciated.

In addition to the code page used by the client, some Unix file systems use a character set to store filenames on disk. Figure 5.3 illustrates how Samba attempts to fill the gap between DOS code pages and Unix character sets.

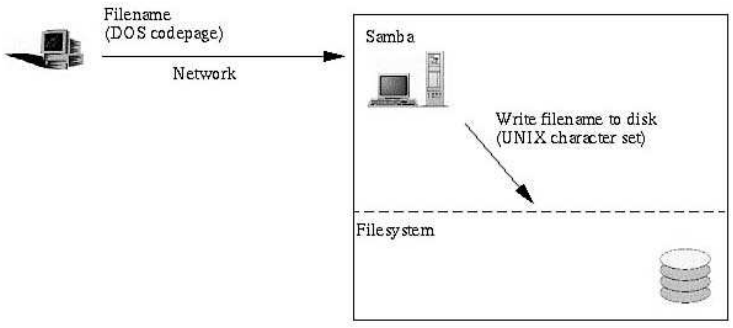


FIGURE 5.3
Samba's interaction with DOS code pages and Unix character sets.

The `client code page` Parameter

A DOS code page is used to define non-ASCII characters using two bytes instead of the one byte needed to represent 7-bit ASCII. This code page enables the use of extended characters, such as á or ö.

There is no means of determining the code page used by a client as part of the CIFS protocol negotiation. Therefore it is necessary to manually configure this using the `client code page` parameter. If it were necessary to support Windows clients that used the Simplified Chinese code page, the following parameter would need to be set the [global] section of `smb.conf`:

```
client code page = 936
```

The value `936` was a predefined code page installed with Samba. Table 5.4 lists the currently supported code pages.



It is possible to create new code pages using the `make_smbcodepage` tool included with Samba. Refer to the `make_smbcodepage` man page for more details.

TABLE 5.4 Samba Packaged Code Pages

Code Page	*Description*
437	MS-DOS Latin US
737	Windows 95 Greek
850	MS-DOS Latin 1 (default)
852	MS-DOS Latin 2
861	MS-DOS Icelandic

continues

TABLE 5.4 Continued

866	MS-DOS Cyrillic
932	MS-DOS Japanese Shift-JIS
936	MS-DOS Simplified Chinese
949	MS-DOS Korean Hangul
950	MS-DOS Traditional Chinese

If the Japanese Shift-JIS (932) code page is selected, a `coding system` parameter can be defined to determine how incoming Shift-JIS Japanese characters are mapped from the incoming client code page used by the client into file names in the Unix file system. You should refer to the `smb.conf man` page for more details on the `coding system` setting.

The `code page directory` Parameter

This is a convenience parameter that accepts an absolute directory path that informs `smbd` where to look for the DOS code pages. A common location used by vendors is

```
code page directory = /etc/samba/codepages
```

A normal default configuration will expect the DOS code pages to exists in `/usr/local/samba/lib/codepages/`.

The `character set` Parameter

It would be much easier if Unix systems used the same code pages as Microsoft clients. Of course, I wouldn't be saying this wishfully if they did. If there is a need to translate the DOS code page version of a filename into the Unix character set used by the server, it must be defined and must match one of the static mappings used by Samba. Table 5.5 gives the valid mappings supported by Samba 2.2

TABLE 5.5 Unix Character Set and Equivalent DOS Code Page Mappings Supported by Samba

Character Set	*Supported DOS Code Page*
Western European (ISO8859-1)	850
Eastern European (ISO8859-2)	852
Russian Cyrillic (ISO8859-5)	866
Greek (ISO8859-7)	737
Alternate mapping for Russian Cyrillic (KOI8-R)	866

Most often, the `character set` parameter can be left as the default setting (that is, no character set mapping done from the DOS code page).

State Information

In addition to Samba's static configuration file, certain runtime information is maintained to help configure the state of the server.

The lock directory Parameter

The lock directory accepts a directory path that determines where Samba writes its database files, browse list, and WINS database (if WINS support is enabled). The default lock directory, determined at compile time, depends upon the configure options that were enabled. Under entirely default configurations, this is normally /usr/local/samba/var/locks.

A practical example of when it would be necessary to manually specify a lock directory is the case where the default lock directory would reside on an NFS-mounted file system. Many sites mount a file system at /usr/local/ for the purpose of sharing tools and utilities unique to its network. Although you can share binaries among Samba servers, it is impossible to share a lock directory. This must be located on a local disk and unique to each server.

Under Samba 2.2.x, is it also a requirement that the contents of the lock directory persist across a system reboot. Certain files stored in the lock directory contain configuration information about the state of the server. The /var/cache directory often is a sensible location for this and is the one used by many of the Samba precompiled packages.

```
lock directory = /var/cache/samba
```

Summary

Although there are quite a few smb.conf parameters, you need to set only the ones you want to use or the ones you explicitly want to define if you're as overly cautious as I am. For example, this is probably the simplest working smb.conf file that I can imagine. This is a simple server that offers no file or print shares.

```
[global]
     workgroup = STY-SAMBA
```

In this hour we have covered smb.conf support for

- Runtime variables
- Additional NetBIOS name parameters
- Logging of connection and debugging information
- International clients
- Maintaining state information

How elaborate you make your smb.conf file is up to you and the needs of your network.

Q&A

Q After I make a change to a Samba configuration file, do I need to kill and restart the Samba daemons?

A For most configuration changes, you do not need to do anything. Samba periodically checks to determine whether the configuration file has changed. If it has, it is reloaded. It is possible to force `smbd` or `nmbd` to manually reread the configuration file by sending the process a HUP signal (e.g. `kill -HUP <pid of smbd>`)

Some changes require that Samba be restarted, such as the `NetBIOS name` and the `workgroup` parameters.

Q Can Samba be a member of more than one workgroup at the same time?

A No, Samba can be a member of only one workgroup.



HOUR 6

SWAT and Other GUI Administration Tools

In the previous hour, we began to explore many of the intimate details of Samba's `smb.conf` file—its `[global]` section in particular. Although the remaining hours will continue to present textual forms of `smb.conf`, we will allow ourselves a short detour to view two of the available Web-based configuration tools for Samba.

Directly editing `smb.conf` with your favorite text editor is often the most flexible means. However, a GUI editor has the advantage of being able to provide more context for the parameter values. In this hour, we will cover Samba's Web administration tool (SWAT) and the Samba Webmin plug-in.

SWAT

SWAT is a miniature Web server and CGI scripting application designed to run from `inetd` that provides access to the `smb.conf` file for the system on which SWAT is running. The `inetd` metadaemon handles the startup of most

network servers under Unix and is controlled by the file /etc/inetd.conf. (For more details on inetd, refer to the inetd(8) man page.)

SWAT enables a suitably authorized person to configure Samba via Web pages. SWAT also places help links to all configurable smb.conf options on every page, which lets administrators easily understand the effect of any changes.

SWAT is installed as a part of most precompiled Samba packages. If you have compiled from source code, Samba's configure script supports an option to tell Samba where to place the HTML files used by SWAT. The --with-swatdir switch accepts a directory path as shown here:

```
root# ./configure --with-swatdir=/usr/share/swat
```

If you do not specify a --with-swatdir, which is the most common case, the SWAT HTML files are installed in ${prefix}/swat (i.e. /usr/local/samba/swat).

Once the make install command copies the swat binary and help files to the install directory, only a few minor steps are necessary to obtain a working SWAT server. If you installed Samba from RPMs or another package form, these next steps are frequently performed for you.

There are two popular metadaemon servers installed on Unix hosts—the inetd server and the xinetd server. First we will examine the steps necessary for configuring SWAT for work with inetd.

The inetd Server

The inetd server is controlled by the /etc/inetd.conf configuration file. The actual format of this file can vary from Unix to Unix and distribution to distribution so it is best to consult the inetd.conf(8) man page for the definitive answers on how to add a new entry.

Before configuring inetd.conf, the following line must exist in the /etc/services file:

```
## assign swat to port 901
swat          901/tcp
```

The use of port 901/tcp is more from convention than from a technical reason. It is possible to configure SWAT to use an alternative port by simply modifying the entry in /etc/services.

Next, we must add a line to /etc/inetd.conf like the following:

```
swat  stream   tcp  nowait.400   root   /usr/local/samba/bin/swat  swat
```

After adding the entries to the `services` file and to `inetd.conf`, the `inetd` server must be instructed to reread its configuration file by sending a HUP signal to it. To do this, you can use a variety of methods. The most portable is `kill -HUP PID` where `PID` is the process ID of the `inetd` daemon.

The `xinetd` Server

The `xinetd` server is presented as a more secure version of the traditional `inetd` server and is available from `http://www.xinetd.org/`. The `xinetd` server supports two styles of configuration files for services. The first is a single file commonly named `/etc/xinetd.conf`. The second model is to give each service its own configuration contained in a directory named `/etc/xinetd/`. The following example is from Samba's own packaging files for RedHat 7.x. These files can be found in the `packaging/` subdirectory of the Samba source distribution.

```
## /etc/xinetd/swat.conf
service swat
{
        port            = 901
        socket_type     = stream
        wait            = no
        only_from       = localhost
        user            = root
        server          = /usr/local/samba/bin/swat
        log_on_failure  += USERID
        disable         = no
}
```

Notice that this entry defines the port that should be used and defines that logins should be allowed only from the localhost. There are many more parameters available to `xinetd`, which are outlined in the `xinetd.conf(5)` man page.

Once the SWAT configuration has been created, then the `xinetd` daemon must be instructed to reread its configuration information. This can be accomplished by sending it a USR1 signal by

```
root# kill -USR1 <pid of xinetd>
```

Logging In

Once SWAT has been configured to work with the metadaemon server installed on your system, it should be possible to use any Web browser to access the SWAT server. To do this, visit your Samba server on port 901 by going to `http://your-server:901/`.



All information (configuration file, username, password, and so on) sent from your Web browser to the SWAT server is transmitted in clear text. This means that anyone who can view the traffic on your network will be able to read whatever you type in. This is rarely a good idea for account information. My recommendation is to use SWAT only when you are running the browser on the same machine as the SWAT server and connecting to the URL `http://localhost:901/`.

When your browser contacts SWAT, it will present a dialog box asking for a username and password. In order to access the full functionality of SWAT, you must enter a sufficiently privileged username and password here (for example, `root`). Table 6.1 displays the various SWAT features and the privilege levels required to access them.

TABLE 6.1 SWAT Features

Feature	*Privilege Required*
Viewing Samba's online help files	Any user
Modifying `smb.conf`	Access is controlled by the file permissions on `smb.conf`
Viewing `smb.conf`	Any user with adequate privileges based on the UNIX permissions assigned to the `smb.conf`
Managing user accounts in `smbpasswd`	Root access is required
Changing own password	Any user
Managing `smbd/nmbd` daemons	Root access is required
Viewing status of connections	Any user

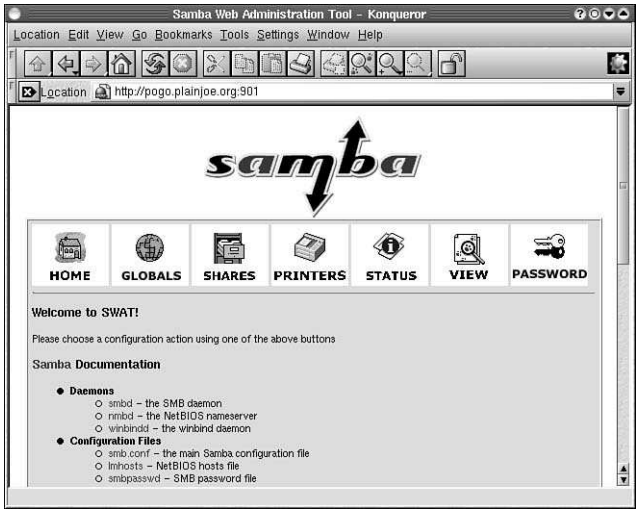
SWAT will always authenticate the logon attempt using the local system accounts and passwords (that is, `/etc/passwd`) and not the LanMan/NT password hashes, which will be covered in the next hour. In order to log on to a server that is configured to use MD5 hashes in `/etc/passwd` (or `/etc/shadow`), Samba must have been configured with PAM support (`--with-pam`), which will be discussed in detail in Hour 16, "Managing User Accounts and Single Sign-On."

Once logged in, you are presented with the main SWAT page, shown in Figure 6.1, and various other buttons depending on the level of privilege assigned to the account used. We will assume that we have logged in using a root account for now. As root, the following buttons appear:

- Home—which takes you back to the SWAT home page. This page also contains links to online documentation, such as the man pages and Samba-HOWTO-Collection.
- Globals—where you can manage the Samba `[global]` section of this Samba server.
- Shares—where you can manage file shares for this Samba server.
- Printers—where you can manage printer shares for this Samba server.
- Status—where you can obtain status information about Samba on this server.
- View—where you can view the current `smb.conf` file.
- Password—where you can manage your password on your Samba server or on a remote machine.

At any time you can return to the SWAT home page by clicking on the Home icon.

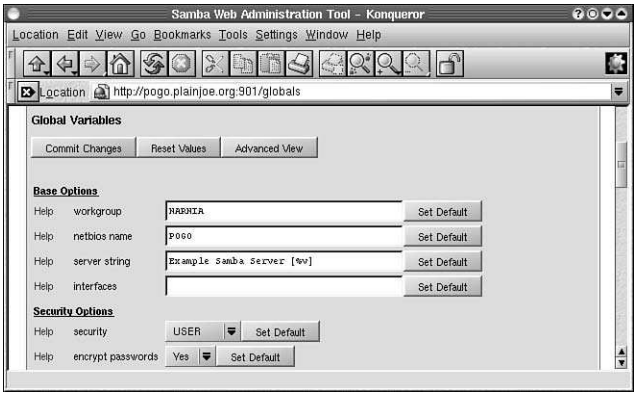
FIGURE 6.1
Main SWAT page.



Managing the `[global]` Section

When you select the Globals icon, SWAT returns with a Web page similar to the one shown in Figure 6.2. This page enables us to modify many of the most relevant Samba global parameters, some of which have already been covered and some of which we have not seen yet.

FIGURE 6.2
SWAT's Globals page.



The Samba global variables are grouped into related options. The Basic View is shown by default and contains only the most common parameters. Clicking the Advanced View button brings up a complete list of Samba's `[global]` options. To make a change, simply scroll down to the parameter you want to change, enter the new value, and then click the Commit Changes button.



You must select Commit Changes whenever you access a new page in SWAT or else all the changes on the previous page will be lost.

It is also nice to know that the HTML version of the `smb.conf` man page is linked to the `Help` tag to the left of each parameter name.

Managing File Shares

Upon selecting the Shares icon, SWAT returns a Web page like the one in Figure 6.3 that enables us to create new shares and modify existing ones.

To modify any of the parameters of an existing share, select the share name from the dropdown list (see Figure 6.3) next to the Choose Share button. Clicking on this button will display a page containing the information regarding the file service you selected.

To create a new share, enter its name in the field next to the Create Share button and then click Create Share. You are then presented with a page similar to that shown in Figure 6.4, with the name of your new share as the choice in the first field.

FIGURE 6.3
Selecting a file share on SWAT's Shares page.

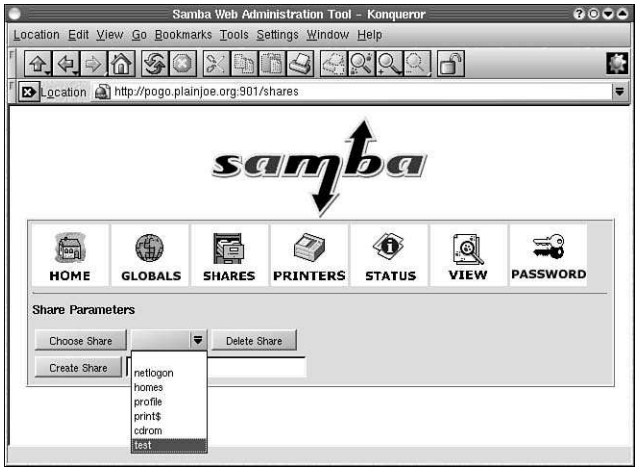
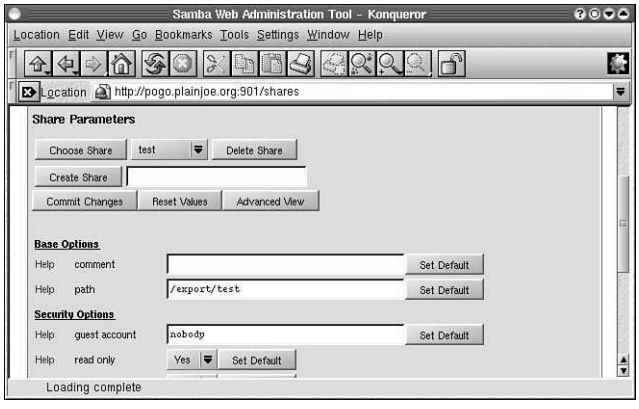


FIGURE 6.4
Editing a file share.



From this page you can

- Choose another share by selecting it and clicking on Choose Share
- Create a new share by entering its name in the appropriate field and clicking on Create Share
- Commit all your changes made so far by clicking on Commit Changes
- Delete the share by clicking on Delete Share

If you need to modify parameters not shown on this page, click the Advanced View button and modify the appropriate parameters.

The Advanced View page shows all the parameters related to the selected share, grouped in the following sections:

- Base Options, such as `comment` and `path`
- Security Options, such as `username`, `guest account`, and so on
- Logging Options, such as `status`
- Tuning Options, such as `max connections`, `sync always`, and so on
- Filename Handling, such as Case-Handling Parameters, and so on
- Browse Options, such as `browsable`
- Locking Options, such as `oplocks`, `strict locking`, and so on
- Miscellaneous Options

When you have made any desired changes, click Commit Changes, and they will be applied immediately.



Any committed changes will immediately be applied to the `smb.conf`. Any currently running `smbd/nmbd` processes will not see the changes until they re-read the configuration file. This can be done manually by sending a HangUp (HUP) signal to a specific process, or waiting for Samba to notice the change during its periodic checks. New processes will read the new `smb.conf` upon start up. If you have changed the `netbios name` or `workgroup` options, Samba must be restarted.

Managing Printer Shares

When you select the Printers icon, SWAT returns a Web page (see Figure 6.5) that enables you to create and modify printers defined in `smb.conf`.

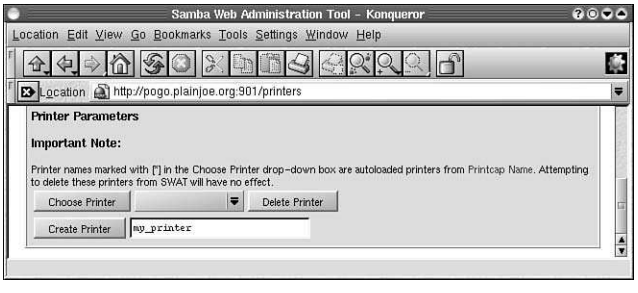


FIGURE 6.5
Creating a new printer in SWAT.

To modify an existing printer, select it from the dropdown list next to Choose Printer and then click on this same button. To create a new printer, enter the name of the printer in the field next to Create Printer and click on the Create button. You are presented with a page similar to that shown in Figure 6.6, with the name of your new printer in the first field.

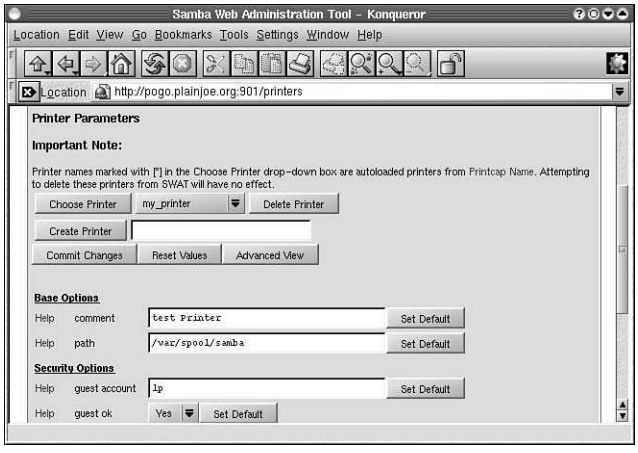


FIGURE 6.6
Defining the settings for a new printer in SWAT.

From this page you can

- Choose another printer by selecting it and clicking on Choose Printer
- Create a new printer by entering its name in the appropriate field and clicking on Create Printer
- Commit all your changes made so far by clicking on Commit Changes
- Delete the printer, by clicking on Delete Printer

If you need to modify parameters not shown on this page, click the Advanced View button and modify the appropriate parameters.

When making the necessary changes, clicking the Commit Changes button instructs SWAT to write these to the `smb.conf` file.

Obtaining Status Information

Besides acting as a graphical `smb.conf` editor, SWAT can also provide information about the currently running Samba server daemons from the Status page shown in Figure 6.7. In addition to displaying share mode and oplock information on files, this page provides controls for restarting Samba daemons or simply terminating individual connections. The status page also provides a means of having it refreshed on a continual basis. Simply specify the refresh interval and click Auto Refresh.

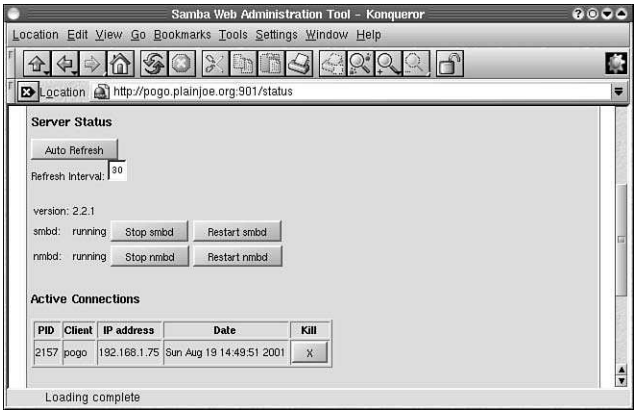


FIGURE 6.7
Viewing the status of a Samba server.

Viewing the Complete `smb.conf` File

When you click the View icon, SWAT returns a Web page that displays the entire contents of the server's `smb.conf` file. SWAT lists the Samba configuration file as it appears in the `smb.conf` file. If you want a listing that includes the values of all the parameters that Samba maintains, simply click the Full View button.



Selecting the Full View option is identical to the output from the `testparm` command in regards to showing default values for parameters.

Changing Your Password

The Password icon is linked to a page, shown in Figure 6.8, that provides a graphical wrapper around Samba's `smbpasswd` command. As root, this means you can add, delete, or modify any user's `smbpasswd` account on the Samba server. A normal user will only be able to change his or her own password on the Samba server (or another CIFS server, such as a remote Windows NT PDC).

Security Concerns

The point was already made that SWAT transmits all information in clear text across the network. It's possible to wrap this information so that it can be sent securely from a remote client to your Samba server. Rather than diving into the details of SSL and cryptography right now, this discussion will be continued in Hour 17, "Security Tips."

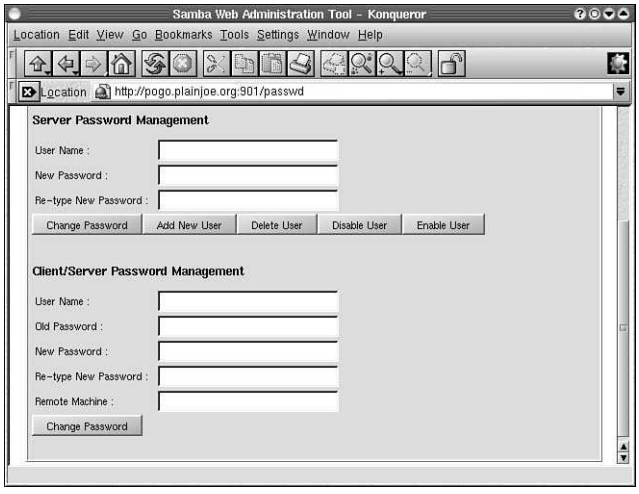


FIGURE 6.8
SWAT's password page while logged on as root.

Webmin

Webmin is a Web-based system administration package for Unix systems. It provides facilities for managing Samba, as well as for setting up accounts, configuring DNS, configuring Apache, configuring sendmail, and performing many other system administration tasks. Our concern here is how to use Webmin's Samba plug-in.

Webmin consists of a miniature Web server written in Perl and a set of CGI programs that implement the functions required to provide system configuration over the Web. Webmin can be obtained from `http://www.webmin.com/webmin/`. Assuming that you have downloaded the gzipped source tarball from www.webmin.com, it is necessary to extract the distribution into a directory of its own by executing

```
root# gzip -dc webmin-VER_tar.gz | tar xvf -
```

where VER is the current version number of the package.

After breaking out the distribution, simply change to the directory you just created, usually `webmin-VER`. You should then refer to the `README` file for installation instructions. Currently this consists only of a single command:

```
root# ./setup.sh
```

The installation script prompts for more information about our system when necessary. One of these questions will be for a password to associate with the first Webmin user, named `admin` by default. This password will be required when we connect to the Webmin page in the same way that SWAT expected us to logon using the root username and password.

Although Webmin's Samba plug-in is not as intimately aware of the `smb.conf` file's parameters as SWAT is (nor could it be in all practicality), Webmin does support HTTPS for secure connections to the server as part of the installation process. This makes it much easier to secure a Webmin server than to secure a SWAT server. In order to enable this security feature, the Net::SSLeay Perl module must also be installed. If you are interested in about this module and Perl modules in general, refer to the Comprehensive Perl Archive Network (CPAN) Web site at `http://www.cpan.org`.

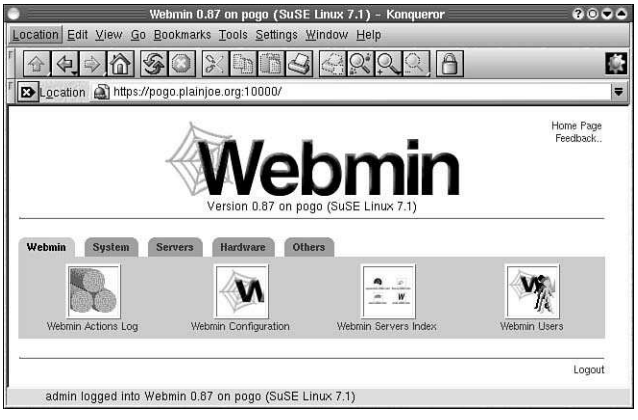


Webmin supports a large variety of systems. You should consult `http://www.webmin.com/` for the latest list.

After installing and starting Webmin, the server can be accessed from any browser by accessing port 10,000 on the Webmin server (for example, `https://localhost:10000/`).

Once we have entered the correct username and password, we are presented with the Webmin home page as shown in Figure 6.9. Webmin can administer many aspects of Unix systems, but we are interested only in its capability to administer Samba. To perform Samba-related configuration, select the Samba Windows File Sharing icon from the page displayed by selecting the Servers tab.

FIGURE 6.9
Webmin's front page.



After selecting the Samba icon, Webmin presents a page dedicated to Samba file and printer shares. From the page shown in Figure 6.10, we can manage file and printer shares, as well as all aspects of the Samba global parameters. From these pages, you can

perform the same set of functions as you can with SWAT. It is even possible to have Webmin act as a wrapper for SWAT by clicking on the SWAT button located at the bottom of the Samba Share Manager page.

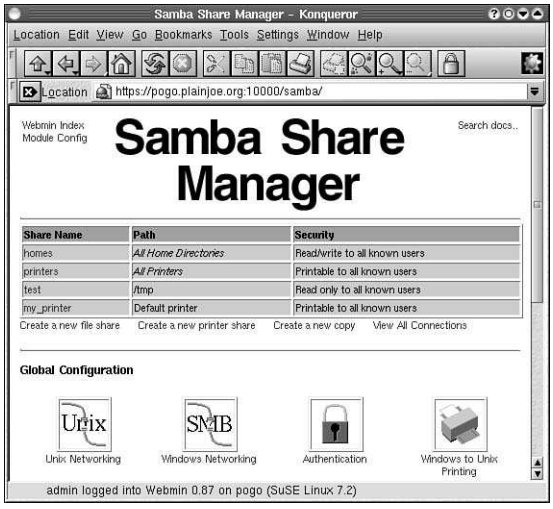


FIGURE 6.10
Configuring Samba shares in Webmin.

In conclusion, be aware that Webmin's Samba plugin comes with pre-configured paths to the Samba binaries and files. These settings may or may not fit your server. There is a link normally located in the upper, right-hand corner of the Webmin Samba page for the "module configuration" screen. Selecting this link will provide access to fields for defining the locations of the various Samba files such as `smbd`, `nmbd`, `smbstatus`, and others.

Summary

Configuring Samba's `smb.conf` can be one of the most difficult aspects of getting a server to function correctly on a system. The two tools that we have explored in this hour, SWAT and Webmin, make it easier to manage Samba. The former tool has the advantage of better integration with Samba's configuration parameters. However, Webmin's support for multiple services may prove to be a better-rounded administration package for your purposes.

Be careful of logging on to either administration tool from a remote machine. Due to the level of privilege needed to modify Samba's configuration (i.e. a root account), sending this information across a network where it could be viewing by other such as the Internet is never recommended without the use of some type of transport layer security such as SSL. This and other security concerns will be addressed in Hour 17, "Security Tips."

In the next hour we will talk about the various security models used by Samba when authenticating users. This will include a discussion about clear text passwords, NTLM, and guest access.

Q&A

Q **I am trying to use SWAT, but I keep getting the message "There was no response. The server could be down or not responding. . ." What is the problem?**

A The most likely cause is that SWAT is not listening to connections, or you have used the wrong URL in trying to connect to SWAT. SWAT usually lives behind port 901, so the URL you should use is `http://your.server.dom:901/`, where `your.server.dom` should be replaced with the DNS name of your Samba server. If you are still having problems, check to see that SWAT is listed in the `/etc/services` file and that `/etc/inetd.conf` has an entry for SWAT as outlined previously.

Q **I cannot logon to SWAT even though I know I am typing the correct password. What could be wrong?**

A In order to logon to SWAT installed on a Linux system that is configured to use MD5 password hashes in `/etc/passwd` (or `/etc/shadow`), it is sometimes necessary to include PAM support when compiling Samba (`--with-pam`). Hour 16, "Managing User Accounts and Single Sign-On," will give the full details on Samba 2.2 and PAM.



HOUR 7

Security Levels and Passwords

After two hours of exploring of Samba's `smb.conf` file, during this hour we take a short detour to examine the different authentication models for validating users that are at our disposal. The goal is to understand what happens, or could happen, when a user connects to a network file share or a printer on a Samba host. At the end of this hour, you will be able to describe Samba's support for clear text and encrypted passwords, local and remote user authentication, and guest connections.

Security Levels and the `security` Parameter

The SMB protocol has two modes of validating connection requests. Which mode Samba uses is determined by the `security` parameter in the `[global]` section of the `smb.conf` file.

The `smb.conf` man page lists four possible values for the `security` parameter. Although this may appear to contradict the earlier statement that the SMB protocol has only two basic modes of authentication, in reality, of the four values that Samba supports—`share`, `user`, `server`, and `domain`—the only two that are distinct are `share` and `user`. These two values correspond to the CIFS security modes of the same names. The other values supported by Samba, `server` and `domain`, are really just variations of user-mode security. The general syntax of the `security` parameter is

```
security = [share|user|server|domain]
```



In this hour, a security *mode* refers to the authentication models of the CIFS protocol itself. Security *level* refers to how Samba implements the various SMB security modes.

We covered CIFS user-mode security somewhat in Hour 2, "Windows Networking Concepts." We now look again at how this works to provide some background before moving on to other Samba security levels.

`security = user`

Figure 7.1 was originally presented in Hour 2. It is interesting to compare Samba's user level security with the remaining three options. We will refer to this diagram in the remaining sections as we learn more about authentication.

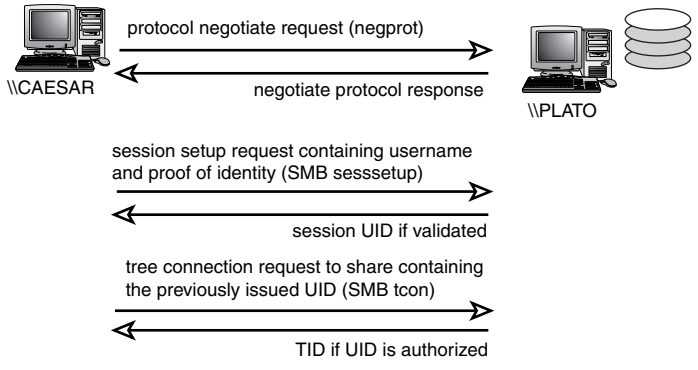


FIGURE 7.1
CIFS user-mode security implemented by Samba's `security = user` *setting.*

Remember the three steps in connecting to a file share when user-mode security is in operation:

1. Decide what CIFS protocol dialect the client and server will speak (the `negprot` request-response pair).
2. Authenticate the user using a login name and some form of credential, such as a password (the `SMBsessetup` request-response pair).
3. Actually connect to the CIFS share (the `SMBtcon` request-response pair).



It is not as easy to describe exactly how connections to printers work as it is to describe connections to file shares. Two common printing methods are used by modern Microsoft clients: LAN Manager–style printing and a system based on Remote Procedure Calls (RPC). More about this is covered in Hour 9, "Samba—The Print Server."

`security = share`

Share-mode security differs from user-mode security in that the client sends a password during the tree connection request as opposed to the `SMBsessetup` request. No associated username is required. Figure 7.2 illustrates the two steps used in a connection to a file share on a CIFS server in share-mode security.

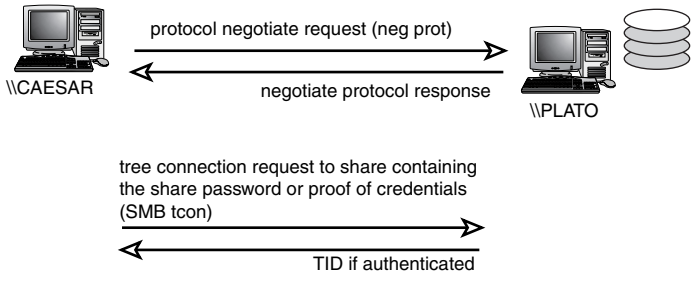


FIGURE 7.2
CIFS share-mode security implemented by Samba's `security = share` *setting.*

A common example of a share-level CIFS server is a Windows 9x client with file and print sharing enabled. In this scenario, the access control to a file share is determined by which password was used for authentication. One pass phrase is associated with read-only

access, while another allows full read-write access. Figure 7.3 shows the directory properties dialog from a Windows ME client that is currently sharing a directory named `TEMP`. Notice the entry fields for the read-only and read-write access passwords.

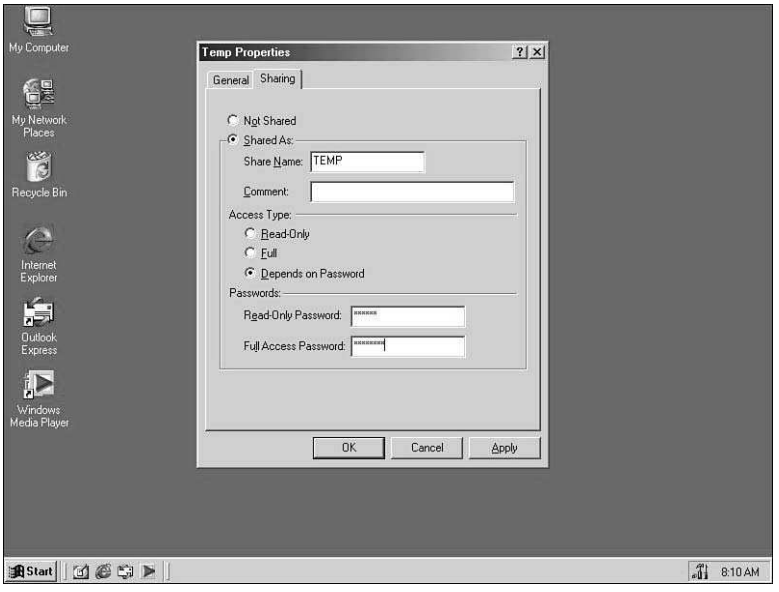


FIGURE 7.3
Windows ME and share-mode security.

By now you might be thinking, "Share-level security seems to violate the Unix username and password authentication model." You are correct. The concept of share-level security does not work well in a multiuser environment, such as Unix, but Samba goes to great lengths not to compromise the Unix security model.

Although the client expects the share-level SMB server to associate a password with each share, Samba uses the standard Unix username and password scheme. In spite of the fact that clients connecting to a server in share-level security send the password in the tree connection request, many modern clients also send a session setup request (`SMBsessetup`) containing a username. Samba adds this username to a list of names that it attempts to validate using the transmitted password. Figure 7.4 shows a modified version of Figure 7.2 with the optional setup step added. While Figure 7.4 may appear identical to Figure 7.1 (user-mode security diagram), be aware that the password (or proof of identity when using password encryption) is still transmitted in the `SMBtcon` request.

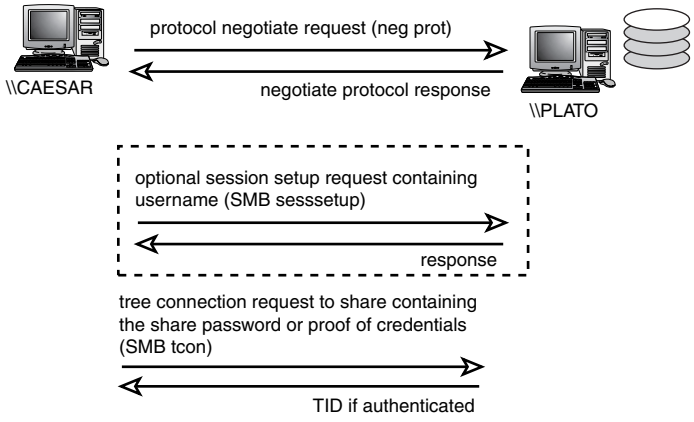


FIGURE 7.4
CIFS share-mode security with the optional (but often present) `SMBsessetup` *step.*

You can add other usernames to the list by using the `user` parameter in the service definition in Samba's `smb.conf` file:

```
user = jerry, smbguest, jdoe
```



File and printer service sections are covered in full detail in Hours 8 and 9.

Samba attempts to validate the connection by trying to validate each username with the password until a pair is successfully authenticated or all possible variations fail, in which case the tree connection is refused. Because Samba always attempts to validate a username-password pair anyway, share-level security is not recommended. It is generally better to use one of the forms of user-mode security.

`security = server`

Samba's server-level security is a variation on user-mode security. If you have ever operated a Windows 9x/ME host as a file server and leveraged user accounts from a central Windows NT/2000 Primary Domain Controller (PDC), then you can easily understand how Samba's server-level security functions.

For the authentication stage, Samba acts as a passthrough server meaning that the protocol negotiation and session setup steps are repackaged and sent to the `password server`. The responses from this machine are then repackaged by `smbd` and passed on to the client. This allows Samba to correctly report certain authentication related capabilities of the `password server`, such as whether or not the server supports password encryption, to the client so that the she can respond with the appropriate information.

Figure 7.5 illustrates this process. The chronological flow of the diagram is from top to bottom. If you follow the arrows, you'll see that at the point where the client requests a session with the server, the Samba host sends a session request to the password server. Only after the server has received a response from the password server is the client's session request accepted or rejected.

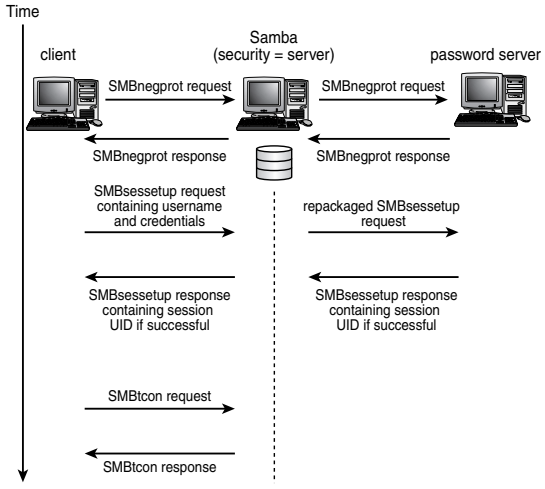


FIGURE 7.5
Connecting to a file share under Samba's `security = server` *setting.*

The `password server` parameter has the following syntax:

```
password server = NetBIOS name of SMB server
```

You can list multiple NetBIOS names in the list, such as

```
password server = DOMAINPDC DOMAINBDC1 DOMAINBDC2
```

This allows Samba to attempt to send a session setup request to each machine in the list in order until a server is contacted. The next machine in the list is contacted only if the previous machine was unreachable, not if the session request to the first machine fails.

You can instead instruct Samba to attempt to locate a domain controller for the domain name given as the value of the `workgroup` `smb.conf` parameter by using the asterisk character in the `password server` parameter:

```
password server = *
```

There is an important point about using Samba's server-level security that applies to domain-level security as well. After Samba has granted the session setup request to the client, it must then have some means of obtaining a Unix `uid` for the user in order to control access to files. This means that although local accounts are not used to validate the connection, the user must have a `uid` on the local server. There are three possible solutions to this problem when using server-level security:

- You can create a local account for all users that access the Samba server and simply disable the password field in `/etc/passwd` (or equivalent file). Setting the password field to the asterisk character (*) normally does this. This is often a good solution if the usernames on the Windows NT/2000 DC are synchronized with login names stored in some type of Unix-centered network directory such as LDAP or NIS.
- User accounts on the Unix host can be created as needed whenever the password server authenticates a user but a user of the same login name does not exist on the Samba server. The `add user script` parameter in `smb.conf` is covered in Hour 14, when domain-level security is presented as a means of replacing a Windows NT file and print server.
- Windows usernames can be mapped to one or more Unix login names using the `username map` parameter discussed later in this hour.

`security = domain`

Samba's domain level security is essentially the same as server-level security, except that the Samba server must be a member of a Windows NT domain. The result is that the Samba server can participate as a client in things such as trust relationships. There are several other advantages to using `security = domain` rather than `security = server`; however, I defer this discussion to Hour 14, "Replacing a Windows NT File and Print Server—Lock, Stock & Barrel." For the meantime, consider them equivalent.

Usernames

Now that we understand how `smbd` authenticates user connections, let's turn our attention to the details of usernames and passwords and how they are sent from the CIFS client to the Samba server. To begin, look at the network packet taken from a connection request from a Windows ME client to a Samba 2.2 server.

LISTING 7.1 Portion of an `SMBsessetup` Request Packet Sent from a Windows ME Client

```
SMB (Server Message Block Protocol)
 Message Type: 0xFF
 Server Component: SMB
 SMB Command: SMBsesssetupX (0x73)
 ...
 Capabilities: 0x0001
 Byte Count: 56
 ANSI Password: SECRET
 Account Name: GUEST1
 Primary Domain:
```

7

continues

LISTING 7.1 Continued

```
Native OS: Windows 4.0
Native LanMan Type: Windows 4.0
...
0  0050 5601 0000 0050 56d3 93a4 0800 4500 .PV....PV.....E.
10 00bf 0014 4000 8006 ca4d c0a8 d784 c0a8 ....@....M......
20 d701 0401 008b 0077 da81 f2a9 f40e 5018 .......w......P.
30 441c 8bd0 0000 0000 0093 ff53 4d42 7300 D..........SMBs.
40 0000 0010 0000 0000 0000 0000 0000 0000 ................
50 0000 0000 d71a 0100 0101 0d75 0075 0068 ...........u.u.h
60 0b32 0000 00cb 5200 0018 0000 0000 0000 .2....R.........
70 0001 0000 0038 0053 4543 5245 5400 0000 .....8.SECRET...
80 0000 0000 0047 5545 5354 3100 0000 0047 .....GUEST1....G
90 5545 5354 3100 0057 696e 646f 7773 2034 UEST1..Windows 4
a0 2e30 0057 696e 646f 7773 2034 2e30 0004 .0.Windows 4.0..
b0 ff00 0000 0200 0100 1300 005c 5c50 4f47 ...........\\POG
c0 4f5c 4950 4324 003f 3f3f 3f3f 00        O\IPC$.?????.
```

As you can see, the username and password both are sent completely in uppercase letters. Very few Unix administrators force the user password to be composed solely of capital letters.

username level

To work around clients that send the username in all uppercase letters, Samba attempts to look up this username in the following order:

1. In all lowercase letters
2. As it was transmitted
3. In all uppercase letters (if this would be a new combination)

Samba is unable to locate mixed upper- and lowercase Unix usernames, such as `jCarter`, with this method if username lookups are actually considered to be case sensitive by the server's operating system.

The `username level` global parameter is provided to work around circumstances like this (although it is rarely necessary in practice). With this option, we can specify the maximum number of capital letters in the username. Samba then attempts to discover the username by using a brute-force method of trying all permutations of capitalized letters from `1` to the value defined.

If we have the following global setting in `smb.conf`

```
username level = 1
```

and apply it to the login name `jCarter`, Samba attempts to locate the following names in the local list of system accounts (for example, by using the C library `getpwnam()` function):

```
jcarter
JCARTER
Jcarter
jCarter
```

If the lookup for `jCarter` had failed as well, the next combination in line would be `jcArter`, and so on. Once the username is located, the search terminates. The greater the `username level`, the more combinations of uppercase and lowercase letters tried and hence the longer the delay before success or failure is reported. If all the Unix account names are in the standard format of all lowercase letters or if login names are case insensitive, this parameter becomes unnecessary altogether.

username map

One of the main problems of connecting Unix and PC operating systems is synchronizing user account information. Some variants of Unix restrict the username to eight or fewer characters, whereas some Windows clients allow a free-form string including white space. Many administrators are required to connect two already-established systems, both with existing account names. The `username map` parameter enables you to specify a file that maps the username sent during the session setup to a local username.

Samba does not perform any type of username mapping from one string to another by default. To enable this mapping, we define the location of the file containing the mapping entries, as shown here:

```
username map = /usr/local/samba/lib/users.map
```

Each entry in the map file takes the form

```
unix username = client username . . .
```

To map the Windows `Administrator` or `Admin` username to the Unix `sysadmin` account, the username map needs this entry:

```
sysadmin = Administrator Admin
```

A user attempting to connect to a share as `Administrator` needs to supply the password for the `sysadmin` account.



Items on the right-hand side of the equal sign in a `username map` entry can be separated by whitespace, a comma, or tabs. If the Windows login name contains spaces such as Gerald Carter, the string should be enclosed in quotation marks (e.g. `jerry = "Gerald Carter"`).

Mapping Windows login names to Unix usernames is only one of the features of the `username map` option. It is also possible to map Unix groups to a single Unix user account. The following line maps any user in the `staff` Unix group to the `staffsmb` account:

```
staffsmb = @staff
```

In addition, Samba supports wildcards as part of the right side of a map entry. This entry maps all users to the guest account:

```
guest = *
```

Beware of a catch regarding substitutions: `smbd` parses the file line by line and carries out any substitutions at the end of the file. This can lead to multiple mappings from `username` to `new_username1` to `new_username2`. If you want to stop parsing the file after a map is made, you should preface the line with an exclamation point (`!`). If no match is found in the file, Samba uses the original username.

Passwords

Before discussing the details of using either clear text or Windows encrypted passwords with Samba, it is a good idea to understand the effects of using one over the other. While this is not a complete list, it offers some general guidelines for when one form is more appropriate than the other.

- Plain-text passwords allow Samba to use the same password database (for example, `/etc/passwd`) as other Unix services such as Telnet and FTP. Often such services also transmit passwords in plain text across the network, so Samba is not lowering existing network security.
- When using plain-text passwords, there is no need for anything other than normal Unix system files to be stored on disk. This can make user management much easier because there is only one set of accounts to handle.
- Windows NT, however, does not like plain-text passwords and will not allow you to browse a server that does not support encryption. It also prompts for passwords to connect to nonencrypted shares, which can get extremely annoying if you connect to a large number of shares.
- Keeping the `smbpasswd` and `unix passwd` synchronized can be difficult. See Hour 16, "Managing User Accounts and Single Sign-On," for more details.
- The CIFS challenge-response authentication model that uses Windows encrypted passwords does not transmit the password over the network. Therefore, the password cannot be viewed using a network monitor tool such as `tcpdump` or `ethereal`. Of course, viewing clear-text passwords sent from clients is easy to do with the right set of tools.

password level

When plain-text passwords are used for authentication, the issue of case sensitivity raises its head again, as it did with login names. This portion of the packet shown in Listing 7.1 reminds us that the password, `secret`, is sent in uppercase letters:

```
70 0001 0000 0038 0053 4543 5245 5400 0000 .....8.SECRET...
80 0000 0000 0047 5545 5354 3100 0000 0047 .....GUEST1....G
```

The `password level` parameter is analogous to the `username level` parameter. The sole difference is that Samba attempts to use the password in this order:

1. As it is sent from the client
2. In all lowercase letters

The `password level` parameter accepts an integer value that defines the maximum number of uppercase letters allowed in the password. Samba then attempts to validate the username using all permissible permutations of capital letters in the original password string. The higher the value, the more combinations Samba tries and the longer the delay in authentication attempts. You need to determine what is acceptable for your server. A password level of `8` on most systems means that the passwords are no longer case sensitive. I have found that a level of `4` is generally acceptable and does not cause too much trouble with existing passwords. However, it is also helpful to have a password change facility that enforces the level you have chosen.

LAN Manager and Windows NT Password Encryption

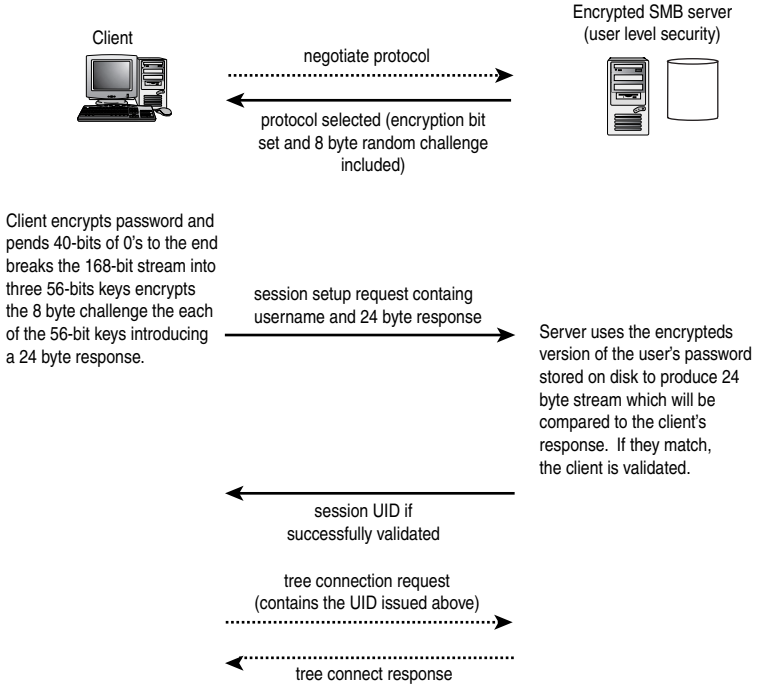
Samba supports both the LAN Manager and Windows NT CIFS password-encryption algorithms. This means that Samba can authenticate users in the same manner that Microsoft servers can.

If you are familiar with Unix's password-encryption scheme, some points might seem similar. For example, the LAN Manager and NT password hashes are irreversible, as are Unix passwords stored in `/etc/passwd` (or `/etc/shadow`). *Irreversible* means that the only way to determine whether a user has entered the correct password is to encrypt the entered password and compare it against the encrypted version stored on disk. There is no means of decrypting a LAN Manager or NT password hash other than by brute-force methods, such as a dictionary attack.

However, one major difference between the password encryption used by Windows clients and that used by Unix hosts is that the LAN Manager and NT password hash-generation technique always produces the same result if given the same input. This means that if you encrypt the password `secret` 200 times, the encrypted password is the same every time. This creates what is known as a *plain-text equivalent password*. The example in Figure 7.6 clarifies this.

1. The client sends a protocol-negotiation request to the server.
2. If the server supports password encryption, the appropriate bit to indicate this is set in the response packet and the server includes an 8-byte challenge in the packet. The challenge is randomly generated and different for each client.
3. Figure 7.7 illustrates the generation of the client's response. The client uses the encrypted password appropriate for the negotiated protocol level (either LAN Manager or NT) with 5 null bytes appended (this creates a 168-bit stream) to generate three different 56-bit DES keys, which are each then used to encrypt the 8-byte challenge. The three 8-byte results are concatenated to form the 24-byte response. This response is then sent to the server.
4. The server performs the same steps using the encrypted version of the user's password stored on disk. The resulting 24-byte stream is compared with the one sent by the client to determine whether the client knew the correct password.
5. If the server's 24-byte stream and the response sent by the client match, the user session setup request (or tree connection request, in the case of share-level security) is accepted. If they don't match, the client did not know the correct password.

FIGURE 7.6
The CIFS challenge-response authentication model known as NTLMv1.



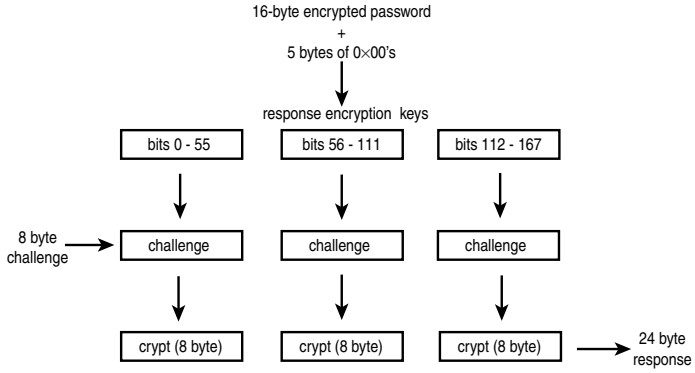


FIGURE 7.7
Generating the client's 24-byte NTLMv1 response.

Don't worry about being able to repeat the process verbatim. It is included to prove a point. The user's password is never transmitted across the network. Only data generated from the password is sent. The result is increased security.

Now back to plain-text equivalent passwords. Remember that the password always hashes to the same value. The server must store the encrypted password somewhere in order to produce the 24-byte value to verify the client's response. Therefore, if someone knows the encrypted version of the password, that person could send the correct 24-byte client response and be authenticated without ever knowing the password!



Make sure that the Samba file containing the user password hashes is kept secure and readable only by a root account.

If you decide to use password encryption, which is turned off by default, you must enable it in the `smb.conf` file by adding the following line to the global section of `smb.conf`.

```
encrypt passwords = yes
```

After enabling password encryption, you must now keep track of a second user account file. This file, usually named `smbpasswd` and stored in a subdirectory named `private` under the Samba installation directory, is where Samba stores the LAN Manager and NT hashes of user passwords. The format is very similar to `/etc/passwd`:

```
username:uid:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:account flags:lastset:
```

The `username` and `uid` fields are self-explanatory. The next two fields contain the 16-byte LAN Manager and NT hashes of the user's password, respectively. The `account flags` field determines the type of the account, such as a user account or a machine account (machine accounts are discussed more in Hour 22, "Domain Control for Windows NT 4.0/2000"). The `lastset` field records the time of the last password change.

Here's a sample entry:

```
jerry:1009:AAD3B435B51404EEAAD3B435B51404EE:
   31D6CFE0D16AE931B73C59D7E0C089C0:[U          ]:LCT-36918AD9:
```

If you decide to give the file containing the encrypted passwords a name or location other than the default, you must define the encrypted password file to be used by setting the `smb passwd file` parameter. The value should be an absolute path to the SMB password file:

```
smb passwd file = /etc/smbpasswd
```

Creating the initial SMB password file and setting passwords can be an extremely daunting task if you have a large number of existing Unix accounts. There are two common solutions to this. Both solutions require you first to create an initial `smbpasswd` entry for each user. This can easily be done using one of the scripts included with the Samba distribution:

```
cat /etc/passwd | mksmbpasswd.sh > /usr/local/samba/private/smbpasswd
```



The `mksmbpasswd.sh` script is normally located in the `source/script` subdirectory of the Samba distribution. It does not convert password from `/etc/passwd` into LanMan/NT hashes. It only creates the initial structure of the `smbpasswd` file for you.

If the Unix box obtains account information from NIS or NIS+, you can substitute the preceding `cat` command with either `ypcat` or `niscat`, depending on your system. The resulting `smbpasswd` file contains all the users from `/etc/passwd` with their LAN Manager and NT hashed passwords set to 32 X's. Samba will never authenticate a user whose password entry is set to this value.

Null Passwords in `smbpasswd`

If you want to set the value to an empty password, you must change

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

to

```
NO PASSWORDXXXXXXXXXXXXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

by issuing the following command as root:

```
root# smbpasswd -n <username>
```

The username string should be replaced with the login name of the user whose password is set to an empty value. Alternatively, you can edit the smbpasswd file with a text editor and change the string yourself. However, if you do edit the smbpasswd file by hand, make sure that the LAN Manager and NT password fields contain exactly 32 characters, no more and no fewer. If these fields do not have exactly 32 characters, Samba will not be able to correctly read the entry.



Samba provides the smbpasswd utility to manipulate entries in the smbpasswd file. This is analogous to the standard Unix command /bin/passwd, which is used to set passwords for users in /etc/passwd.

After changing the smbpasswd entry, you need to set the null passwords parameter to yes in the [global] section of smb.conf:

```
null passwords = yes
```

Harvesting Password Hashes

The main disadvantage of using Samba's smbpasswd file and password encryption is that each user will have two passwords associated with his or her account. Couple this with the fact that it is impossible to convert a password hash from /etc/passwd to a LAN Manager or NT password hash (or vice versa) and it can be a very daunting task to create a working smbpasswd file from a large group of existing Unix accounts.

How can we populate the smbpasswd file using existing information? Unfortunately, the answer is probably not what you want to hear. In the general sense, there is no way to simply convert a working /etc/passwd (including passwords) into a working smbpasswd file.

However, if your Samba server is currently configured to use clear-text authentication for clients, it is possible to gradually migrate to encrypted passwords for users. By enabling the Boolean update encrypted parameter

```
update encrypted = yes
```

smbd writes the user's LAN Manager and NT password hashes in the smbpasswd file each time the user is successfully authenticated. The only requirement is that the user has a valid entry in the existing smbpasswd file. Whatever the previous value of the user's password hash field was, it is now set to the user's current password. This method is only plausible when operating in user-level security.

Samba's default behavior is to disable the update encrypted parameter. When using this parameter, make sure that the encrypt passwords parameter is set to no.

Figure 7.8 displays how the `update encrypted` parameter works. First the client and server select to use the SMB protocol dialect, and then the client sends the username and password in plain text within the session setup request. If the user can be successfully authenticated against `/etc/passwd`, `smbd` encrypts the password and writes the information to the `smbpasswd` file. The user's entry in `smbpasswd` is never used in the validation process. It is used only as a storage facility for the encrypted password.

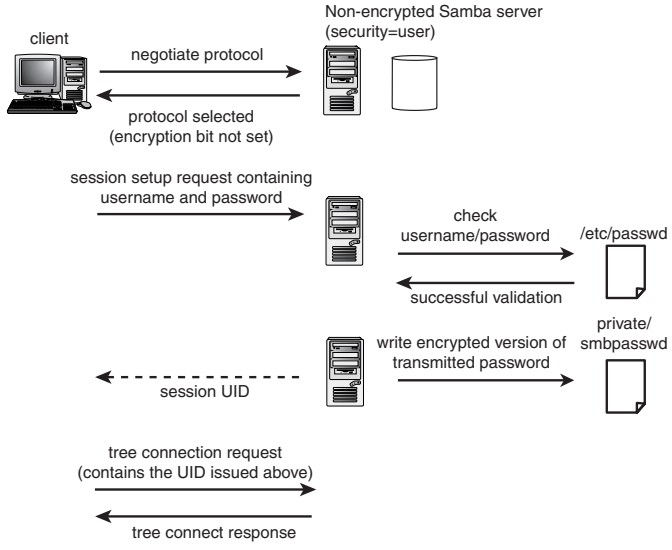


FIGURE 7.8
Implementing the `update encrypted` *parameter.*

This solution allows the Samba server to run for a few days or weeks, whatever length of time is necessary, capturing passwords and populating the `smbpassword` file. When the `smbpasswd` file contains enough entries for your satisfaction, simply change the following global parameters in `smb.conf` to switch to using encrypted passwords:

```
encrypt passwords = yes
update encrypted = no
```

Most of your users will never know that anything has changed. Windows clients begin to use encrypted passwords the next time a connection to the Samba server is established.

The following command will add an entry for the user guest2 to the `smbpasswd` file

```
root# smbpasswd -a guest2 secret
```

assuming that guest2 exists in the system's `/etc/passwd` file.

Integrating `smbpasswd` into Account Management Tools

I mentioned earlier that Samba includes a utility named `smbpasswd` for manipulating entries in the `smbpasswd` file. This tool is located in the `bin/` subdirectory and is the Samba equivalent of the Unix `/bin/passwd` program.

When a user receives a new Unix account, most sites assign a random password and then instruct the user on how to change it to something that is more memorable or personal. If you are just beginning to deploy a Samba server, it might be easiest to simply assign users an SMB password at the same time that they are issued a Unix account.

Along with the standard instructions for using `/bin/passwd` to change their Unix passwords, you should also include instructions for using the `smbpasswd` command to change Windows passwords. This is certainly the easiest solution because the responsibility for synchronizing the passwords, if desired, has been handed over to the user. However, this could result in more help desk calls, depending on the caliber of your users. It can be very easy to confuse which password goes with which logon if the accounts become out of sync; users can be very adamant in their belief that they are entering the correct passwords for their accounts. You must be the judge of which solution is best for you.

For further discussion regarding methods of synchronizing Unix and Windows passwords, refer to Hour 16, "Managing User Accounts and Single Sign-On." It is probably also a good idea to refer to Hours 10 and 11 for details of how the various Microsoft operating systems utilize clear-text and encrypted passwords.

Guest Access

This entire hour has been spent covering the models Samba uses to authenticate users. In this final section, we explore how we can avoid authentication altogether by providing guess access to users. Guest access is implemented by mapping users to a specific guest account. This account is a normal user account with very limited access to system resources such as files and printers.

The global `smb.conf` parameter `map to guest` is used to control `smbd`'s behavior when a connection request cannot be validated. This parameter accepts one of three values:

- `Never`—Samba rejects sessions with an invalid password. This is the default behavior.
- `Bad User`—If the client sends an invalid password, the session is rejected unless the username is unknown to Samba (that is, it cannot be found in the local system password file). In this case, the connection is accepted, and the user is mapped to the `guest account` specified in `smb.conf`.
- `Bad Password`—This setting causes any invalid username-password pairs to be accepted as guest connections. However, users thus connected to the guest account are not told this and might complain about being unable to access their files.



Mapping connections that transmit an incorrect password to the guest account can cause confusion for users who do not realize their mistake. They might still be able to connect to the normal set of network shares but be denied their customary level of privilege. It is best to stick with the value of `Never` or `Bad User` for the `map to guest` parameter to avoid confusion.

In conjunction with the `map to guest` global parameter, three service-level parameters mold Samba guest-access policy:

- `guest account`—The name of the Unix account that will be used for all file or printer access when an unauthenticated connection is granted guest access. The default account used for this parameter is the `nobody` account, although any valid Unix account can be used.
- `guest ok`—This Boolean parameter determines whether or not a connection that has been mapped to the guest account is allowed to access a given share at all. It is disabled by default.
- `only guest`—Should all access be done using the defined guest account? This Boolean parameter is also disabled by default.

Imagine that you want to define a read-only file share containing documentation that should be accessible to every client machine on your network. However, you know that only a handful of people have accounts on your Samba server. The easiest way to do this is to use settings similar to the ones shown in Listing 7.2. In this way, all users who have an account continue to connect to the server normally. Users who are unknown to the server are mapped to the `pcnoone` account.

LISTING 7.2 Entries in `smb.conf` for a Read-Only Guest Share

```
[global]
  map to guest = Bad User
  security = user
  ; remaining global parameters not shown here

[docs]
  guest ok = yes
  guest account = pcnoone
  path = /export/docs
  read only = yes
```

It is often a good idea to use the default guest settings unless you have a legitimate reason for changing them. If you cannot think of a legitimate reason, that's probably a good rationale to just leave them alone.

Summary

The CIFS protocol supports two modes of connection authentication. Samba supports share-mode and user-mode security and also provides the user-mode variations server- and domain-level security.

All the `security` options in Samba with the exception of domain-level security (covered in more detail in Hour 14) can utilize either plain-text passwords or encrypted passwords. Plain-text passwords are validated against the standard Unix account database (for example, `/etc/passwd` or the network equivalent). Password encryption, however, requires that Samba keep a separate file containing the password-encryption hashes.

Samba can provide guest access to users by mapping a failed authentication request to a defined guest account. The `map to guest` `smb.conf` parameter determines when, if at all, `smbd` will map a user to the guest account.

Q&A

Q **Are any external libraries needed to enable password encryption in Samba?**

A Although it was true in older versions that the administrator had to obtain an external DES library to link against, newer versions of Samba do not need this. All the source code necessary is included with the Samba distribution.

Q **Can a Samba server be configured to enable both plain-text passwords and encrypted passwords simultaneously?**

A Yes and no. A single Samba server cannot be configured to use both plain-text and encrypted passwords to validate a user in a single session connection. It is possible to use clear-text passwords to validate users from one client and the CIFS challenge-response model for users from another client. This can be configured using `smb.conf` variables and conditionally including files in `smb.conf`, which is covered in Hour 15, "Server Side Automation."

Q **Can some shares be configured to use share-level security and others on the same server be configured to use user-level security?**

A Yes and no. Samba's `security` parameter is a global option. Global parameters can be set only once per session.

Q **Does Samba support PAM?**

A Yes. The details of configuring Samba with PAM support are covered in Hour 16.

New Terms

plain-text equivalent password A password for which the encryption algorithm used always generates the same byte string given the same input; in other words, a password that always encrypts to the same value. Obtaining the encrypted version of the password enables an intruder to successfully take part in the challenge-response authentication scheme used by SMB servers such as Samba and Windows NT.

NTLM The challenge/response authentication protocol available to Windows clients and servers for validating connection requests.



HOUR 8

Samba—The File Server

Samba's file serving capability is considered by many to be the software's single most important feature. Therefore, it is no surprise that the majority of Samba `smb.conf` parameters relate specifically to file services. During this hour, we will cover many of these settings as we create the following:

- A file share that supports collaboration by members of a Unix group.
- A file share that enforces restrictions on regarding the users and groups that are allowed to access the service. This share will also support specific DOS semantics such as DOS attributes and timestamps.
- A public share similar to an anonymous FTP server.
- A special share that allows users access to a personal home directory.

Our goal is to divide up the parameters into smaller groups based upon their functionality. This will help us to focus on the particular feature we are implementing without becoming distracted by non-essential settings. You might want to dog-ear this hour and use it for later reference as well.

Beginning Configurations and Accounts

We have already covered some of the basic parameters needed to create a Samba file server in Hours 4 and 5. Some of these will be repeated here for the sake of completeness.

To begin, we must define the basic `[global]` settings for our server. Listing 8.1 displays a bare-bones `smb.conf`, onto which we can append new file shares as they are needed.

LISTING 8.1 Beginning `smb.conf` for New File Services

```
## Sample smb.conf file for Samba as
## a File Server
##
## Sam's Teach Yourself Samba in 24 Hours
## <jerry@samba.org>
[global]
     netbios name = POGO
     workgroup = STY-SAMBA
     security = user
     encrypt passwords = yes
```

We have chosen to use encrypted passwords on our Samba server in order to avoid some of the problems with plain text passwords described in Hours 10 and 11. This means that we will need an account that can be used in our tests and examples. All configurations in this hour will assume the existence of the following users in the local `/etc/passwd` file, each of which has a corresponding `smbpasswd` entry.

```
jerry:x:780:100:Gerald Carter:/home/gcarter:/bin/bash
sam:x:781:100:Sam Samba:/home/sam:/bin/bash
kristi:x:782:900:Kristi Carter:/home/kristi:/bin/bash
```

If you need a reminder on how to create an `smbpasswd` file and add users to it, see the examples in the previous hour, "Security Levels and Passwords."

The group ids (`gid`) 100 and 900 used in the primary group field for these users are defined in `/etc/group` as

```
users:x:100:
guest:x:900:
```

A Basic, Group Accessible Share

Our first file share in this hour will be a basic service designed to support coordination by members of a single Unix group. This is a similar scenario to the one used in Hour 4, when we built and installed our first Samba server. There will be a small amount of repetition with examples from previous hours, but on the whole, we will pick up where Hour 4 left off.

We begin with a simple share named `[24hrs]` that allows members of "`users`" to have read-write access to the directory. To set up this directory on disk, we will create the folder and assign its permissions to be `rwx` for the owner (`root`), `rwx` for the group owner (`users`), and `---` for everyone else.

```
root# mkdir /export/24hrs
root# chgrp users /export/24hrs
root# chmod 770 /export/24hrs
```

Next, we will add the service definition to `smb.conf`.

```
## group file share
[24hrs]
     comment = Basic group share for STY-Samba users
     path = /export/24hrs
     browseable = yes
     read only = no
```

Samba always grants the most restrictive set of permissions to users. Based upon the permissions assigned to `/export/24hrs`, only members of the Unix "`users`" groups will be able to access the share, even though the service definition would allow any authenticated user to establish a connection. If the user `kristi` attempted to connect to `\\pogo\24hrs`, Windows would report that the "network name could not be found," or that the "shared directory cannot be found."

Table 8.1 is a quick reminder of the meaning of each parameter used in the `[24hrs]` service. The default value is also described if the option is not explicitly set.

TABLE 8.1 Basic File Service Parameters

Name	*Default*	*Description*
`browseable`	`yes`	Should the service be displayed in the list of enumerated shares on the server? If it is disabled, clients must know the UNC path to the share in order to establish a connection to it.
`comment`	`""`	Defines the string displayed when viewing the properties of the share.
`path`	`/tmp`	Specifies the directory on disk to be shared.
`read only`	`yes`	Should `smbd` allow clients to write to the share? This is an inverse synonym for the `writable` parameter.

The `create mask` and `directory mask` parameters were briefly introduced in Hour 4. There we used these settings to ensure that a file was not created with permissions that

allowed world access. In other words, a `create mask` and a `directory mask` represent those bits which will be removed from the final permissions. There are complementing `force create mode` and `force directory mode` parameters that can be used to specify permission bits that will always be present. All of these parameters are summarized along with their default values in Table 8.2.



When changing permissions or creating new file systems objects, smbd always sets the read (`r`) bit for the owner of a file and the read (`r`), write (`w`), and execute (`x`) bits for the owner of a directory, regardless of the value of the create mask or force create mode parameters. This means that it is impossible for a user to lock himself out of a directory or file.

By adding the following four lines to the `[24hrs]` definition

```
create mask = 0660
force create mode = 0660
directory mask = 0770
force directory mode = 0770
```

we are guaranteeing that all files created in the `[24hrs]` share will have the permissions `rw-rw----` and that all directories will have the permissions `rwxrwx---`. It will be impossible for a user to create a read-only file, either for herself or for her group.

TABLE 8.2 Parameters That Affect the Permissions on New Files/Directories

Name	*Default*	*Description*
`create mask`	`0744`	Unset bits (`0`) are removed from the permissions on new files.
`directory mask`	`0755`	Unset bits (`0`) are removed from the permissions on new directories.
`force create mode`	`0000`	Set bits (`1`) are added to the permissions on new files.
`force directory mode`	`0000`	Set bits (`1`) are added to the permissions on new directories.
`inherit permissions`	`no`	Should new permissions assigned to files and directories be controlled by those set on the parent directory?

The first four parameters presented in Table 8.2 are applied to all new files and directories within a share. If finer-grained control of new permissions is required, the `inherit permissions` parameter can be used. This parameter allows files to inherit the `read` and `write` bits from the parent directory and new folders to inherit the `read`, `write`, and `execute` bits. The `group id` bit is propagated as well, but the `setuid` bit is never inherited.

A quick example will help to make this clearer. If the parent folder, `/export/test`, has permissions

```
$ ls -ld /export/test
drwxrwsr-x    7 root     root    207 Sep 15 14:48 /export/test
```

then any files created below this directory would have permissions `rw-rw-r--`. Any folders would have the permissions `rwxrwsr-x`.

The entire process of determining the final permission set for a new file is rather complicated. It can be summarized as a set of four steps.

1. First, any requested DOS attributes are mapped onto the permission set. The heuristics for this step are described later in this hour.
2. Any inherited permissions are applied.
3. The `create mask` parameter is applied.
4. The `force create mode` parameter is applied.

Most problems arise when attempting to represent too many attributes using a finite number of bits. Samba 2.2 does support many file system dependent Access Control Lists (ACL). We will cover ACLs and how to configure Samba to interact with these systems in Hour 14, "Replacing a Windows NT File and Print Server—Lock, Stock & Barrel."

Differences Between File Systems on Windows and Unix

All file systems commonly used on Windows clients (for example, FAT16, FAT32, and NTFS) are case preserving and not case sensitive with respect to filenames and directories. Unix file systems, on the other hand, such as `ext2`, `ufs`, `vxfs`, and `xfs`, are case preserving and case sensitive.

It is Samba's responsibility to bridge this gap for its users. If a Windows application attempts to open a file named "`My File.ext`," `smbd` must scan the directory for all variations, such as "`my file.ext`" and "`MY FILE.EXT`." Developers have provided several parameters that can be used to tune `smbd`'s behavior in this situation.

By default, `smbd` will perform non case sensitive searches when looking for a file requested by a Windows client. This behavior can be disabled by setting the `case sensitive` `smb.conf` option.

```
case sensitive = yes
```



Enabling the `case sensitive` parameter can break many Windows applications. You should never need to set this option.

Three of the most common case options are

- `preserve case`
- `short preserve case`
- `default case`

When a Windows client creates a new file or directory on a Samba file share, `smbd` must determine what the actual filename on disk should be. The first two parameters define what to do with long filenames and with DOS 8.3 filenames, respectively. By default, Samba will always preserve the case of filenames. If one of these parameters is disabled, `smbd` will convert all corresponding filenames into the `default case`. Table 8.3 gives a summary of the case settings and their default settings.



An 8.3 filename is composed of one to eight uppercase characters followed by a period followed by one to three uppercase characters in the file extension. An example would be `DEFAULT.TXT`.

Service level parameters can be placed in the `[global]` section of `smb.conf` to define configuration-specific default values. It is a good practice to specify a `default case` policy by setting these parameters globally. These are settings that I commonly use on a production server.

```
[global]
     ...
     ## define a global case policy for file shares
     case sensitive = no
     preserve case = yes
     short preserve case = no
     default case = lower
```

TABLE 8.3 Samba's Case Parameters

Name	*Default*	*Description*
`case sensitive`	`no`	Should `smbd` perform non case sensitive file/directory lookups?
`preserve case`	`yes`	Should `smbd` preserve the case of long filenames?
`short preserve case`	`yes`	Should `smbd` preserve the case of short (8.3) filenames?
`default case`	`lower`	What case should filenames that are not preserved be converted to (lower or upper)?

Many users still require old 16-bit and DOS-based applications. Frequently, these programs cannot deal with long filenames. Unlike FAT and NTFS, Unix file systems do not store an associated 8.3 filename with a normal, long filename. Instead of storing this information on behalf of the file system, Samba generates this short filename as needed. Table 8.4 presents several parameters that control how `smbd` generates this new string. We will use the default values for all of these mangling parameters in the `[24hrs]` service.

TABLE 8.4 Case-Mangling Parameters

Name	*Default*	*Description*
`mangle case`	`yes`	Should Samba generate mangled (8.3) names for filenames that are not in the default case? This is the only way to see both Mail and mail in the same directory.
`mangled names`	`yes`	Should Samba generate 8.3 filenames at all? If not, files/directories with long filenames will be unavailable to DOS clients.
`mangling char`	`~`	What character should be used as a stand-in for the characters replaced to create the 8.3 mangled name?

A Restricted Access Service

Our next file share, named `[software]`, will be a variation on the group share presented in the previous section. Initially, our file service definition appears very similar to the `[24hrs]` share.

```
## file share for distributing a specific application
[software]
     comment = Limited access share for running Foo.exe
     path = /export/foo
     read only = no
```

However, we have different requirements for access control in this section. This share should be accessible only to the users `jerry` and `kristi`. Our problem is that these accounts are not members of the same group, so the solution used for the `[24hrs]` share, delegating the access control solely to the file system, will not work.

Instead, we will revert to the solution used in Hour 4. In that example, the `force group` (see Table 8.5) parameter made `smbd` perform all file system access as a specific group rather than as the authenticated user's primary group. Rather than using a group account for access, we will use the `force user` parameter in its place.

First, we will create a new user account specifically for this share.

```
software:x:783:900:Samba account:/dev/null:/bin/false
```

Next, we will change the ownership of all files in `/export/foo` to this account.

```
root# chown -R software /export/foo
```

Finally, we will add the new `force user` value to the `[software]` definition.

```
[software]
     comment = Limited access share for running the Foo.exe
     path = /export/foo
     read only = no
     force user = software
```

Even in the face of the `force user` option, the user must still be authenticated using his or her login name and password. The `force user` parameter does not take effect until after the user has been validated.

TABLE 8.5 Forcing File Access as a Specific Account or Group

Name	*Default*	*Description*
`force group`	`""`	Define the group id to be used for all file access in the place of the user's primary group.
`force user`	`""`	Define the user id to be used for all file access.

There is a problem with our current share. It does not restrict access to the two accounts we specified, `jerry` and `kristi`. In fact, file system permissions are ineffective for controlling access in this case. Any user who connects to `[software]` will have full control over all the files it contains because of the `force user` parameter.

In order to correctly control our share and to meet our initial requirements, we need to introduce a collection of options for restricting access from users and hosts. A summary of these parameters is given in Table 8.6.

TABLE 8.6 Parameters for Controlling Access to Shares

Name	*Default*	*Description*
`admin users`	`""`	A list of users and/or groups for which `smbd` will perform file operations as root. Be very careful with this option.
`hosts allow`	`""`	Analogous to the access control by the `/etc/hosts.allow` file used by Wietse Venema's TCP wrappers package. This parameter will be covered in more detail in Hour 17.
`hosts deny`	`""`	Analogous to the access control by the `/etc/hosts.deny` file used by Wietse Venema's TCP wrappers package. This parameter will be covered in more detail in Hour 17.
`valid users`	`""`	A list of users and/or groups that are allowed to access the share if (1) they are successfully authenticated and (2) the file permission access checks are passed.
`invalid users`	`""`	A list of users and/or groups that will never be allowed to establish a successful connection to the share.
`max connections`	`0`	Specify a maximum number of simultaneous connections. A value of `0` indicates no maximum limit.
`read list`	`""`	A list of users and/or groups that will only be allowed read access regardless of the file system permissions.
`write list`	`""`	A list of users and/or groups that should be given write access even if the `read only` parameter has been enabled.

Many of the parameters presented in Table 8.6 accept a list of users and/or groups. The syntax of the list warrants a little explanation.

- Items in the list are separated either by spaces or by a comma.
- The list can contain usernames that can be validated by the operating system (that is, Unix account names such as `kristi`).
- Groups can be prefaced by (1) '`@`' to instruct `smbd` to look the name up as an NIS/YP netgroup first and as a Unix group if the previous query failed, (2) '`+`' to indicate that the group membership should be looked up in the list of Unix groups, and (3) '`&`' to indicate that the group membership should be expanded by searching the group map for the server's NIS/YP domain.

To ensure that Samba will restrict access to the `[software]` share for the two named users, `jerry` and `kristi`, we will add a `valid users` setting to the service definition.

```
[software]
     comment = Limited access share for running the Foo.exe
     path = /export/foo
     read only = no
     force user = software
     valid users = jerry, kristi
```

We will continue to use the parameters in Table 8.6 throughout this hour and the remaining hours in this book.

File Share Access Control Lists

There is a means of placing access restrictions on file shares besides the parameters presented in Table 8.6. Samba 2.2 introduces the capability for administrators to define true Windows NT-style ACLs and assign them to any file share. This should not be confused with Samba's ability to change file system permission bits or the support for file system ACLs, which will be covered in Hour 14. File share ACLs are stored in a TDB maintained by Samba (`<lock directory>/share_info.tdb`) and are available without any extra compile time options (for example, the `--with-acl-support` option is not required).

Manipulating a file share ACL can be done using a tool such as the Windows NT 4.0 Server Manager shown in Figure 8.1 and only when connected to the Samba server as root.

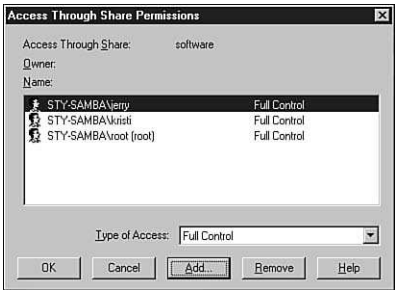


FIGURE 8.1
Modifying a file share ACL using the Windows NT 4.0 Server Manager.



Releases of Samba prior to version 2.2.2 had a bug that failed to allow a root user to modify file share permissions if the root account had been removed from the ACL. The only workaround was to remove the `share_info.tdb` file and start over. Just to be safe, always include the root user with `Full Control` as part of the ACL on a share.

Samba's file share ACLs are entirely separate from parameters such as `valid users` or `write list`. Realizing that a share ACL is checked after all of the `smb.conf` checks have been performed is the key to understanding the interaction between a share ACL and the

service's corresponding `smb.conf` definition. This means that settings for `valid/invalid users`, `max connections`, and `hosts allow/deny` take precedence over the ACL entries.

Samba again uses a filter approach to applying security settings. A user will receive the most restrictive access after taking the union of the file system permissions, the service definition, and the share ACL. For example, a user can be given write access in `smb.conf` but actually be prevented from writing to files or creating directories by either (a) the file system permissions or (b) an entry in the ACL that restricts write access. In its default state, a file share ACL assigns both read and write access to `Everyone` and therefore has no affect on security.

DOS Attributes and Timestamps

As surprising as it may seem, many Windows applications rely upon support for certain DOS attributes. Because non-DOS file systems have no concept of these bits, Samba again attempts to bridge the gap between Windows and Unix. There are four DOS attributes that Samba supports.

- Archive (`A`)——Samba uses the owner's `execute` bit on a file to represent the DOS archive bit.
- Hidden (`H`)——Samba represents the `H` attribute using the file's world `execute` bit.
- Read Only (`R`)——Samba represents this by removing the `write` bit from the owner's permission set on a file. It is not possible to set the `R` attribute on a directory.
- System (`S`)——This DOS attribute is represented using the group `execute` bit on a file.

The `A` and `R` attributes are the only ones supported by default Samba installations. Table 8.7 presents four `smb.conf` parameters that control Samba's behavior regarding DOS bits.

TABLE 8.7 Samba's DOS Attribute Relate Parameters

Name	*Default*	*Description*
`delete read only`	`no`	Should `smbd` allow a user to go against normal Unix semantics and delete a read-only file?
`map archive`	`yes`	Should Samba map the DOS archive bit onto the user `execute` bit on a file?
`map hidden`	`no`	Should Samba map the DOS hidden bit onto the world `execute` bit on a file?
`map system`	`no`	Should Samba map the DOS system bit onto the group `execute` bit on a file?

The main problem that occurs when mapping DOS attributes onto Unix permission is that Samba can run out of bits to represent data. This solution can also be derailed if users are able to reset permissions outside of Samba such as with the `chmod` command. Samba's solution in this case is a best-effort solution. You should be aware of this risk and prevent users from modifying permission bits unnecessarily if it can result in broken Windows applications.

The `dos filemode` and `dos filetimes` parameters are used to make Samba file shares more intuitive to Windows users. Under normal Unix semantics, only the owner of a file is able to change its permissions or timestamps. In the world of FAT file systems and Windows 9x clients, this can be confusing, since a user may be able to write to the file but not set attribute bits. Table 8.8 describes these two options, which grant a person the ability to change these file characteristics if the user has write access to a file.

TABLE 8.8 DOS Permission and Timestamps

Name	*Default*	*Description*
`dos filemode`	`no`	Should `smbd` allow a user to change the permissions on a file if the user has write access to it?
`dos filetimes`	`no`	Should `smbd` allow a user to change the timestamps on a file if the user has write access to it?

Neither of these parameters is of any use to the `[software]` service, because all accessing is done as a single user. This user owns all files and directories contained in the share and can therefore change whatever he or she desires.

A Publicly Accessible Share

A common question by new Samba administrators is "How do I set up a file share that is available for public access?" During this section, we will build a share that mimics an anonymous FTP server minus the writable upload directory for guests. People who have been authenticated are allowed access with their individual accounts. All other users are mapped to a specified guest user.

First, we will create the initial service definition as we have done in the past.

```
## guest accessible share for distributing public files
[public]
     comment = public file share
     path = /export/public
     read only = no
```

The permissions on `/export/public` have been set to `rwxrwxrwxt` by executing

```
root# chmod 1777 /export/public
```

This will allow users to create files and directories but will restrict them from removing objects that they do not own.

In Hour 7, we covered the basics of how Samba controls guest logins using the `map to guest` parameter. This option, along with the remaining service level guest parameters, is described in Table 8.9. For our server, we will choose to map logins to the guest account when the client sends us an unknown username. This will prevent unnecessary confusion for users who have a hard time entering their password correctly. The additional `map to guest` parameter has been added to the `[global]` section of `smb.conf` here:

```
[global]
     netbios name = POGO
     workgroup = STY-SAMBA
     security = user
     encrypt passwords = yes

     ## guest access settings
     map to guest = Bad User
```

TABLE 8.9 Guest Related Parameters

Name	*Default*	*Description*
`map to guest`	`Never`	This global parameter controls the circumstances under which `smbd` will map a user login to the `guest account`—`Never`, `Bad User`, `Bad Password`. Refer to Hour 7 for the details on each of these values.
`guest ok (public)`	`no`	Should Samba allow a user who has been mapped to the `guest account` to access this share?
`guest account`	`nobody`	What Unix account should be used when mapping a user to a guest login?
`guest only`	`no`	Should Samba only allow guest access to this share? If enabled, `smbd` will map all users, including previously authenticated ones, to the guest account when accessing this service.

Once `smbd` has decided that a session should be mapped to a guest login, each service is required to specify the Unix account that should be used to represent this guest user. A common choice is to create a user for the sole purpose of acting as a Samba guest and

defining this as the default `guest account` for all shares. The following user has been added to `/etc/passwd` solely for this purpose.

```
guest1:x:783:900:Samba guest account:/home/guest1:/bin/bash
```

Next, we will add a server-wide default `guest account` by specifying this account in the global section of our `smb.conf`

```
[global]
     netbios name = POGO
     workgroup = STY-SAMBA
     security = user
     encrypt passwords = yes

     ## guest access settings
     map to guest = Bad User
     guest account = guest1
```

To finish configuring `[public]` for anonymous access, we must enable the share to allow access from guest connections by adding `guest ok = yes` to the service definition.

```
[public]
     comment = public file share
     path = /export/public
     read only = no
     guest ok = yes
```

In order to prevent a user mapped to the `guest1` account from creating files or directories, we can add the guest account to the list of users who are not granted write access.

```
     read list = guest1
```

The final semantics of the `[public]` share is

> Authenticated users are able to add files and directories to the top level of the share and in any directory that they own. By default, guest users are able to read all files contained in the share, but are not able to create new files or write to existing ones.

Modifying Unix Permissions from Windows Clients

By reviewing the default values for the `create mask` and `directory mask` options in Table 8.2, we know that all files created in `[public]` should be viewable by all other users. Perhaps there are particular files that should not be viewable by all users. In the spirit of making our service as flexible as possible for the user, Samba allows Windows clients to view and set Unix permission bits using the security tab of the Windows Explorer.

Assume that we have created a file named `example.txt` in the root of the `[public]` share with the permissions

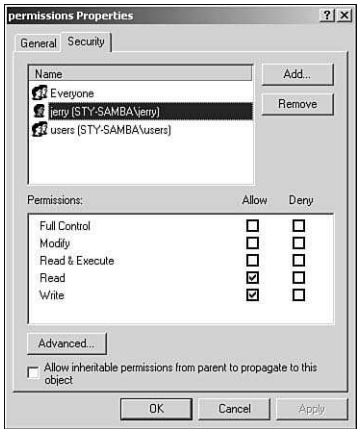
```
-rw-rw-r--    1 jerry    users    0 Sep 23 22:19 /export/public/example.txt
```

Figure 8.2 shows how the permissions of this file would appear in the Windows 2000 Explorer.



Samba reports no permissions as the Take Ownership (`O`) bit. For example, if a file does not grant any permissions to the other group (that is, `---`), Windows NT will display the group as the Everyone built-in group, having "Take Ownership" permission on the file. This does not mean that any user can actually change the ownership of a file. The "Take Ownership" action can only successfully be performed by a client connected to the server as the root account.

FIGURE 8.2
Viewing the permissions of `example.txt` *from a Windows 2000 client.*



Samba's `smb.conf` file includes five parameters for restricting those permission bits that can be modified by users and those that must always be enabled on files and directories. The options, summarized in Table 8.10, are similar to the `create mask` family of parameters given in Table 8.2.

TABLE 8.10 Permission Mask Parameters

Name	*Default*	*Description*
`nt acl support`	`yes`	Should Samba report that it supports persistent NT ACLs to Windows clients?
`security mask`	`0777`	Any bits set to `0` in this value can never be set for a file's permissions.
`directory security mask`	`0777`	Any bits set to `0` in this value can never be set for a directory's permissions.

continues

TABLE 8.10 Continued

`force security mode`	`0`	Any bits set to `1` in this value will always be enabled in the file's permissions.
`force directory security mode`	`0`	Any bits set to `1` in this value will always be enabled in the directory's permissions.

Although it makes sense to allow a user to restrict access to a file by disabling all of the world access bits, it is probably not a good idea to allow users to make files world writable. The following `security mask` related settings prevent users from setting the other write bit on files and directories.

```
security mask = 0775
directory security mask = 0775
```



Samba will never remove the owner's "`r`" bit on files or the owner's "`rwx`" bits on a directory.

Share Mode, Deny Modes, and Oplocks

We will conclude our examples using the `[public]` by discussing file locks and CIFS caching mechanisms. Things like open modes, deny modes, and file leases are often foreign concepts to Unix administrators. Some of these techniques have been added to the next revision of the NFS protocol (v4), but their success in that area has yet to be determined.

When a CIFS client attempts to open a file, it requests two access related parameters. One, the open mode, for defining its access, and another, the deny mode, for specifying the access granted to other clients. The open mode is analogous to the normal `READ_ONLY/READ_WRITE/WRITE_ONLY` flags used to open a file. The deny mode specifically requests that the server deny certain access, such as `DENY_NONE` and `DENY_READ`, to other CIFS clients. Samba's `smbstatus` tool is able to print the current share modes and deny mode for a file. The `-L` switch is used to only view information on locked files.

```
root# smbstatus -L
Locked files:
Pid    DenyMode   R/W        Oplock           Name
--------------------------------------------------
19098  DENY_NONE  RDWR       EXCLUSIVE+BATCH  /home/pogo/gcarter/test.doc

Sun Sep 23 22:52:49 2001
```

Here, the file `test.doc` has been opened for read/write access. The client has not specifically requested that other clients be denied any access. It is the server's responsibility to coordinate lock requests on files between multiple clients.

The `smbstatus` output also displays that the client has an exclusive oplock on `test.doc`. An *oplock,* short for opportunistic lock, is an aggressive caching mechanism used by CIFS clients to enhance performance. Some benchmark tests have indicated that oplocks can increase the performance of certain applications by up to 30%. A discussion of how oplocks are implemented is lengthy and beyond our current scope. For full details, you will need to refer to the Samba source code and the current CIFS documentation.

There are three types of oplocks defined by the existing CIFS/SMB documentation.

- Exclusive Oplocks
- Batch Oplocks
- Level II Oplocks

In practice only Exclusive and Level II oplocks are used across a network. An exclusive oplock is requested by a client in order to cache the file for local reads and writes. By definition, there can be only one exclusive oplock on a file at any given time. If the client ever receives an oplock break request from the server, the client must immediately flush all cached writes back to the share. This break request will only occur when another client tries to open the cached file. The server can then ensure that multiple clients see a consistent view of the file.

Windows NT/2000/XP clients support a read-only file caching mechanism called a Level II oplock. This type of oplock is most useful for executables (that is, `*.exe` files) and other files that should never be modified. There can be several outstanding Level II oplocks on a file at any given time, assuming that no client has gained an exclusive oplock on the file.

One of the problems with past releases of Samba is that the oplock enforcement controls were internal to Samba's `smbd` daemon. This meant that other Unix processes were unaware of a file cached by a CIFS client. This is particularly a problem when a file is accessed from both NFS and CIFS concurrently. The `kernel oplock` parameter is used to enable support for registering file leases (that is, oplocks) with the kernel so that all Unix processes, including `smbd`, can have a consistent view of file data. Kernel level oplocks are currently supported only by recent releases of Irix and the Linux 2.4 kernel.

TABLE 8.11 Oplock Parameters

Name	*Default*	*Description*
`kernel oplocks`	`yes`	Should Samba attempt to register oplocks on files with the kernel? This parameter is effective only if the server's operating system supports kernel level file leases (for example, Linux 2.4 and Irix).
`Level II oplocks`	`yes`	Should Samba support CIFS level II oplocks? Relevant only if `oplocks = yes`.
`oplocks`	`yes`	Should Samba support exclusive oplocks?
`veto oplock files`	`none`	Define a set of filenames or wildcards for which Samba should refuse to grant oplocks.

Table 8.11 described Samba's oplock related parameters. Normally, these default values will never need to be changed. However, in the absence of kernel oplock support, it might be prudent to disable the oplocks parameter for a share if clients will be accessing the directory tree outside of Samba.

User Home Directories

The final file service we will examine during this hour is a special share for providing users access to a personal home directory. The `[homes]` share is a built-in section like the `[global]` section presented in Hour 5 and the `[printers]` service, which will be covered in Hour 9, "Samba—The Print Server."

To understand how the `[homes]` service can be used, imagine the following scenario:

1. Samba receives a request from a user to connect to `\\POGO\user1`.
2. After verifying that the server's name in the UNC path matches its own name (for example, `POGO`), Samba searches the shares in `smb.conf` for an explicit definition of a service named `[user1]`. If a match is found, the connection request is processed using that share. The process then skips to step 6.
3. If the `[homes]` share is defined, `smbd` will search the local set of user accounts for a match to the login name "`user1`." If a match is found, a copy of the `[homes]` service is created and renamed to "`user1`." The connection request is then processed using this share. The process skips to step 6.
4. If there exists a definition for the `[printers]` share, Samba will attempt to locate a match for the login name "user1" in the file specified by the `printcap` name parameter. The details of this are given in Hour 9. The connection request is then processed using this share. The process skips to step 6.

5. If there is a default service defined in the configuration file, replace the share name in the tree connection request with this value and use it when processing the request.
6. Return the result of the SMBtcon&X request to the client.

The [homes] service can contain any of the normal file share parameters covered so far. In order to see how the process works in practice, we will add a new definition to our smb.conf.

```
## Export users' home directories
[homes]
     comment = home directory for %u
     path = %H
     read only = no
```

If Samba receives a connection to \\POGO\sam, it will first search for an explicit share definition of [sam] in smb.conf. This will fail because we have defined only the [24hrs], [software], [public], and [homes] shares. Next, because the [homes] share does exist, smbd will search the local /etc/passwd file for a user named "sam." This lookup will succeed. Therefore, Samba will create a new share dynamically by copying the [homes] service and renaming it to [sam].

```
[sam]
     comment = home directory for sam
     path = /home/sam
     read only = no
```

The values for the %u and %H variables have already been filled in to make it more apparent what their final values will be. Assuming that the requesting client has permissions to access Sam's home directory, the connection will then be allowed.

A simple means of preventing a user from connecting to a home directory other than his own is to add

```
     valid users = %S
```

to the [homes] share. The %S variable is expanded to the name of the current service so our new version of [sam] would appear as

```
[sam]
     comment = home directory for sam
     path = /home/sam
     read only = no
     valid users = sam
```

This now states, "Only the user sam can connect to the share named [sam]."

Symbolic Links

Samba does not support the creation of arbitrary symbolic links on the server's file system by Windows clients. This does not prevent users from creating such links if the user is able to access the shared directory without going through `smbd` (for example, a shell prompt). In this case, Samba provides two parameters to control how symbolic links are handled.

The `wide links` option determines whether or not Samba should follow links outside of the shared directory. The `follow symlinks` parameter can be used to prevent Samba from following symbolic links at all, inside or outside the directory tree. Both of these settings are presented in Table 8.12, along with their default values.

TABLE 8.12 Parameters for Controlling Access to Symbolic Links

Name	*Default*	*Description*
`follow symlinks`	`yes`	Should Samba honor symbolic links at all?
`wide links`	`yes`	Should Samba follow symbolic links to directories outside of the current shared directory tree? This does not apply to symbolic links to files outside of the shared directory tree.

A common situation where it is advisable to disable the `follow symlinks` parameter is when a file server is exporting directories via both NFS and CIFS (Samba). A user who logs on to a Unix client that mounts his home directory via NFS is able to create a symbolic link to the system's `/etc/passwd` file. This in itself is not entirely helpful to the user because if he is able to access a shell prompt on the Unix client, then the file can be viewed via normal tools such as `vi` or `cat`.

However, as is pointed out in Figure 8.3, if Samba honors the symbolic link, the user is now able to access the `/etc/passwd` file on the Samba server from a Windows client. In this case, setting `follow symlinks = no` enforces the security policy of not allowing a user to gain unnecessary information about the configuration of the Samba/NFS server.

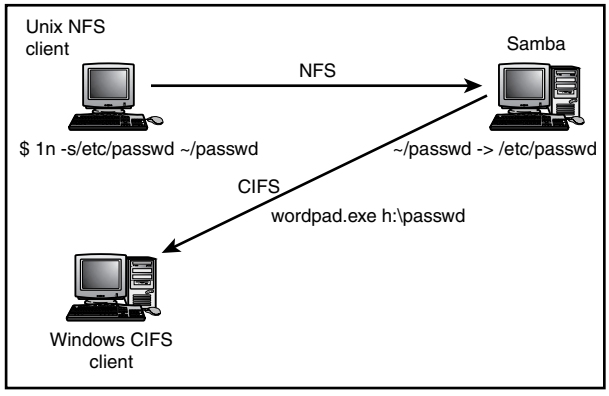


FIGURE 8.3
Using symbolic links to view `/etc/passwd` *on a file server.*

File System Quotas

Samba includes support for interacting with some file systems quota implementations, such as Linux 2.2 and 2.4 quotas and Veritas VXFS quotas. Quotas are a means of restricting the amount of disk space consumed by a particular user or group. This can be useful if your users lack self-control when it comes to downloading mp3 files to their home directory.

Samba does not implement quotas internally. This is done entirely by the file system. Samba's support for quotas only involves correctly reporting the available disk space to a user.

This support is a compile time option. The `--with-quotas` option must be passed to the `./configure` script to enable the internal code to correctly interact with a quota-enabled file system.

```
$ ./configure --with-quotas
<...output deleted...>
checking whether to support disk-quotas... yes
<...output deleted...>
```

There are no `smb.conf` parameters specifically related to file system quotas, nor are any of the parameters covered here affected by enabling Samba's quota support.

You will need to refer to your systems documentation to determine whether or not the operating system supports quotas and, if so, how to enable them.

Summary

During this hour, we have covered many of Samba's file share related parameters. These options have been grouped according to functionality and have helped us to create file services to

- Give access to a share of a specific Unix group.
- Provide group access to a share for users who are not members of a common Unix group.
- Support DOS attributes and access semantics.
- Provide public access to data.
- Provide access to personal home directories for users.

Finally, we examined how to enable Samba's support for file system quotas and what this support really means.

Q&A

Q **My `smb.conf` file has a file share called `[docs]` in it. This share is intended for document writers to save their documents in. The directory has a group owner of `docs`, which all the document writers are in, and the mode of the directory is `0770`, but no one can write into the directory. What have I done wrong?**

A Do you have one of the following parameters in the share definition?

```
writable = yes
writeable = yes
read only = no
```

Remember, a share is read-only by default, and you have to make it writable before writes are allowed, regardless of directory or file permissions in the share.

Q **I have defined a new share called `kits`, but no one can connect to it. They keep getting a message: "The specified share directory cannot be found." Other users get a dialog box saying, "Cannot access `\\server\kits...`." What could the problem be?**

A Check the path statement in your `[kits]` share section. If the path does not exist or the users do not have access to it, they can get these sorts of messages.

Q **What is the smallest `[homes]` section you can get away with? Hint, if the share is not browseable, it doesn't really need a comment. Also, remember that the default value for the path parameters in a `[homes]` section is `%H`.**

A You need to make the share writable, so users can at least write into their home shares and so the smallest `[homes]` section has two lines. For example:

```
[homes]
     read only = yes
```



HOUR 9

Samba—The Print Server

If there were one feature that could be noted as changing the most between Samba 2.0 and Samba 2.2, it would be the printing support for Windows clients. An old adage says that with great power comes great responsibility. In the case of Samba, this would be translated to: the more features software implements, the more complex it will be to configure.

If you have never configured a Samba 2.0 server, you will have a small advantage over those who are upgrading to Samba 2.2 because it will not be necessary to unlearn certain configuration steps. Samba now supports the Windows NT/2000 set of Remote Procedure Calls (RPC) used for spooling print jobs, managing printers, and downloading print drivers to clients. As a result, clients running Windows NT/2000 with a Samba 2.2 server will behave in an entirely different manner from clients running Windows NT/2000 with a Samba 2.0 server. If all of your clients are running Windows 95, 98, or ME, on the whole, your clients will behave the same with both Samba 2.0 and Samba 2.2.

During this hour, we will explore Samba's support for sharing printers and how to manage these printers from Windows clients. We will also learn how to configure Samba to provide printer drivers that can be downloaded to clients as needed without using intervention. Because many `smb.conf` parameters are common to both file shares and printer services, you should be comfortable with the information presented in the previous two hours, "Security Levels and Passwords" and "Samba—The File Server," before we begin.

Prerequisites

In order to minimize confusion, it is necessary to understand what will and what will not be covered during this hour. We will examine how to use Samba to make existing printers available to Windows clients. We will not cover any specifics for installing printers on a Unix/Linux system.

This means that a prerequisite for the hour is to have a working printer already installed on the Samba server. It is a good idea to verify that it is functioning properly before preceding. Samba's printing philosophy is, "If Unix can print to it, so can I." The converse of this is also true, "If Unix cannot print to it, neither can I."

Given the variety of printer models, Unix systems, and seemingly infinite combinations of the two, it is impossible to outline a quick method for initially testing the server's printing system. However, a common test case is to attempt to print the local `/etc/hosts` file on the server. For `lpd`-based systems, this can be done by executing

```
$ lpr -P<printer name> /etc/hosts
```

The <printer name> should be replaced with the name of the printer that you are testing. If there are any errors or failures at this point, refer to your server's documentation for the necessary steps to resolve the problem.

Samba's Printing Support

Samba's printing support is best described as a spooling system. From the point of view of a CIFS client, printing involves opening a file on a special file share, writing to it, and then closing that file. What happens to the file after the client has closed it is of no concern to the client, but users generally would like such files to be printed. When you configure a print share under Samba, certain parameters, which will be covered shortly, define what is actually done with the file once it has been spooled by the client.

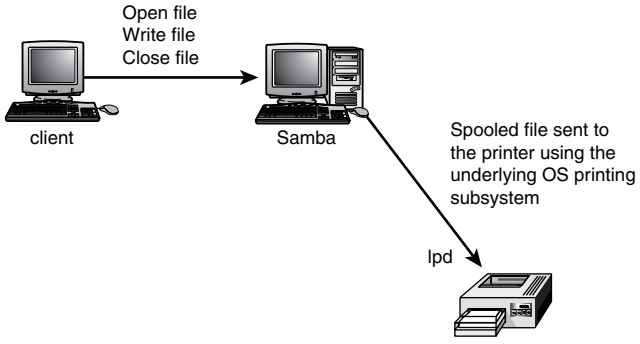


FIGURE 9.1
Sending a file to a printer on a CIFS server.

Figure 9.1 gives a chronological overview of how a file is printed through Samba. The steps taken by a client are

1. Open a file for writing on the printer share. The server must provide file system space where this file can be stored.
2. The client writes the print file. A Windows 9x/ME client uses normal CIFS operations for this step, whereas a Windows NT/2000 client uses RPC operations.
3. The client closes the file when all data has been written.
4. The server sends the spooled file to the underlying printing system.

Though it is easy to describe how a client prints to a Samba printer once it has been configured, it is a little more involved to explain how to configure Samba's new printing functionality. This becomes much easier when you understand that Samba 2.2 is designed to function exactly as a Windows NT 4.0 print server. The implication of this is that connecting to a Samba printer from a Windows (or OS/2) host is much easier if the necessary client drivers for the printer have been installed on the Samba server beforehand.

Though this may seem like an unreasonable requirement, it is actually a limitation of Windows NT/2000 itself. A Windows NT/2000 server will remove any entries from its list of available printers that do not have a valid printer driver assigned to it. This means that a Windows NT client can never connect to a printer on another Windows NT/2000 host that does not have a valid driver assigned to it.

However, Samba is more flexible than NT in this case and will happily make printers available without a driver. Apparently, this is a scenario that never got tested in Windows NT by Microsoft Engineers. The result is that strange things begin to occur when a client expects the server to have a valid print driver, and the server responds with a shrug. There are `smb.conf` parameters available to help alleviate some of the peculiarities, but in general, things work much more smoothly if the server provides drivers to the client as requested.

Configuring Our First Printer

In order to begin our discussion of printing parameters in Samba, we will first need an initial `smb.conf` file that covers the basic settings. Listing 9.1 defines a bare-bones server to which we can add new printer definitions.

LISTING 9.1 Initial `smb.conf` for Our Samba Print Server

```
##
##       Mon Aug 27 01:09:17 CDT 2001
##       jerry carter <jerry@samba.org>
##
##       Sams Teach Yourself Samba in 24 Hour
##       smb.conf for print server

[global]
        netbios name = POGO
        workgroup = STY-SAMBA
        security = user
        encrypt passwords = yes
```

We are now ready to configure our first print service. Because a print share is basically a file share with a few extra attributes, we already know the majority of the parameters necessary to create a new print service in `smb.conf`. In fact, the only extra parameter required for a print share is the `printable` (or `print ok`) parameter that is disabled by default. The print service definition shown here is functional, but not yet complete.

```
[my-printer]
    comment = My first printer
    path = /var/spool/samba
    printable = yes
```

The directory defined by the `path` parameter is used to store the spooled files written by the client so that they can then be sent to the server's printing system. Typically, this directory is world writable and would have the sticky (`t`) bit set similar to the permissions on `/tmp` in order to prevent people from deleting files they do not own.



The `min print space` parameter specifies how much free room must be available in the directory in order for `smbd` to accept a print job to that printer. The default is to accept jobs as long as there is space. A value of `min print space = 5000` would inform `smbd` not to accept print jobs if there is less than 5Mb of space available in the spool directory.

After restarting Samba, we should be able to see a window similar to the one shown in Figure 9.2. Note that the "Printers…" folder will appear only if you browse the Samba server from a Windows NT/2000/XP client.

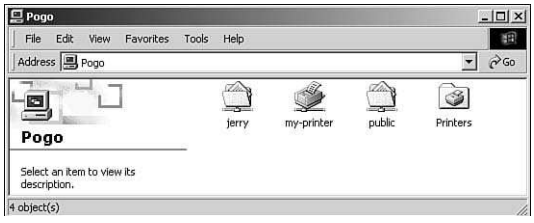


FIGURE 9.2 *View the list of shares including printers on* `\\POGO` *from a Windows 2000 client.*

At this point, we can see the printer but we have not attempted to print anything to it. If we did, how would Samba know what to do with it? In order for `smbd` to know how to do such operations like printing files, removing jobs from print queues, and pausing printers, we need to define the appropriate commands to be run in each case. Table 9.1 displays all of the available printing hook parameters.

TABLE 9.1 Print Hook Parameters Used by `smbd`

Parameter	*Description*
`lpq command`	Command to list the content of a queue
`lprm command`	Command to remove a job from a queue
`lppause command`	Command to pause the printing of a specific job in the queue
`lpresume command`	Command to resume printing a specific job in the queue
`print command`	Command to print a spooled job
`queuepause command`	Command to pause a printer's queue
`queueresume command`	Command to resume a printer's queue

The `smb.conf`(5) man page has examples for each of these. The most commonly used parameters are

- `print command`
- `lpq command`
- `lprm command`

Assuming that our Samba server is running on a host that uses the Berkeley LPD printing system, we could add the following definitions to our printer share in order to make it more functional.

```
[my-printer]
    comment = My first printer
    path = /var/spool/samba
    ; add hooks for printing files, viewing queue
    ; status, etc...
    print command = /usr/bin/lpr -Pqueue1 %s; /bin/rm -f %s
    lpq command = /usr/bin/lpq -Pqueue1
    lprm command = /usr/bin/lprm -Pqueue1 %j
    printable = yes
```

In this example, the printer that we are actually sharing is referred to by the name `queue1`, as denoted by the `-P` option. This is the link between what the Windows client sees and the final destination for printed files. There are also two new `smb.conf` variables that we have not encountered before—`%s` and `%j`. These stand for, respectively, the absolute pathname of the spooled file sent by the client and the job id number of a particular entry in the print queue.



For more information on configuring printers for your host, refer to the system documentation as to what printing system is installed.

The meaning of each command in the various printing hooks in our printer service should be obvious, with the exception of the `print command`. Put simply, this command prints the spooled file and then removes it using the rm command. Why is the call to `rm` necessary? Our use of `lpr` simply prints the file and leaves the original spooled file in place. Under normal circumstances, this is what is desired. However, `smbd` will also leave the file in the directory defined by the `path` parameter. Over time, if these spooled files are not removed, they can fill up all the available space on that disk partition. Therefore, the `rm` is necessary to throw away the file once we have printed it, in order to prevent this from happening.

Table 9.2 displays all the new variables available to printer shares.

TABLE 9.2 Printer Variable Substitutions

Parameter	*Description*
`%p`	Replace with printer name
`%j`	Replace with job number
`%s`	Replace with spool file and full pathname
`%f`	Replace with spool filename (no path)

The %p variable is expanded to the value of the printer name parameter, which is set on a per share basis. We could rewrite our printer service as

```
[my-printer]
    comment = My first printer
    ; define the destination printer's name
    printer name = queue1
    path = /var/spool/samba
    print command = /usr/bin/lpr -P%p %s; /bin/rm -f %s
    lpq command = /usr/bin/lpq -P%p
    lprm command = /usr/bin/lprm -P%p %j
    printable = yes
```


The default value of the printer name parameter is the name of the current service, so the printer share could even be further simplified to

```
[queue1]
    comment = My first printer
    path = /var/spool/samba
    print command = /usr/bin/lpr -P%p %s; /bin/rm -f %s
    lpq command = /usr/bin/lpq -P%p
    lprm command = /usr/bin/lprm -P%p %j
    printable = yes
```

Of course, this does change the name of the service, but it also makes it more obvious what printer is being shared.

Samba's autoconf script (that is, configure) attempts to determine what printing system is installed on the host by checking for various libraries and identifying the server's operating system. This is for the purpose of defining the default value for the printing service level parameter. There are eleven current possible values, shown in Table 9.3, for this parameter.

TABLE 9.3 Values for Samba's printing Parameter

Value	*Description*
BSD	The Berkeley printer daemon
SYSV	System V (for example, Solaris)
CUPS	The Common Unix Printing System from http://www.cups.org
LPRNG	The LPR Next Generation print spooler from http://www.astart.com
HPUX	The HP-UX standard printing system
AIX	The AIX standard printing system
PLP	The Portable Line Printer package from Patrick Powell
QNX	The standard QNX printing system
NT	The remote LPD queue resides on a Windows NT/2000 server
OS2	The remote LPD queue resides on an OS/2 print server
SOFTQ	The SOFTQ printing system

The `printing` parameter controls how `smbd` parses the output of the `lpq command` so that it can correctly list the current entries in the print queue on behalf of Windows clients.

Our final printer service looks like

```
[queue1]
    printing = bsd
    comment = My first printer
    path = /var/spool/samba
    print command = /usr/bin/lpr -P%p %s; /bin/rm -f %s
    lpq command = /usr/bin/lpq -P%p
    lprm command = /usr/bin/lprm -P%p %j
    printable = yes
```

Adding Print Drivers to the Server

Adding a print driver to a Samba printer service can be done one of two ways:

- Using the Windows NT "Printers…" folder and the Add Printer Wizard (APW).
- Using `smbclient` and `rpcclient` to perform the same steps as the Windows NT/2000 APW.

During this section, we will focus primarily on the first method. This means, of course, that it is necessary to have at least one Windows NT or 2000 client on the network for administrative purposes. If you are interested in how to use `smbclient` and `rpcclient` to add printers and drivers to your server, see the Imprints installation client at `http://imprints.sourceforge.net`.

The best place to begin our overview of installing print drivers on a server is the "Printers…" folder displayed when browsing a Samba or Windows NT/2000 server from an NT/2000 client (see Figure 9.2). Windows 9x/ME clients are not able to see this special folder because its existence is based solely on support for the MS-RPC printing functions. These clients lack this functionality.

Figure 9.3 shows the contents of the "Printers…" folder on our server. By definition, the only printers that Samba is aware of are those that are shared. Therefore, the list of printers in this folder must be the same as the list of printers in the enumerated shares one level up. However, you may or may not see the APW icon, depending upon the level of privilege of user you are connected to the server as. More regarding this topic will be covered in a few moments.

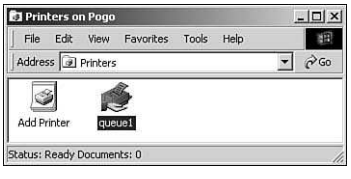


FIGURE 9.3
The Printers... folder on `\\POGO`.

In order to store the printer driver files on the server, we must define a special file share named `[print$]`. This service mimics the `%SYSTEMROOT%\System32\Printers\drivers` directory on Windows NT/2000 print servers and must contain certain directories determined by the printing clients it will support. Table 9.4 lists the directory names for various Windows clients.

TABLE 9.4 Printer Driver Directory Names

Directory Name	*Client OS*
`WIN40`	Windows 95/98/ME
`W32X86`	Windows NT/2000 (intel)
`W32ALPHA`	Windows NT (alpha)
`W32MIPS`	Windows NT (mips)
`W32PPC`	Windows NT (power pc)

First, we will define a basic file share in `smb.conf` that looks like

```
# special share for storing printer drivers
[print$]
     path = /usr/local/samba/printers
     read only = yes
     write list = jerry
     admin users = jerry
```

That is enough to begin with. We can assume that the `/usr/local/samba/printers/` directory is owned by the root account and is world-readable by executing

```
root# chmod 755 /usr/local/samba/printers
```

By setting the permissions and ownership to allow only the superuser to place new files in the share, we prevent a malicious user from placing a virus or Trojan driver files there. The user `jerry` has been given write access to the share and all file operations will be done under the context of the root account. This works around the strict file permissions we have set.

Using the directory names given in Table 9.4, we have decided that our server will provide drivers to Windows 9x/ME and Windows NT/2000 x86 clients only. Therefore, we must create the `WIN40` and `W32X86` subdirectories off the root of the `[print$]` share. In order to do this, we must first connect to the share from a Windows client.

```
C:\WINNT\> net use p: \\pogo\print$ /user:jerry secret
The command completed successfully.
```



The directories can also be created from a shell prompt on the Samba server.

The `smbpasswd` entry for the user `jerry` was created in the previous hour. Next, we can create the directories by executing

```
C:\WINNT\> mkdir p:\win40
C:\WINNT\> mkdir p:\w32x86
```

At this point, our server is set up to store drivers for known printers, but we have not actually installed any drivers on the server. This is done by using either the APW or the Properties dialog for an existing printer.



The APW should be used only if you need to actually add a printer to the server from scratch. Otherwise, use the Printer Properties window for changing settings such as the associated driver.

We were authenticated by the server when we connected to the `[print$]` share. This is a nice trick when you need to ensure that you are connected as a specific user in order to manage the printer settings. Changing the driver assigned to a printer requires access to

- Copy the new driver files to the `[print$]` share.
- Update the printer information stored in the `ntprinters.tdb` file located in the `lock directory`.

We have already learned how to give a user permission to copy files to a file share. Samba enforces its own access control mechanisms for securing updates to the `ntprinters` database. A user is given permission to update a printer entry if he or she is connected as

- The root user (not just a member of the `admin users`).
- A user or member of a group listed in the `printer admin` value.

The `printer admin` parameter controls whether or not a user is allowed to open a printer with administrative access. This access is needed to update printer settings such as the defined page size, printer permissions, and other driver properties. If a user does not have administrative access to a printer, the properties page for that device will be disabled, as shown in Figure 9.4.

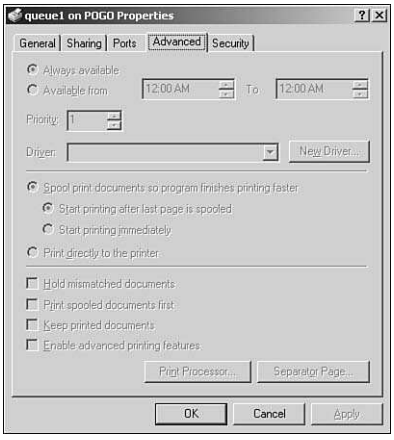


FIGURE 9.4
Viewing printer properties without administrative privileges.



If the client does not have the necessary printer drivers installed, Windows will ask the user whether or not these drivers should be installed for her. In the case of a new printer share that does not have a driver assigned to it, this step will always fail to install a driver (because one does not exist). You can safely select "no" when prompted to install the non-existent driver.

The `printer admin` list is designed to be set on a per printer basis. However, it also has the side effect of determining whether or not the APW icon is displayed as part of the "Printers…" folder. When this folder is first opened, the APW icon is shown only if the connecting user maps to the root account or is a member of the default `printer admin` users. We can set the default `printer admin` list for our server in the same fashion that we set other service level defaults—by setting the parameter in the `[global]` section of our `smb.conf`.

```
[global]
        netbios name = POGO
        workgroup = STY-SAMBA
        security = user
        encrypt passwords = yes
        ; set the default printer admin users for all printers
        printer admin = jerry
```

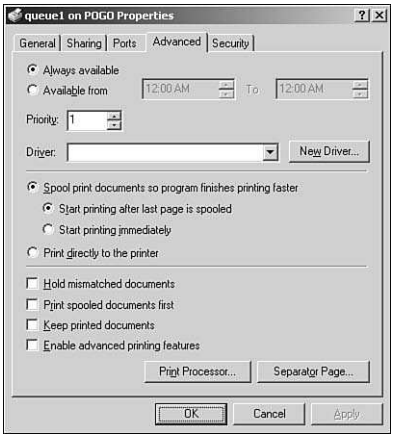
Once we have made this change, we must instruct the `smbd` process associated with our connection to reread its configuration file (that is, kill `-HUP <pid` of our `smbd>`). We can determine which process corresponds to our connection by using the `smbstatus` command.

```
root# smbstatus -u jerry
Samba version 2.2.2
Service  uid    gid    pid  machine
----------------------------------------------
print$   jerry  users  2023 caesar (192.168.215.129) Tue Sep 11 08:26:00 2001

root# kill -HUP 2023
```

Now when we open the properties window for the printer named `queue1`, all of the configuration fields should be enabled, as shown in Figure 9.5. Notice that the driver name is currently empty. This is the default state for a new Samba printer.

FIGURE 9.5
Getting ready to define a new driver for a Samba printer.



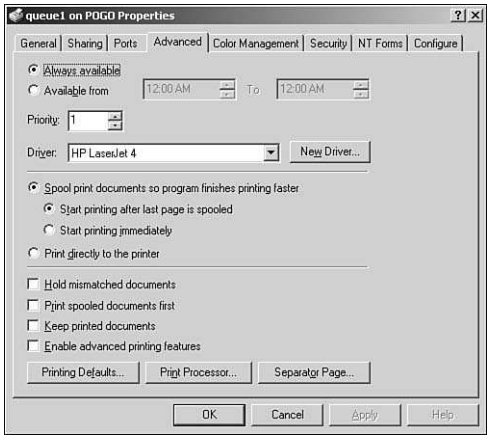
Selecting the "New Driver…" button retrieves a list of printer drivers that are included on the client's installation CD. The client first verifies actually wanting to change the existing driver. Given that there is no driver currently, the answer to this question should be obvious. If the driver for the printer is not shipped with the operating system, as is the case with our HP Laserjet 4 driver, a custom or updated driver can be installed by telling Windows that the necessary files are located on another disk somewhere. For this printer, we will use the "HP LaserJet 4" driver (See Figure 9.6), available from Hewlett-Packard's Web site (`http://www.hp.com`).

FIGURE 9.6
Selecting a model from the list of available printer drivers.



At this point, adding a new printer driver is just like changing the driver for a local printer. After letting Windows install the driver for us and after clicking the apply button once, the driver is uploaded. The new properties page is shown in Figure 9.7.

FIGURE 9.7
The printer `queue1` *with the installed HP LaserJet 4 printer driver.*



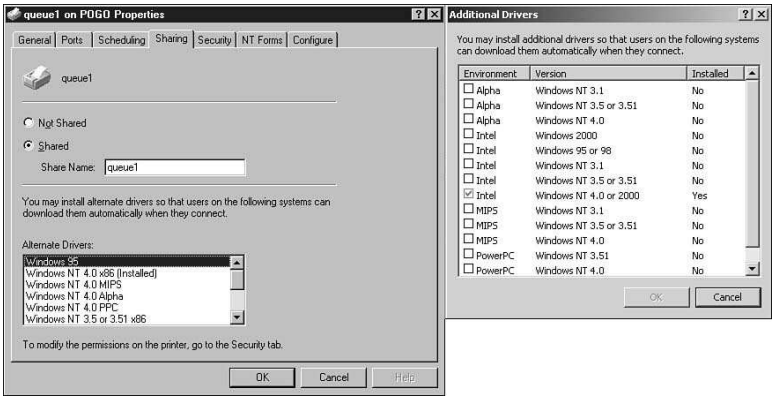
Now we are finally ready to connect a client to the printer. The context menu is displayed by clicking with the right mouse button on the `queue1` icon (located in the "Printers…" folder on `POGO`), which provides an "Install…" option. Selecting this item will create a connection to the printer in the client's local Printers folder (see Figure 9.8). These printer connections are specific to the user logged on to the Windows NT/2000 client. In other words, if a different user logs on to the NT/2000 client, this printer will not exist in the local printers folder. The user will need to establish her own connection.

FIGURE 9.8
A connection to the printer `\\POGO\queue1` *in the local Printers folder.*



Windows 9x/ME clients are also able to download drivers from the Samba host, assuming that the necessary files have been installed on the server. Samba allows drivers for multiple client architecture to be installed in the `[print$]` share in the same manner as Windows NT or 2000. From a Windows NT client, alternate drivers are installed from the Sharing tab. From a Windows 2000 client, this same operation is available under the Sharing tab of the Printer Properties window by selecting the "Additional Drivers…" button. These dialogs are displayed in Figure 9.9.

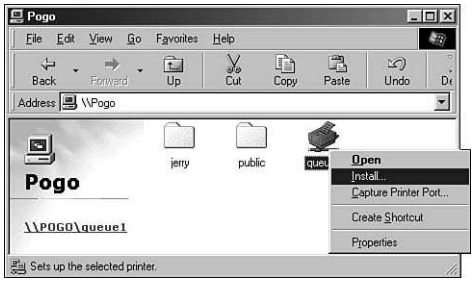
FIGURE 9.9
Installing drivers for client operating systems in addition to Windows NT x86 from a (a) Windows NT client and (b) a Windows 2000 client.



In order to install these alternate drivers, the necessary files from the client's OS (for example, the Windows 98 print drivers and INF files) must also be in hand. However, these cannot be copied directly from the client's installation CD. In this case, it is often much easier to download the latest set of drivers from the manufacturer than to locate all the necessary files from Windows itself.

Once the appropriate drivers have been installed, Windows 9x/ME clients can connect to the printer by selecting "Install…" from the context menu on the printer icon as well. Remember that Windows 9x/ME clients will not be able to see the "Printers…" folder because they do not use MS-RPC for printing. We must install the driver from the `queue1` icon in the list of available shares, as displayed in Figure 9.10.

FIGURE 9.10
Connecting to a Samba Printer from a Windows 98 client.





Samba 2.0 included several parameters and tools for installing Windows 9x drivers on the server. These included

- The `make_printerdef` tool
- The printer driver, printer driver location, and printer driver file parameters

All of these items have been deprecated in the 2.2 release and will be removed completely at a later time.

Samba also supports providing printer drivers to OS/2 clients. The details of this are outlined in the Samba-HOWTO-Collection PDF (or HTML) file included with the Samba distribution.

Using the Add Printer Wizard to Create and Delete Printers on the Server

Thus far we have been installing only drivers for existing printers already defined in the `smb.conf` file. Samba does possess the capability to allow printers to be added and deleted solely from a Windows client. This is a complicated process because it tends to be different for every printing system. For this reason, Samba does not currently include any `addprinter command` example scripts with the distribution. As most people do not require this feature and assign drivers only to existing printers, this hour will cover only the basics of how the necessary parameters function.

To add new printers from a Windows NT/2000, it is necessary to define a working `addprinter command` to the `[global]` section of `smb.conf`. This parameter defines an external program to be called. It should do two things:

- Create the new printer for the underlying printing system. Adding the new printer entry to `/etc/printcap`, for example.
- Add the new share definition to `smb.conf` if necessary. This is not necessary if `smb.conf` has the `[printers]` service defined to automatically share the new printer.

When the Windows client attempts to add a new printer, `smbd` will invoke the `addprinter command` as the current user with the following parameters (in this order):

- Printer name
- Share name
- Port name
- Driver name
- Location
- Windows 9x/ME driver location

If, after executing the `addprinter command`, `smbd` is still unable to locate the new printer share in `smb.conf`, the client will receive an error message that access was denied.

If a `printer admin` (or root) attempts to remove an existing printer from the "Printers…" folder, `smbd` will execute the `deleteprinter command` on his or her behalf. This hook is invoked with only the share name as its parameter. The only required function of the `deleteprinter command` is to remove the printer from `smb.conf`. Of course, the script can do as much or as little in addition to this as you like. If the user does not have the necessary rights to remove the printer, `smbd` will respond to the client with an "access denied" error message.

Both the `addprinter command` and `deleteprinter command` parameters were designed with the idea that some people would want to use a Samba server as a black box and manage it using the native Windows NT/2000 tools just like a normal Windows server. Neither parameter has a default value, and both are ignored unless you specifically configure them.

Using Local Print Drivers on Clients

There are times when it is necessary to install a local print driver on a client for a network printer. At such times, the network printer is considered by the Windows NT/2000 client to be a local printer, because the driver settings for the print queue are stored locally rather than on the printer server itself. Figure 9.11 displays the dialog box that appears when the server does not have a valid printer driver installed for the queue. The client is then allowed to select which driver should be installed from a type of local media, such as a CD-Rom.

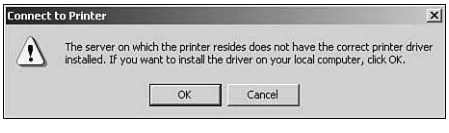
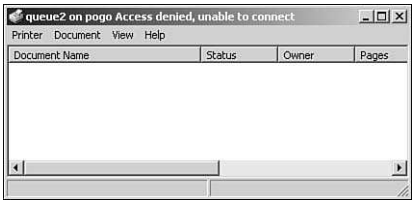


FIGURE 9.11
Installing a local printer driver on a Windows 2000 client for a network printer.

Two parameters affect how Samba serves printers and drivers to Windows NT/2000 clients. The first parameter, `disable spoolss`, is provided for backwards compatibility with Samba 2.0 only. If enabled, this global setting will remove all of the MS-RPC printing functionality in `smbd`. This means that Samba will not be able to provide printer driver downloads for any Windows NT or 2000 clients. Of course, Windows 9x/ME will still be able to obtain driver files upon demand.

The second parameter can be used on a per printer basis. The `use client driver` parameter allows `smbd` to work around a specific anomaly in Windows NT/2000 clients when using a local printer driver on a network printer. The gist of the problem is that the NT client will attempt to open the printer with the access level of the local user. Of course, a local administrator does not necessarily have administrative access to the server of a network printer. When this attempt to open the printer fails, the client displays an "Access denied, unable to connect" message in the printer queue window, as shown in Figure 9.12.

FIGURE 9.12
A failed attempt to open a printer for a network printer using a local driver on the client.



By enabling this `use client driver` parameter, you are informing `smbd` to ignore requests to open this printer with administrative access. All such requests are mapped to the normal privileges necessary to send files to the printer. If you enable this setting, you should not attempt to provide downloadable print drivers from the server for this queue. Strange and bizarre behavior will ensue.



Windows 9x/ME clients are unaffected by these parameters and can always install a local print driver if one has not been previously installed on the print server.

The `[printers]` Share

If a new printer share had to be created for every new printer added to your Samba server, administration of the Samba server would be tedious and complicated. To simplify life, Samba provides a scheme for sharing printers similar to the `[homes]` service described in Hour 8, "Samba—The File Server." When a client requests a connection to a share, Samba performs up to 5 checks when attempting to locate the service. Once the

share is located, the authorization checks needed to connect the user to the service proceed as normal. These five checks are as follows:

1. Samba searches for an explicitly defined share in the `smb.conf` file; if found, this share is returned to `smbd`.
2. If the share is not found, but a `[homes]` section is present, Samba looks for a user that matches the requested share name in the local password file. If the user is found, it copies the `[homes]` services, renames it to the requested login name, and returns that as a share to `smbd`.
3. If the share is still not found, but a `[printers]` section exists, Samba searches for a printer with the same name in the file specified by the `printcap name` parameter. If located, the `[printers]` service is copied to a new share, renamed to the located printer name, and returned to `smbd`.
4. If the share is still not found, Samba looks for a service specified as the `default service`. If a default service has been defined, it is returned to `smbd` as a match.
5. If the share is still not found, Samba returns an error to the client indicating that the network name is not present.

Step three requires a little elaboration on what `smbd` searches through to verify a printer name. The short answer is that Samba will search the file specified by the `printcap name` parameter. The default value for this parameter is determined by which printing system is selected by the `./configure` script. For LPD and LPRNG, the `/etc/printcap` file is used by default, whereas systems with CUPS or System V printing can dynamically retrieve a list of valid printers as needed. Other systems, such as AIX, search the associated printer configuration file (for example, `/etc/qconfig`).

If your system does not fall into one of these categories, it will be necessary to define a dummy `printcap` file, which Samba can use to verify printer names. The format of this file resembles the format of LPD's `/etc/printcap`, except that only queue names are required. The following list defines three printer names for use by `smbd`.

```
queue1
queue2
queue2
```

If this list were stored in a file named `/usr/local/samba/lib/smb.printer`, we would instruct `smbd` to look up printer names here by including

```
printcap name = /usr/local/samba/lib/smb.printer
```

in the `[global]` section of `smb.conf`. Of course, this defines only possible values for the `%p` variable in `smb.conf`. The server's printing system is responsible for handling the details of sending a job to a printer.

Samba includes a utility name `testprns` to allow an administrator to test various names in a given `printcap` file.

```
$ testprns queue3
Looking for printer queue3 in printcap file /usr/local/samba/lib/printers
Printer name queue3 is valid.
```

The `[printers]` share uses the same parameters available to any normal printer service. We can make all the printers in our `printcap` file available to users by changing the name of our existing printer share to `[printers]`.

```
[printers]
    printing = bsd
    comment = My first printer
    path = /var/spool/samba
    print command = /usr/bin/lpr -P%p %s; /bin/rm -f %s
    lpq command = /usr/bin/lpq -P%p
    lprm command = /usr/bin/lprm -P%p %j
    printable = yes
```

When a user attempts to connect to a service and Samba deduces that this share is a printer in the `printcap` file, `smbd` will create a copy of the `[printers]` service and rename the new share to the located printer name. For example, if a client attempted to connect to `\\POGO\queue2`, Samba would create a new share that was defined as

```
[queue2]
    printing = bsd
    comment = My first printer
    path = /var/spool/samba
    print command = /usr/bin/lpr -P%p %s; /bin/rm -f %s
    lpq command = /usr/bin/lpq -P%p
    lprm command = /usr/bin/lprm -P%p %j
    printable = yes
```

In this way, printing variables, such as `%p`, are expanded to their appropriate values.

If the `[printers]` service is defined, `smbd` will, by default, create printer shares for all known printers at startup rather than as needed. The reason for this is so that these printers will be displayed in the list of available shares when browsing the server, as shown in Figure 9.13.

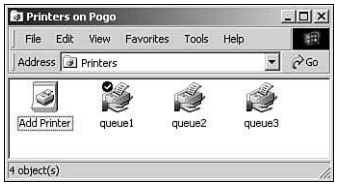


FIGURE 9.13
Including all known printers in the list of enumerated shares on a Samba host.

This behavior is controlled by the `load printers` parameter, and there are valid reasons to disable it in some environments. If the `printcap` file contains a large number of printers (~200 or so), the memory usage of each `smbd` process will increase slightly. Whether or not this is an issue for your server can easily be determined by using utilities such as `top` and `ps` to examine the system's amount of available memory.

Securing Printer Shares

Many administrators prefer to leave printers accessible to anyone who wishes to submit a job to the queue. Historically, printing systems, such as LPD, had no means of enforcing access controls on a per user basis. However, Windows NT has always provided fine-grained access control lists on printers and files alike.

Samba provides the flexibility necessary to appease both camps. Using the same guest access related parameters covered in the previous hour when configuring file shares, a Samba printer can be made available to all users on a network, regardless of whether or not the client can be authenticated. However, if the printer requires tighter controls regarding who is able to send jobs to the printer (the printer that is responsible for the company's payroll for example), Samba supplies two levels of security checks.

The first level of control is applied when the client first attempts to connect to the printer share. Samba enforces the same parameters for restricting access displayed in Table 8.6 from last hour. A list of `valid users` functions identically for printers and file shares alike.

The second level of access checks is performed when a client attempts to manipulate a print queue in some way, such as submitting a job or removing an entry from the queue. Samba 2.2 includes the capability of assigning Windows NT–style ACLs to printers and storing these settings in a local database. This is done much in the same way as file share ACLs were described in Hour 8. Figure 9.14 displays the Security dialog for a Samba printer named `hp-print`. In order to modify the ACL, you must have an entry giving him Full Control of the printer.

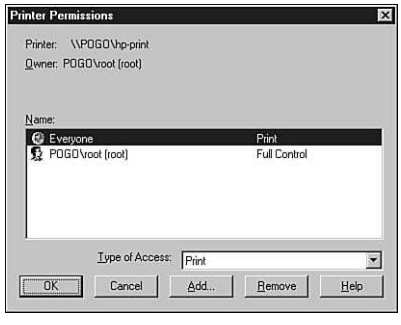


FIGURE 9.14
Viewing the ACL for `hp-print` *from a Windows NT 4.0 client.*

For more information on Windows NT ACLs on file and printers, refer to the Windows NT Resource Kit or another book covering Windows NT/2000 security.

Troubleshooting Printer Shares

We will conclude this hour with some basic steps for troubleshooting a printer service when things do not go as smoothly as planned. Probably the most frustrating problem to track down is when a Windows user prints a file and it just disappears. Nothing is ever generated by the printer itself. When this happens the first step to verify is that the user can print to the printer at all from a shell prompt on the server. If the printer in question was named `queue1` and the user `toby` is unable to print from his Windows client, first use the `su` command to log on to the server as this user.

```
root# /bin/su - toby
```

Next, attempt to send a test page to the printer using the exact same syntax as the `print command` defined for the printer share. The following example sends the `/etc/hosts` file to the printer named `queue1` using the Berkeley `lpr` command.

```
$ /usr/bin/lpr -Pqueue1 /etc/hosts
```

If this file fails to print, follow the documentation for your server's OS and printer to determine the cause. If the test page is printed successfully, move on to verifying step two.

Next, we will make sure that the print job spooled by the Windows client is in the correct format. This is done by capturing the file in the spool directory without ever actually sending it to the printer. The easiest means of accomplishing this is to temporarily change the `print command` for the share to

```
print command = echo %s >> /tmp/%S.log
```

This maintains a log file of the name of each spooled job so that the actual file can be examined using a text editor. Remember that the `print command` does not remove the spooled jobs unless instructed to. Our debug `print command` allows us to capture these jobs without making any changes on the Windows client.

If you find that Windows is never transmitting the job to the server, verify that the client has not marked the printer as offline, which causes it to appear as a grayed out icon (see Figure 9.15). If this is the case, it is a good idea to clear the local print queue on the Windows system, mark the printer as online, and reboot the client. If the client OS continues to automatically mark the printer as offline, follow the troubleshooting techniques outlined in Hour 13, "Troubleshooting Techniques," to ensure that the Windows client and Samba server can properly communicate with each other.



FIGURE 9.15
A disabled network printer on a Windows 98 client.

Assuming that the client does spool the file to the Samba server, attempt to print the file as the user again by using the same syntax as the original `print command` (without removing the file, however). If the printer share is mapping connections to a guest account (for example, `guest only = yes`), ensure that the specified `guest account` is valid and can send files to the printer. If the file fails to print, begin looking for items such as

- A mismatch between the print driver installed on the client and the model printer assigned to the queue
- Hardware limitations of the printer, such as insufficient memory, that would prevent it from handling the file

Although it is impossible to outline every possible difficulty that may arise when administering printers, the steps covered in this section will help to diagnose 80–90% of the problems that normally occur.

Summary

Samba 2.2 introduces a significant upgrade to Samba's support for sharing printers to Windows clients, Windows NT/2000 clients in particular. The basic steps for exporting a printer to users are

1. Define the printer share in `smb.conf`.
2. Install the necessary printer drivers for clients on the Samba host.
3. Install the printer on the Windows host.

By using the special `[printers]` share, step 1 can be done for all known printers on a system at once.

Samba 2.2 also introduces the capability to add and remove printers on a system by using the native Windows NT printer management tools, such as the Add Printer Wizard. In Hour 14, "Replacing a Windows NT File and Print Server—Lock, Stock & Barrel," we will discuss more `smb.conf` hooks to allow Samba to be managed using other native Windows NT tools.

Q&A

Q My users are printing to a PostScript printer, but the PostScript is being printed rather than the actual page. How do I fix this?

A Many Unix systems have a PostScript printer filter that converts text files into PostScript (for example, `enscript`). The filter is smart enough to leave PostScript alone so it prints correctly. However, some Windows printer drivers place a Ctrl-D (`^D`) at the beginning of the file. This confuses the filter, which is usually looking for `%!`, and the PostScript in the print job is converted to PostScript and you get pages and pages of rubbish. You can fix this by adding the parameter `postscript = yes` to the printer share.

Q I have a System V machine. How do I tell Samba about all the print queues on my server so my clients can browse them?

A Here you should ensure that `printcap name = lpstat` is being used by `smbd`. This is the default value for systems such as Solaris. If, however, the `printcap name` parameter has been manually set in `smb.conf` to a different value, `smbd` will be unable to automatically determine the complete list of available printers.

Q My clients are getting the error message that the "Disk is Full" when trying to print to my Samba server, but there is plenty of space. What is the problem?

A If `smbd` is unable to write the spooled file to the directory defined by the `path` parameter for a printer if the `write` permission were denied, for example it would respond to the client with the message, "Disk is Full." Samba will also return this error message if the amount of free disk space in the spool directory has fallen below the value specified by the `min print space` parameter.



PART III
The Social Life of Samba

Hour

10 Microsoft's DOS Network Client and Windows 95/98/ME

11 Windows NT 4.0/2000

12 CIFS Clients for Unix

13 Troubleshooting Techniques



HOUR 10

Microsoft's DOS Network Client and Windows 95/98/ME

Up to this point, we have focused almost exclusively on Samba and the steps necessary to configure our CIFS server. Beginning with this hour and continuing through the next hour, we will turn our attention to the basic steps of configuring Microsoft-based CIFS clients. At the end of this hour, we will have covered all the necessary details for connecting to a Samba server from both a DOS client and a Windows 9x/ME client. Our goal is not to understand how to use DOS/Windows networking features. Rather it is to ensure that we have the necessary components installed in order to connect to a Samba server.

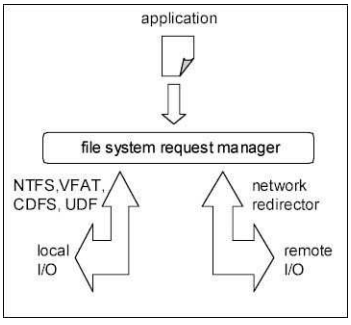
This hour assumes that you are already familiar with adding hardware and software to a PC running either DOS or Windows. If you are not comfortable with either of these topics, it is a good idea to refer to a book such as *Special Edition: Using Windows ME.*

The Network Redirector

Before looking at the details of how to configure a DOS or Windows network client, let's take a few moments to explore the basic Windows network architecture. Be warned that the Windows 9x network redirector and the one that is a part of Windows NT differ significantly, but the basic theories behind them both are similar.

Figure 10.1 shows a simplified version of how Windows handles access to remote network shares. I will assume for my description that I am referring only to disk shares. The explanation for remote printers is similar.

FIGURE 10.1
Windows network redirector architecture.



There are three basic parts to the implementation:

- The application requesting the disk I/O.
- Some type of file system request manager. Windows 95 and 98 implement this using an installable file system manager. The Workstation service handles the user-level requests in Windows NT.
- The network redirector.

The process works something like this:

1. The application requests some disk I/O.
2. The request is sent to the file system request manager.
3. The file system request manager then forwards the request to the appropriate file system device driver. If the requested I/O is local, the local file system driver (NTFS, VFAT, CDFS, UDF, and so on) will service it. If the request is for I/O on a network share, the information is packaged and sent to the requested server.
4. The requested I/O is returned to the application through the file system request manager.

This is an oversimplified view of things, but at least when the term *network redirector* comes up, you will have a general idea what I am talking about.

Microsoft Network Client Version 3.0 for MS-DOS

It might be hard to believe that with the current releases of Windows ME, Windows 2000, and Windows XP, a DOS network client is worth mentioning at all. Who in their right mind would still want to use an operating system that can only do one thing at a time and has a memory limit of 640KB? The truth is that a DOS boot floppy can prove to be quite helpful. After booting the system to a minimal state, you can then proceed to use things that Windows 9x tends not to like, such as disk editors and certain PC games.

These boot floppies also tend to be very helpful when used in conjunction with disk imaging software such as Norton Ghost. By using an SMB client on a DOS floppy, it is possible to boot onto the network, mount any necessary drives, and load the machine from an image file located on a Samba server. The entire process takes about 20 to 30 minutes to get a working Windows 98/ME machine up and running on the network. Of course, a few other small details make this possible, such as the use of DHCP and identical hardware on all lab machines. Another useful situation is when you need to install Windows on a machine that does not have a CD-ROM drive. Loading current Windows releases from floppy disks is not as reasonable as it used to be. By mounting a shared CD-ROM and loading the software across the network, installation times comparable to using a local CD-ROM drive can be achieved.

The remainder of this section explains how to create a bootable DOS floppy that allows you to mount shares from a CIFS server. We will cover three basic steps:

1. Obtaining the client software
2. Installing the network client software onto the local hard disk
3. Creating the network boot disk from the files installed in step 2

Obtaining the Software

Microsoft freely distributes its network client for DOS. If you have a copy of the Windows NT 4.0 Server CD-ROM, you can save yourself the download time by accessing the two-disk set in

```
X:\clients\msclient\disks
```

where `X:` is the drive letter of your CD-ROM drive.

If you do not have a copy of the CD-ROM, you can download the DOS client install disks from

```
ftp://ftp.microsoft.com/bussys/Clients/MSCLIENT/
```

The directory has two files:

```
DSK3-1.EXE
DSK3-2.EXE
```

Download both files into a temporary directory (for example, `c:\temp`) and extract each disk by executing

```
C:\> mkdir c:\temp\disk1
C:\> cd c:\temp\disk1
C:\> c:\temp\dsk3-1.exe
C:\> mkdir c:\temp\disk2
C:\> cd c:\temp\disk2
C:\> c:\temp\dsk3-2.exe
```

At this point, no matter what method you use to obtain the network client files (CD-ROM or FTP), you should be ready to copy the files onto two 1.44MB floppies and install the client.

Installing the Client

To create a network boot floppy, first go through the steps for installing the DOS network client onto a hard drive. Then copy only the files required onto the boot floppy. For reference sake, I will use MS-DOS 6.22 as the operating system.



Practice has shown me that it is easier to get things up and running under normal conditions before attempting to create the boot floppy.

After copying the files onto two disks, we are ready to install the software. In this example, I assume that your floppy drive letter is `A:`. Insert the first disk into the client computer and type **`C:\> a:\setup.exe`**.

What results should be the screen displayed in Figure 10.2.

Continuing to the next screen, you accept the default install location of `C:\NET`. After examining the system for a few seconds, the setup program returns with a list of possible network adapter drivers to install (see Figure 10.3).

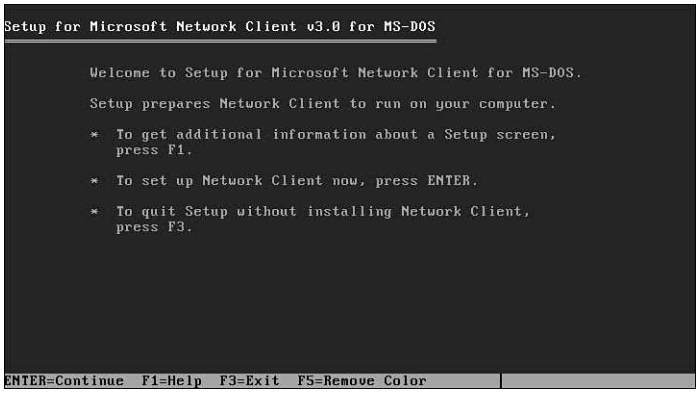


FIGURE 10.2
Installation screen for Microsoft's DOS network client 3.0.

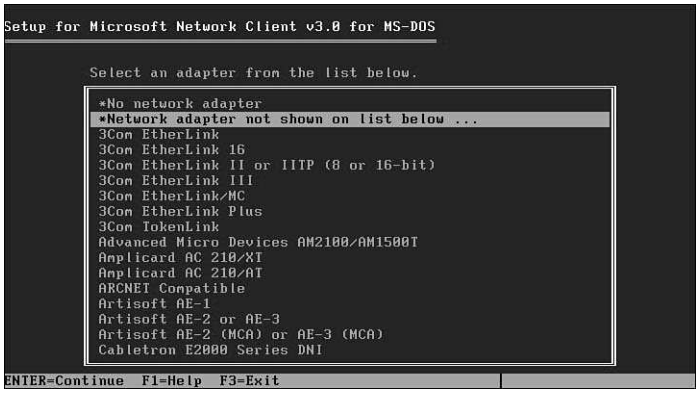


FIGURE 10.3
Selecting a network adapter driver to install.

This can be the trickiest part of the setup. If you do not see your particular network adapter listed—and chances are you won't—locate the MS LanMan for DOS drivers for your card. I normally use the standard NDIS2 drivers shipped with the network card. The network client's setup program attempts to locate a file named `OEMSETUP.INF` in the directory designated as containing the card's drivers. If the setup program does not find this file, it complains that it could not find any legitimate drivers. This can be a hassle. It can take a few tries, but you should be able to select your card from a list of drivers that were located. And as always, check the vendor's Web site for any updated drivers.

The next step is to specify a username that will be used by default. This is more for convenience because you can always specify another when actually logging on to the network.

Figure 10.4 displays the options screen. The Names setting allows for changing the NetBIOS machine name and the workgroup name used for browsing and the domain name used for validation if the Logon to Domain option is enabled. Figure 10.5 displays the defaults for these three values.

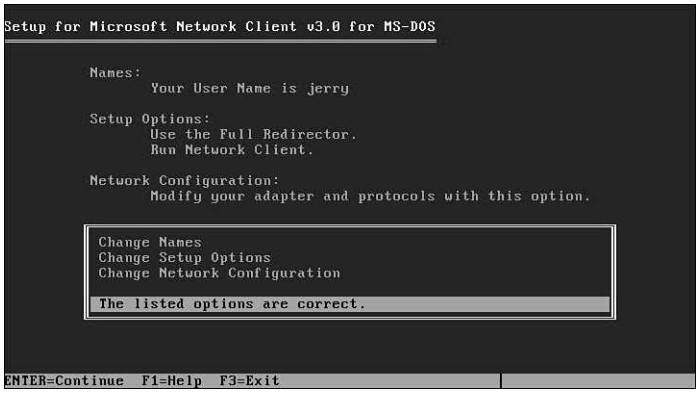


FIGURE 10.4
DOS client general configuration screen.

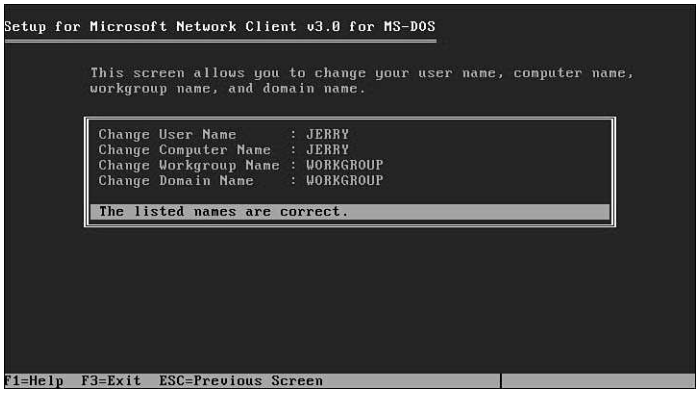


FIGURE 10.5
NetBIOS name configuration screen.

The Setup Options probably need a little clarification. The setting shown in Figure 10.6 says that you are configured to use the Full Redirector. There are two options available to you. The full network redirector allows for domain logins and uses about 100KB of memory. The basic redirector uses considerably less memory but only allows for mounting printer ports and network drives. The basic redirector also requires that you enable password encryption on your server. The full redirector supports plain-text passwords and encrypted passwords. Choose the Full Redirector for now. If you later discover that you need to reduce the amount of memory in use, you can run the setup program again to modify your settings.

FIGURE 10.6
DOS client detailed network configuration screen.

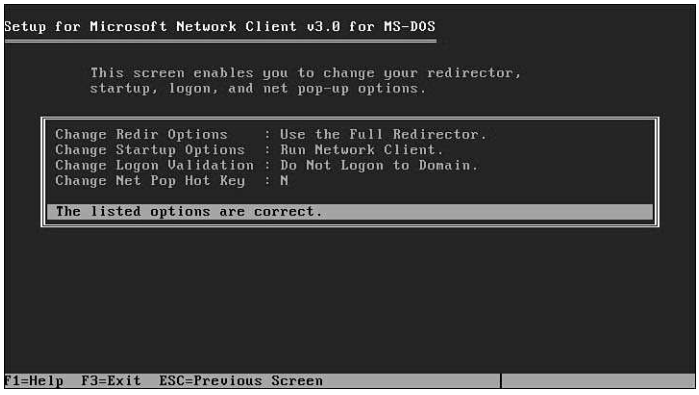
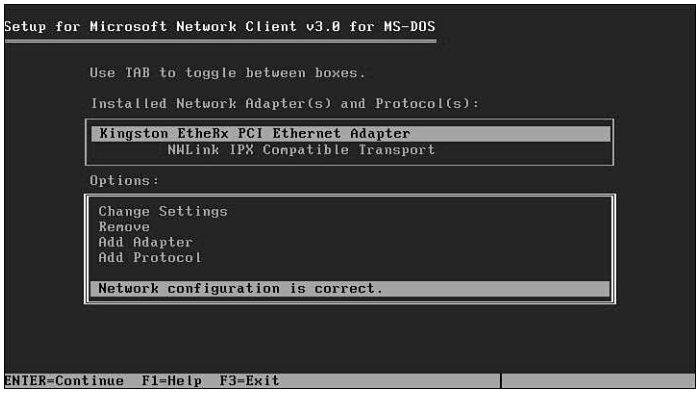


Figure 10.7 displays the installed network components. This is where you choose what network protocols should be installed. In our case, we must add TCP/IP and remove other default network protocols such as NWLink. The network adapter has already been selected and therefore should not need to be changed.

FIGURE 10.7
Installed network adapter and default installed network protocols.



First, add TCP/IP. After adding the TCP/IP protocol, use the Tab key to select the NWLink entry and delete it by using the Remove option listed on the screen. Before returning to the previous screen, shown in Figure 10.4, manually configure settings for your IP address and subnet mask. Figure 10.8 shows the default values for the network addresses. The only ones you should concern yourself with now are the Subnet Mask, the IP Address, and Disable DHCP (set to the value `1`). If you need to access hosts across a router, you can also specify a default gateway to use.

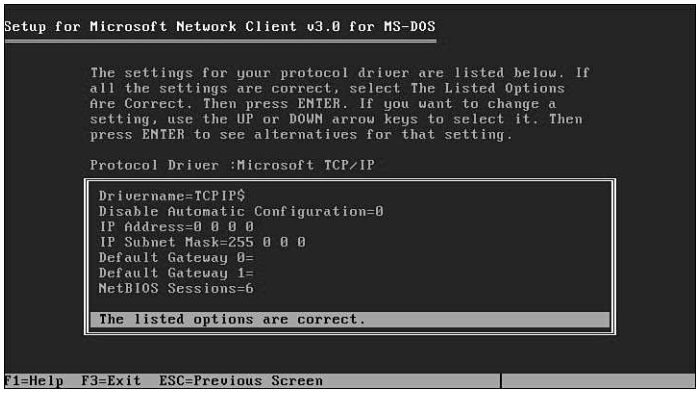


FIGURE 10.8
TCP/IP networking settings.

Now the network configuration is correct, and so you return to the screen shown in Figure 10.4 again. After copying the necessary files, setup informs you that everything completed successfully and that you should reboot for the new configuration to take effect.

When the machine comes back up, you will be asked to log in to the network. You'll receive a prompt similar to mine:

```
Type your user name, or press ENTER if it is JERRY:
```

After pressing the Enter key, you are prompted for your password:

```
Type your password:*******
```

Following this, the DOS network client looks for a password cache file to verify the entered password against. Because this is your first time logging in, of course it does not find one. Therefore, it asks whether you want to create one. Given my natural animosity towards caching passwords on the disk of an insecure computer, I decline:

```
There is no password-list for JERRY.
Do you want to create one? (Y/N) [N]:
```

After being informed that `The command completed successfully`, you can use the basic set of options with the `net.exe` command to access drives and browse the network. For example, you can use the `net use` command to mount a shared Zip drive from a Windows NT 4.0 server or a Samba server for that matter:

```
C:\> net use z: \\pogo\zipdisk
The command completed successfully.
```

All connection attempts to servers will use the login name entered when the client was started on the PC. There is no way to work around this.

Making the Network Boot Disk

Now that we have a working setup on the local disk, we can begin to build the network boot floppy. First we must create a simple boot disk by executing

```
C:\> format a: /s
```

After the system files have been transferred to the disk, make two directories, one named `DOS` and one named `NET`. The first directory will contain any required DOS utilities, such as memory managers, and the second will contain the necessary files for the DOS network client.

If you examine the `\NET` directory located on the hard disk, you will find approximately 1.6MB of files. Finding the right files to copy to the floppy disk can be a process of trial and error. For that reason, the files you use on the boot floppy are given in Listing 10.1. The disk's `config.sys` file in Listing 10.2 and its `autoexec.bat` in Listing 10.3 follow the directory listing.

LISTING 10.1 Files Located on the Network Boot Disk

```
 Volume in drive A has no label
 Volume Serial Number is 2629-09D8

Directory of A:\

dos          <DIR>         02-01-99   2:20a
net          <DIR>         02-01-99   2:20a
autoexec bat           245 02-01-99   2:40a
command  com        54,645 05-31-94   6:22a
config   sys           132 02-01-99   2:39a
        5 file(s)         55,022 bytes

Directory of A:\DOS

.            <DIR>         02-01-99   2:20a
..           <DIR>         02-01-99   2:20a
format   com        22,974 05-31-94   6:22a
sys      com         9,432 05-31-94   6:22a
emm386   exe       120,926 05-31-94   6:22a
fdisk    exe        29,336 05-31-94   6:22a
vi       exe        46,130 01-24-96   2:23p
himem    sys        29,136 02-13-94   6:21a
        8 file(s)        257,934 bytes



Directory of A:\NET

.            <DIR>         02-01-99   2:20a
```

continues

LISTING 10.1 Continued

```
..           <DIR>          02-01-99   2:20a
hosts                 715 08-31-94   7:37p
lmhosts               817 08-31-94   7:36p
networks              395 08-31-94   6:52p
protocol              795 08-31-94   6:52p
services            5,973 05-08-95   2:34p
wfwsys   cfg          840 02-01-99  12:54a
netbind  com        8,513 08-31-94  12:00a
umb      com        3,325 08-31-94  12:00a
connect  dat           40 02-01-99   2:44a
ktc40    dos       49,057 07-21-95   7:31p
protman  dos       21,940 08-31-94  12:00a
nemm     dos        2,619 08-31-94  12:00a
tcpdrv   dos        4,174 08-31-94  12:00a
protman  exe       13,782 08-31-94  12:00a
emsbfr   exe        4,294 08-31-94  12:00a
nmtsr    exe       22,826 08-31-94  12:00a
ping     exe       66,460 08-31-94  12:00a
tcptsr   exe       71,040 08-31-94  12:00a
tinyrfc  exe       37,024 12-01-94   7:39p
net      exe      450,326 02-07-95  12:40p
system   ini          497 02-01-99   2:42a
protocol ini          356 02-01-99   1:51a
tcputils ini          233 08-31-94  12:00a
net      msg       76,234 03-03-95   7:11p
neth     msg      123,066 03-03-95   7:12p
shares   pwl          622 02-01-99  12:54a
ifshlp   sys        4,644 08-31-94  12:00a
       29 file(s)        970,607 bytes

Total files listed:
       42 file(s)      1,283,563 bytes
                          18,432 bytes free
```

LISTING 10.2 `CONFIG.SYS` File from Network Boot Floppy

```
device=a:\dos\himem.sys
device=a:\dos\emm386.exe noems
dos=high,umb
files=99
buffers=45
lastdrive=z
device=a:\NET\ifshlp.sys
```

LISTING 10.3 AUTOEXEC.BAT File from Network Boot Floppy

```
@echo off
set prompt=$p$g
set dircmd=/l/oe/p
set copycmd=/v
set PATH=a:\NET;a:\dos

a:\NET\net initialize
a:\NET\netbind.com
a:\NET\umb.com
a:\NET\tcptsr.exe
a:\NET\tinyrfc.exe
a:\NET\nmtsr.exe
a:\NET\emsbfr.exe
a:\NET\net start
```


As I have mentioned before, one of my previous responsibilities was to manage several public computer labs. Each lab was set up to contain machines with homogenous hardware. Therefore I could clone the disks and simply place them in another machine without too much disruption. Many software packages are available for cloning hard disks. You can even use the `xcopy` command to some extent. The particular package I use allows an entire hard disk to be dumped to a single file. These files can be fairly large and can change often due to updates and fixes in the lab.

Originally, I used a writable CD-ROM drive to place the image file on a CD, but this proved to be bothersome. After stumbling across the necessary software I can simply pull an image from the server across the network to each machine in the lab at the same time! Given that a standard configured machine loads the image in about 30 minutes and one lab has close to 50 machines, my rollout time was cut substantially as you can see.

Windows 9x/ME

Windows ME is the latest evolution in the line of Windows operating systems designed for personal use. However, although it is the latest operating system designed for personal use, my experience has been that Windows 98 is more stable for use on a network. Your mileage may vary, so take that opinion with a grain of salt. The screenshots for examples in this section use a Windows 98 client.

Configuring the Client

There are three components to configuring the Windows 98/ME client:

- The network interface adapter driver
- The TCP/IP protocol stack
- The SMB client (also known as Client for Microsoft Networks)

The main dialog of the Windows 98 network control panel is shown in Figure 10.9.

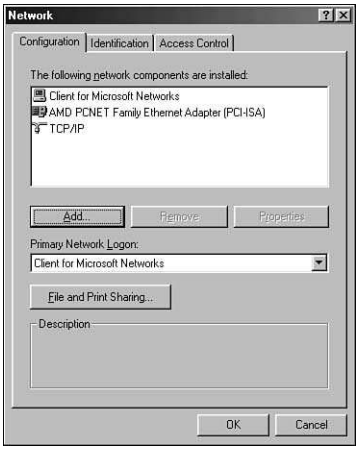


FIGURE 10.9
The network control panel for a Windows 98 client.

The Network Interface Adapter

The network adapter driver is usually the first item that you will install. It is beyond the scope of this book to provide detailed instructions for every network adapter card you could possibly install, but Windows 9x/ME automatically detects and configures most modern network cards. If you are using older 16-bit ISA cards, you will probably have to configure the card manually and install the software drivers yourself. Network cards are usually distributed with a floppy disk that contains instructions for installation and the latest drivers. However, checking the manufacture's Web site to verify that you have the most recent version of the driver's disk is always a good idea.

As part of the initial installation of a network card (also known as NIC), Windows 98 automatically adds two networking components in addition to the NIC device driver. This occurs regardless of whether Windows locates and installs the network adapter drivers automatically. The two additional components are

- Client for Microsoft Networks
- TCP/IP Protocol

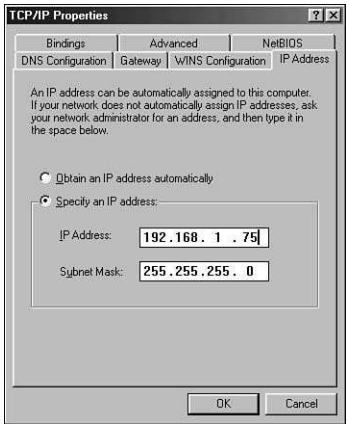


Windows 95 adds additional network protocols as well as a client for NetWare networks. These should be removed.

The TCP/IP Network Protocol Stack

Several configuration options are available for Microsoft's TCP/IP protocol stack on the PC by selecting the TCP/IP protocol from the list and clicking the Properties button. The first settings page appearing in the resulting window enables you to set the local IP address and subnet mask. There are two possibilities. The first enables you to obtain an IP address from a DHCP (Dynamic Host Configuration Protocol) server, and the second lets you manually configure the two settings. For this example, you will manually configure the IP address and subnet mask (see Figure 10.10).

FIGURE 10.10
Configuring the IP address and subnet mask.

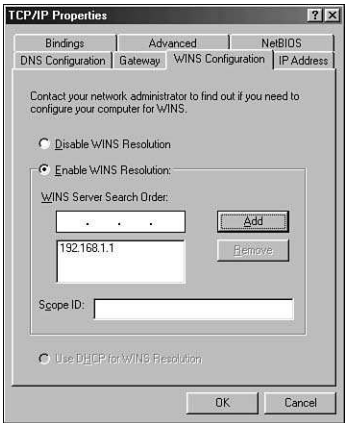




For more information on networking TCP/IP computers, refer to Sams' *Teach Yourself TCP/IP in 24 Hours* or *TCP/IP Primer Plus*.

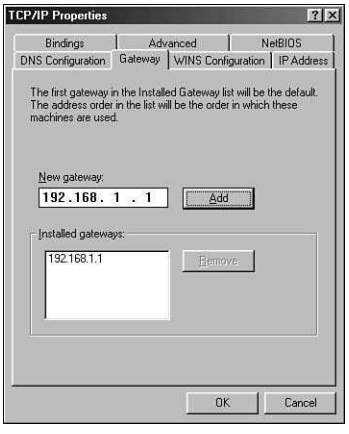
Figure 10.11 displays the next TCP/IP settings tab. This page allows you to enable WINS resolution and to specify the IP addresses of your WINS servers. If WINS resolution is enabled, you can also define a Scope ID string. The third possibility is to obtain the IP addresses for the WINS servers from the DHCP server.

FIGURE 10.11
Windows 98 WINS configuration tab.



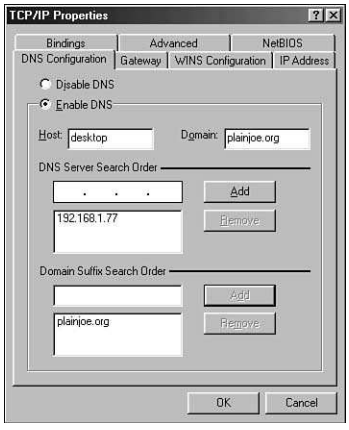
If your subnet has a default gateway for traffic destined to another IP network, you can specify the IP of the router in the Gateway tab (see Figure 10.12). For a small network not connected to the Internet this field is frequently empty.

FIGURE 10.12
Configuring a default router in Windows 98.



The final TCP/IP settings that I will discuss are the DNS settings for the PC (see Figure 10.13). You have the choice of either enabling or disabling DNS. If you enable DNS, you can define the IP addresses of your DNS servers, the domain suffix search order, and the PC's hostname and domain. As a side note, it is possible to set the DNS server's IP addresses from a DHCP server if you so choose.

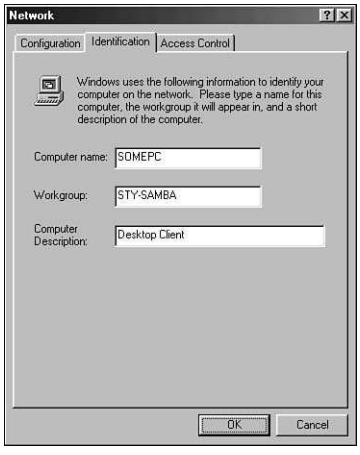
FIGURE 10.13
Configuring DNS settings.



Setting the Machine Name and Workgroup Name

The final step for configuring our Windows 98 PC to access shares located on an SMB server is to set the NetBIOS machine name and workgroup name. You can refer to Hour 2, "Windows Networking Concepts," for a reminder of what constitutes a valid NetBIOS name. In order to define these two strings in the identification page (see Figure 10.14), the Client for Microsoft Networks must be installed. The `Comment` field enables you to set the text string that is displayed next to the machine name if you have enabled file and print sharing on the PC. This setting is the equivalent of Samba's `server string` parameter in `smb.conf`. After you have set the desired values, you can click the OK button. Windows begins copying the necessary files from the installed `*.cab` files or from the Windows CD-ROM. When the operating system is finished copying files, it prompts you to reboot your computer. When you have done so, you can move to the next step of logging in to the network.

FIGURE 10.14
The identification tab for defining NetBIOS names.



Logging In to the Network

When the PC has rebooted, you will be prompted to log in to the network (see Figure 10.15). There are a few points to remember about logging in to the network:

- The username and password you specify in the login box are the defaults used by Windows to connect to remote servers. Windows 9x/ME does not enable you to specify a different username to use in connections to different servers. However, if the username and password that you entered in the login box fail to connect to a server, Windows uses the same username but prompts you for a new password.
- The username and password you enter are not validated at login, but rather when you connect to a server. This means you will have no feedback if you type in the wrong password until the client attempts to access a share on the CIFS server.
- You must log in to access nonguest shares on a CIFS server.

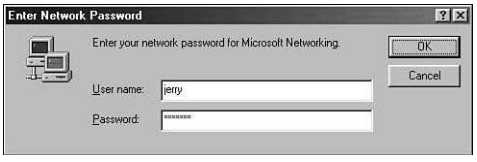


FIGURE 10.15
The logon dialog window.

Encrypted and Plain-Text Passwords

The original distributions of Windows 95 would gladly use a plain-text password for SMB connections if the server asked for it. When an SMB server responds to a negotiate protocol request, the response packet contains a bit to indicate whether the server supports the challenge/response authentication model described in Hour 7, "Security Levels and Passwords."

With the release of the network redirector update for Windows 95 (`vrdrupd.exe`), Microsoft changed the default so that Windows 95 clients would not send the plain-text password to a server that did not support encryption. This was a good thing because it meant that a rogue server could not capture a user's clear-text password simply by telling it, "I don't support encrypted passwords. Please send me your username and password in clear text."

What this means to you as a Samba administrator is that if you attempt to use any of the following clients with a Samba server that does not have `encrypt passwords = yes`, the user will either continually be prompted for a password that will never be accepted or will receive an error message from Windows NT that says, "This account is not authorized to log in from this station."

- Windows 95 with the network redirector update
- Windows 98 or Windows 98 Second Edition
- Windows ME
- Windows NT 4.0 Service Pack 3 or later
- Windows 2000 or Windows XP

It is possible to determine if a Windows 95 client has the network redirector installed by examining the `vredir.vxd` and `vnetsup.vxd` driver files. If these files are the following versions or later, the update has been installed.

```
windows\system\vredir.vxd     (4.00.1114 6/2/97 11:14a 156,773)
windows\system\vnetsup.vxd    (4.00.1112 6/2/97 11:12a  17,595)
```

There are two possible solutions:

- Set up the Samba server to use encrypted passwords using the steps outlined in Hour 7.
- Enable the Windows client to use plain-text passwords.

If you choose the second solution, there are several registry settings files distributed as part of the Samba source release that will enable the client to downgrade to clear-text passwords. Search for a file named `XXX_PlainPassword.reg` where the `XXX` matches your operating system (for example, Win95, Win98, and so on . . .) and merge this file into the registry on the local Windows client. The client (including Windows 2000 and XP) must be rebooted in order for the change to be applied.



A Windows client with this registry setting applied will still use NTLM for authentication if the server supports it. In other words, this change does not disable the client from using password encryption. It only allows the client to downgrade to clear-text passwords where necessary.

Summary

This hour has presented two very different SMB clients. Microsoft's DOS network client is distributed free from various locations such as the Windows NT server CD-Rom and Microsoft FTP site. Windows 9x/ME clients ship a fully functional SMB client as part of the operating system.

Both DOS and Windows hosts require three components to connect to a CIFS server such as Samba.

- A working network interface card.
- A correctly configured version of the TCP/IP network protocol stack.
- A network redirector which supports the CIFS protocol.

Q&A

Q When I try to mount a drive from a Samba server using the MS-DOS network client with the basic redirector, I keep getting an error message saying that the password is invalid. Why is this?

A The DOS client basic redirector transmits only the LanManager 24-byte hash. If your Samba server does not have password encryption enabled, you need to either use the DOS full redirector or enable encrypted passwords on your server.

Q When I click on my Samba server in the network neighborhood, I'm continually prompted for a password to the IPC$ share no matter what I enter.

A The Windows client is attempting to use encrypted passwords. However, the Samba server is configured to support only clear-text passwords. You should either enable encrypted passwords on the server or enable clear-text passwords on the Windows client.

Q Can the MS-DOS Network Client also connect to shares provided by Windows NT, Windows 2000, and Windows 9x/ME hosts?

A Yes. This was Microsoft's original intention for the DOS network client. The steps for connecting to a Windows-based server are exactly the same as those for connecting to a Samba server.

New Terms

NDIS2 Version 2 of the Network Desktop Interface Specification. This creates a defined interface between the hardware and upper-level network protocol that vendors can use to write compliant device drivers for network components. There is also a version 3 of this specification, referred to as NDIS3.



HOUR 11

Windows NT 4.0/2000

Although it shares the same user interface as Windows ME, Windows 2000 is an entirely different operating system (OS) from the ground up. The advantage of having the same graphical user interface (GUI) is that many of the steps for using the Windows 2000 CIFS client are similar to the steps for using a Windows 98/ME client.

Accessing CIFS servers from newer Windows operating systems is fairly simple to configure, and Windows NT 4.0/2000 is not an exception. In this hour, we will explore the details of how to install, configure, and use the CIFS client included with both Windows NT 4.0 and Windows 2000.

Windows NT

Windows ME and Windows 2000 appear very similar from the user interface. Windows NT and its 95/98 counterpart, however, only use the same interface most of the time. One area where these internal differences can be seen on the outside is the native networking support. Both operating systems use a type of network redirector similar to one described in Hour 10,

"Microsoft's DOS Network Client and Windows 95/98/ME." Under Windows NT, the `Workstation` service takes the role of the Windows 9x Client for Microsoft Networks. The remaining two required components are the network adapter and the TCP/IP network protocol stack just as was the case with Windows 98.

Installing the Network Adapter

We begin with the hardware portion of Windows NT's network configuration. Unlike Windows 9x, Windows NT requires that you manually add and configure the network adapter drivers. This means you must know some details about your particular card. Walking through all the possible steps for installing every network card is beyond the scope of this book. Normally, any issues with installation are covered by documentation provided by the network card's vendor.

Before starting, make sure you have the latest drivers from the hardware manufacturer. It is often the case that vendor specific drivers perform better than the plain-vanilla Windows NT drivers included as part of the OS. For some newer cards, you may find that the only available drivers are those from the NIC's vendor.

Once the card has physically been installed in the PC and we arrive at the Windows NT Network control panel (Start, Settings, Control Panel, Network), notice that the adapters, protocols, and services are all listed on different pages.

First navigate to the Adapters tab (see Figure 11.1). Now select the Add button and locate the appropriate driver for your card. If you need to install a custom driver, pressing the Have Disk button will prompt you for the location of the files.

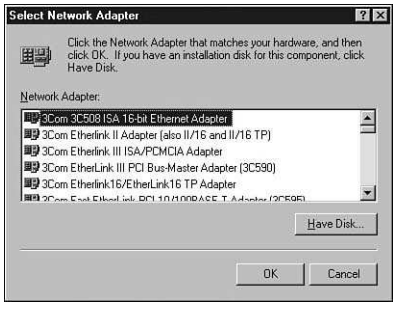


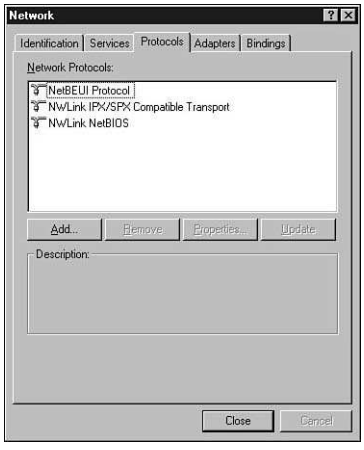
FIGURE 11.1
Windows NT 4.0 Network Adapters tab as part of the Network control panel.

After Windows NT copies the necessary driver files from the Windows NT installation CD-ROM or drivers disk, we are ready to move to the next step of installing the TCP/IP protocol stack.

Installing the TCP/IP Protocol

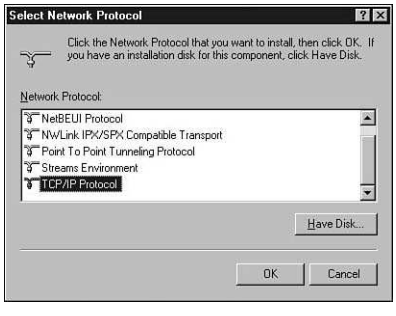
Windows NT, like Windows 95, automatically adds two protocols for you when the first network adapter is installed. Figure 11.2 shows the two NWLink entries and the one for the NetBEUI protocol. The NWLink NetBIOS entry depends on the NWLink IPX/SPX protocol and is installed with it, so I refer to both as one item.

FIGURE 11.2
Default network protocols installed with a new network adapter in Windows NT 4.0.



First we will add the TCP/IP network stack by clicking the Add button and selecting the protocol's entry from the list as shown in Figure 11.3. Then we remove the NWLink entries and the NetBEUI protocol from the installed list so only the TCP/IP network protocol remains.

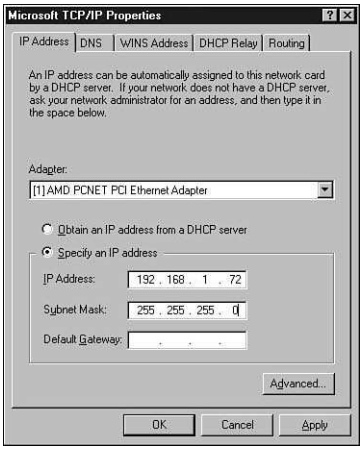
FIGURE 11.3
Installing TCP/IP on a Windows NT 4.0 client.



When we choose to confirm the network changes, Windows NT verifies the adapter and protocol bindings and prompts for any missing information as shown in Figure 11.4. The only information I have chosen to specify at the moment is the IP address of the machine

and the subnet mask. If your setup requires setting a default gateway or indicating the IP addresses of your DNS servers, you should do so now.

FIGURE 11.4
Supplying TCP/IP settings for Windows NT 4.0.



The Workstation Service

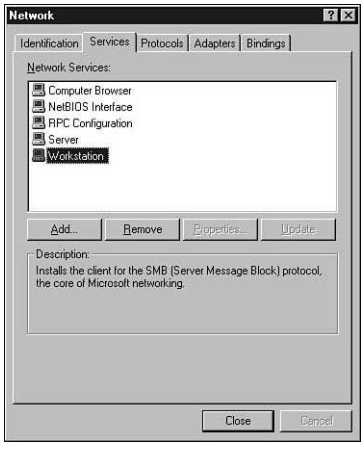
Windows NT 4.0 divides its CIFS client and server implementations into two services. I have already mentioned that the `Workstation` service fills the role of the Client for Microsoft Networks used by Windows 95/98. The complementary service that allows a Windows NT machine to share files to other CIFS clients is named the `Server` service.

The workstation service requires no configuration and is installed by default when you set up networking under Windows NT. Microsoft probably assumed all Windows NT machines would run in some sort of Microsoft network environment, which was a safe assumption. Figure 11.5 shows the Workstation and other services that are installed by default. You simply accept the entries that are set up in the Services tab.



Remember that Microsoft chose the SMB protocol to implement file and printer sharing in its network model.

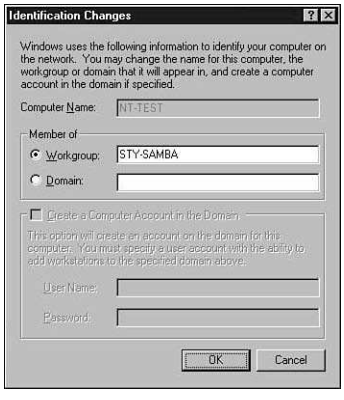
FIGURE 11.5
Default network services installed in Windows NT 4.0.



Setting the NetBIOS Machine Name and Workgroup Name

The final missing piece of information to specify before accepting the changes you have made (by closing the Network control panel) is to set the NetBIOS machine name and workgroup name. By selecting the Change button from the Identification tab, we are able to modify the values as shown in Figure 11.6. The example shown here joins the NT client to the STY-SAMBA workgroup.

FIGURE 11.6
Changing the workgroup membership for a Windows NT 4.0 client.



When you click the OK button to apply the workgroup change, a window should appear saying, "Welcome to the STY-SAMBA workgroup." Of course, *your* message welcomes you to whatever workgroup you entered.



Connecting to a Samba-controlled domain will be covered in Hour 22, "Domain Control for Windows NT 4.0/2000."

Finally we can close the Network control panel and reboot. If all goes well, the NT client will be ready to connect to our Samba server after it has restarted.

Configuring the Windows 2000 Client

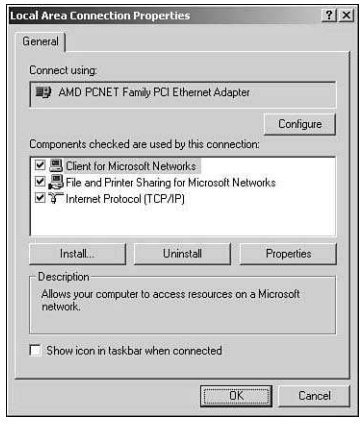
Like the rest of Microsoft's clients, configuring a Windows 2000 workstation to access a CIFS server such as Samba begins with installing the software drivers for the network interface card. Unlike Windows NT 4.0 (but similar to Windows 9x/ME), Windows 2000 possesses support for automatically configuring installed network adapters. If your network card is not automatically detected, you should refer to the support documentation from the NIC's vendor for installation steps.

Once the NIC has been physically installed in the computer and the Windows 2000 OS has been booted, navigate to the Network control panel. From here, view the properties of the "Local Area Connection." The result should appear similar to the window displayed in Figure 11.7.



The Windows 2000 Network control panel is available either by right-clicking on Network Neighborhood and selecting the Properties menu or by opening the Network icon in the system Control Panel (that is, Start, Settings, Control Panel).

FIGURE 11.7
Local Area Connection default installed components.



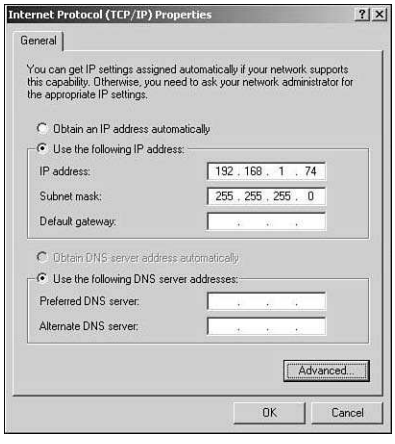
Like Windows NT 4.0, Windows 2000 also splits its CIFS server and client implementation into two services. Windows 2000, however, names these services more in line with Windows 9x/ME. Figure 11.7 displays the familiar "Client for Microsoft Networks" entry for the CIFS client and the "File and Printer Sharing for Microsoft Networks" to indicate the CIFS server is also installed.



The server component is not required to connect to a Samba server. It is provided only to allow other CIFS connects to access shared resources on the host. With that said, many other network features in Windows 2000 *do* require the server component. Therefore, if you remove it, many things, such as remote administration tools, will break.

The TCP/IP configuration screen displayed in Figure 11.8 indicates that Windows 2000 supports static IP addresses as well as assignment via DHCP. The Advanced TCP/IP Settings window (see Figure 11.9) provides access to other related options, such as advanced DNS settings and configuring the client to use a WINS server.

FIGURE 11.8
Basic Windows 2000 TCP/IP settings.





Under most normal circumstances, it is not necessary to reboot the Windows 2000 client to apply the new network settings.

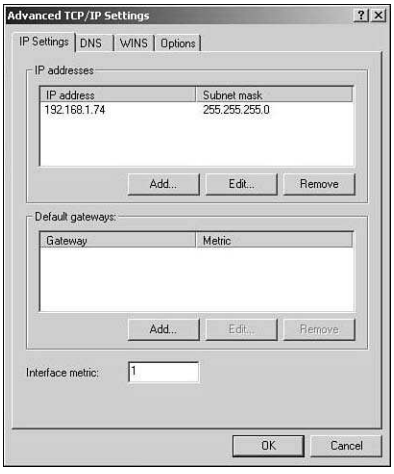


FIGURE 11.9
Advanced Windows 2000 TCP/IP settings.

NetBIOS Names and Workgroups

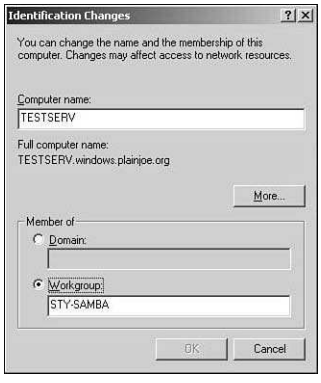
One of the striking changes in the network configuration interfaces between Windows NT 4.0 and Windows 2000 is that the machine's NetBIOS name and workgroup (or domain) settings have been moved out of the Network control panel and into the System Properties applet, which is shown in Figure 11.10. This window can be accessed by either selecting the properties of the My Computer desktop icon or by opening the System icon in the global Control Panel.



FIGURE 11.10
The Windows 2000 System Properties configuration applet.

By selecting the Properties button from the System Properties window, the NetBIOS name and workgroup can be defined just like we did with the Windows NT 4.0 client. Figure 11.11 shows the view when setting the client to be a member of the STY-SAMBA workgroup.

FIGURE 11.11
Setting the workgroup name to STY-SAMBA for a Windows 2000 client.



Whereas it is often not necessary to reboot a Windows 2000 host after making changes to the TCP/IP configuration, applying changes to NetBIOS settings still requires the system to restart. Once the client has rebooted, the system is ready to access our Samba server or other Windows servers.

Conflicting Credentials

There is one Windows NT/2000 annoyance that merits its own section. The Windows NT/2000 network redirector will not allow connections to multiple shares using different user accounts within the same NetBIOS session. To experience the affect of this, first connect to a file share on the Samba server using your normal login account. I have chosen to map my home directory from the server POGO.

```
C:\WINNT\> net use h: \\pogo\jerry /user:jerry
The command completed successfully.
```

If we now attempt to map another network drive to \\pogo\public as a different user (e.g. guest1), Windows NT will complain of a conflicting set of credentials.

```
C:\WINNT\> net use t: \\pogo\public /user:guest1
System error 1219 has occurred.

The credentials supplied conflict with an existing set of credentials.
```

One simple workaround for this behavior is to connect to POGO using its IP address instead of its NetBIOS name. This causes the OS to establish a second NetBIOS session and thus avoid the redirector limitation.

```
C:\WINNT\> net use \\192.168.1.75\public /user:guest1
The command completed successfully.
```

If more than two user accounts are to be used for concurrent connections from the same client, the `netbios aliases` parameter can be used to create additional names for the server. These names can then be used in place of the IP address for the second and third connections.

In this smb.conf file, we have added aliases for POGO named BINGO and PONG.

```
[global]
     <...>
     netbios name = POGO
     netbios aliases = BINGO PONG
     <...>
```

Now we can utilize up to four different user accounts when connecting to the server.

```
C:\WINNT\> net use h: \\pogo\jerry /user:jerry
The command completed successfully.

C:\WINNT\> net use h: \\bingo\public /user:guest1
The command completed successfully.

C:\WINNT\> net use h: \\pong\guest2 /user:guest2
The command completed successfully.

C:\WINNT\> net use h: \\192.168.1.75\software /user:guest3
The command completed successfully.
```

Summary

Configuring Windows NT and Windows 2000 clients to connect to a remote CIFS server such as Samba on a TCP/IP network requires three components:

- A working network interface card.
- A correctly configured version of the TCP/IP network protocol stack.
- Some type of CIFS network redirector. Windows 2000 calls this the Client for Microsoft Networks, whereas Windows NT implements this in the `Workstation` service.

In Hour 12, “CIFS Clients for Unix,” we will examine more CIFS clients. This time, however, the clients will be Unix-based systems instead of Microsoft Windows machines.

Q&A

Q Can I install both the IPX/SPX network protocol and TCP/IP on my computer?

A Yes, but often this can cause problems with network browsing. People have reported not being able to locate servers on the network when both protocols are installed.

Q Why does Windows NT complain and print the error message 2138 when I try to map network drives?

A The Workstation service must be started in order for you to access the network. You can manually start the service by typing **`net start workstation`** in a command prompt window.



HOUR 12

CIFS Clients for Unix

The past two hours have focused on accessing a CIFS (for example, Samba) server from Windows and DOS based clients. It is easy to believe that these are the only types of clients Samba or Windows servers need concern themselves with. There are actually many uses for CIFS clients hosted on Unix machines. Some common examples would be to mount a CIFS file system on a Unix host in order to back up the remote files using a local tape drive or to send jobs from a Unix client to a printer on a remote Windows server.

Samba includes several client utilities. There are also many others, both free and commercial, provided by companies such as Apple, Objective Development, and Thursby Software.

Samba's list of client tools include

- `smbclient`—a standard part of Samba that provides command-line access to CIFS resources. Its FTP-like interface can be used to copy files between Unix and Windows machines and spool files to remote printers. Often, `smbclient` is wrapped in shell scripts; for example, `smbprint` (a print filter for spooling jobs to a remote CIFS printer) and `smbtar` (a script for backing up CIFS file systems).

- `smbfs`—a generic term for a virtual file system that allows Windows file systems to be mounted on Linux file systems. Under Linux, the file system consists of two parts: (a) the file system implementation, which is included in the 2.2 and 2.4 kernel source, and (b) the utilities included with the Samba source code (`smbmount`, `smbmnt`, and `smbumount`) for mounting the file system.

  There is also an implementation of a CIFS file system written by Boris Popov that is available for FreeBSD. The complete source code, including mount tools and documentation, for this file system can be downloaded from `http://people.freebsd.org/~bp/indexen.html`.

Commercial CIFS clients exist for Unix variants as well.

- Sharity, from Objective Development (`http://www.obdev.at/`)—a commercial CIFS file system which is available for Linux, Mac OS X, HP-UX, Solaris, IRIX, *BSD and BSDi, AIX, and a few other operating systems. There is also an older version called Sharity-light that is distributed under the GNU GPL. However, the free version is not available on as wide a variety of platforms as the commercial version.
- Thursby Software (`http://www.thursby.com`) has been selling a version of a CIFS client for the Mac operating system for years. The latest versions also cover Mac OS X.
- Apple is also planning on packaging a CIFS client with later versions of OS X.

During this hour, we will focus our attention on the tools that are included with Samba. We will see how Linux's `smbfs` and `smbclient` can be used to manage files on remote Windows servers and how `smbclient` can be wrapped in scripts to help manage daily network administration duties such as backups and printing.

The Linux Kernel and `smbfs`

Many modern Unix implementations include a file system abstraction layer that allows the heart of the OS, the kernel, to support the operations necessary for storing and retrieving data from many different types of file systems. The Virtual File System (VFS) depicted in Figure 12.1 shows how this type of architecture allows the Linux kernel to access to a local CD-Rom using the `iso9660` file system, an `ext2` file system on the local hard disk, a remote CIFS file system, and support a remote NFS mount as well. This diagram should remind you of the Windows network redirector described in Hour 10, "Microsoft's DOS Network Client and Windows 95/98/ME" (see Figure 10.1).

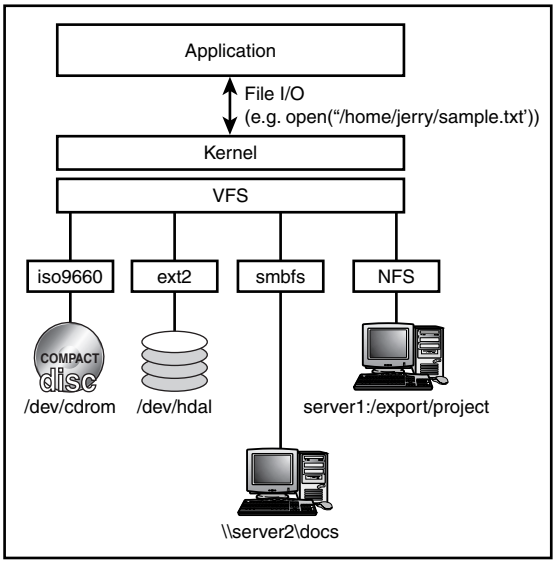


FIGURE 12.1
The VFS in the Linux kernel.

Two pieces are required in order to use the `smbfs` file system on a Linux client.

- A kernel that includes `smbfs` support.
- The utilities for mounting and unmounting an `smbfs` file system.

Practically all current Linux vendors include `smbfs` support in the default kernel for their distribution. It is very easy to verify whether or not this is the case with a particular client by examining the contents of the file `/proc/filesystem`.

```
root# cat /proc/filesystems
nodev   proc
nodev   sockfs
nodev   tmpfs
nodev   pipefs
        ext2
        iso9660
nodev   nfs
        reiserfs
nodev   devpts
nodev   autofs
nodev   usbdevfs
        vfat
```

If the smbfs filesystem type cannot be found in the resulting list, it may be that the support was included as a kernel module that has not been loaded yet. If this is the case, the following command will manually load the specified module and should return without any errors.

```
root# modprobe smbfs
```

We can then reexamine the list of file systems and confirm that smbfs is now supported by our kernel.

```
root# cat /proc/filesystems
nodev   proc
<...output deleted...>
nodev   usbdevfs
        vfat
nodev   smbfs
```

If the client does not possess smbfs support even after attempting to load the correct module, it will be necessary to compile a new kernel to the system. The Linux Documentation Project (`http://www.linuxdoc.org/`) has a document, aptly named the Kernel-HOWTO, which describes the process of compiling and installing a new Linux kernel.

Once our Linux client has a working kernel that includes smbfs support, the next step is to ensure that the Samba tools for mounting and unmounting filesystems are installed. There are three utilities that must be present

- `smbmount`
- `smbmnt`
- `smbumount`

To build these tools from the Samba source code, the `--with-smbmount` must be passed to the `./configure` script. After a successful execution of `make install`, the resulting binaries are built and installed in the default location of `/usr/local/samba/bin/`. We will move these to a directory in our path, such as `/usr/bin/`, so that they will be accessible to users.

```
root# mv /usr/local/samba/bin/smbmount /usr/bin
root# mv /usr/local/samba/bin/smbmnt /usr/bin
root# mv /usr/local/samba/bin/smbumount /usr/bin
```

There is one more detail that must be taken care of if we are to be able to access CIFS file shares using the standard `mount` command. When `mount` is invoked using the `-t smbfs` option, it will in turn look for a program named `mount.smbfs` and pass the remaining command line arguments to it. The `mount.smbfs` command is really just another name for the `smbmount` program. Therefore, the simplest means of meeting this requirement is to create a link to `smbmount` with the appropriate name. Because the `mount.smbfs` tool must be in the root's search path, a common location to place the link is in the `/sbin/` directory. We can create this link by executing

```
root# ln -s /usr/bin/smbmount /sbin/mount.smbfs
```

There is no special step for enabling the `umount` tool to remove an `smbfs` connection to a filesystem.

Assuming that a CIFS server on the network named `CLEOPATRA` provided a file share named `PUBLIC`, we could test our new capability by mapping this service to a directory named `/mnt/smb`.

```
root# mount -t smbfs -o username=user1,password=secret //cleopatra/public /mnt/smb
```



> The `mount` command will fail if the directory specified as the mount point (i.e. `/mnt/smb`) does not exist.

It is immediately apparent that an SMB filesystem has a few non-standard options in addition to the `mount` command. Given our understanding of the CIFS protocol at this point, it should be obvious why the username and password is required. A CIFS server authenticates users (even if the person is using a guest account). It requires this information in order to enforce access control rules to shared resources, such as files and printers.

If a username is not specified on the command line, `smbmount` will use the name of the currently logged on user. If a password is not given, it will prompt the user to enter one as necessary. It is also possible to obtain this information from specific environment variables. The `smbmount(8)` man pages have the full details on these configuration settings.

There are several other options for the `smbmount` command. They are summarized in Table 12.1.

TABLE 12.1 `smbmount` (`mount.smbfs`) Options

Option	*Description*
`codepage=<arg>`	This option, available only in the Linux kernel 2.4.0 and later, defines the codepage used by the remote server. This must be known in advance. An example setting would be `codepage=cp850` for the Western European code page.
`credentials=<filename>`	Specifies the location of a file that contains the username and password to be used for the CIFS connection. The file should contain the expressions "`username = <value>`" and "`password = <value>`" on separate lines. Because this file contains the clear text credentials needed to connect to a remote server, make sure that it can only be read by the appropriate users.
`debug=<arg>`	Specifies the debug level to use when logging messages. This is analogous to Samba's `log level` parameter.

continues

TABLE 12.1 Continued

Option	*Description*
`dmask=<arg>`	Defines the permissions bits to be associated with each directory contained in the `mount`. The default value is calculated from the `umask` of the current process.
`fmask=<arg>`	Defines the permissions bits to be associated with each file contained in the `mount`. The default value is calculated from the `umask` of the current process.
`gid=<arg>`	Defines the numerical `gid` that will be shown as the group owner of all files and directories in the CIFS file system.
`guest`	Instructs `smbmount` not to prompt for a password.
`charset=<arg>`	This option, available only in the Linux kernel 2.4.0 and later, defines the character used from `codepage` conversions. The argument should take the form of a character set name such as `iso8859-1`.
`ip=<arg>`	Specify the IP address of the remote CIFS server.
`netbiosname=<arg>`	Define the NetBIOS name of the local host to be used in the NetBIOS Session Request.
`password=<arg>`	Specifies the password to be used when connecting to the CIFS server. If this option is not defined, `smbmount` will use the value of the `PASSWD` environment variable if present.
`port=<arg>`	Specify the port to connect to on the remote CIFS server. This defaults to 139.
`ro`	Mount the share for read-only access.
`rw`	Mount the share for read-write access.
`scope=<arg>`	Define the NetBIOS scope to be associated with the value of the `netbiosname` option.
`socketopt=<arg>`	Sets the TCP socket options as described in the `smb.conf`(5) man page.
`ttl=<arg>`	This option is available only in the Linux kernel 2.4.2 and later. It specifies the maximum time in milliseconds to cache a directory listing. The default value is 1000ms. Longer times, such as 10000ms (10 sec) can improve performance for large directories.
`uid=<arg>`	Define the numerical `uid`, which will be shown as the owner of all files and directory in the CIFS file system.
`username=<arg>`	Defines the `username` used when connecting to the CIFS server. If this option is not present, `smbmount` will use the value of the `USER` environment variable instead.
`workgroup=<arg>`	Specify the `workgroup` to be used when connecting to the remote CIFS server.

Once we have a mounted filesystem, the files and directories contained inside can be manipulated using standard Linux tools. For example, we can view the contents of the root of the share using the `ls` command.

```
root# ls -l /mnt/smb
total 4
drwxr-xr-x    1 root     root          512 Sep 15 14:48 .
drwxr-xr-x   22 root     root          482 Jul 20 09:19 ..
drwxr-xr-x    1 root     root          512 Sep 15 13:17 New Folder
-rwxr-xr-x    1 root     root            0 Aug 20 17:09 afile
drwxr-xr-x    1 root     root          512 Sep  9 16:56 lj2100
drwxr-xr-x    1 root     root          512 Sep  5 11:20 lj4
drwxr-xr-x    1 root     root          512 Aug 27 14:24 made by foo
-rwxr-xr-x    1 root     root          202 Sep  4 13:31 setupprt.cmd
```

There are a few points to make about our example.

- All files and directories in the filesystem appear to be owned by `root`, even though the connection was made with the account `user1`.
- All files have the permission displayed as `rw-r-r--` and directories are shown to have `drwxr-xr-x`.

By default, `smbfs` maps all file and directory ownership to the `uid` and `gid` of the current user (the root user in our example). This is only a cosmetic feature, however, as the CIFS server ultimately determines all access control. It is not hard to imagine a scenario in which this could be very confusing to a user. Suppose that `CLEOPATRA` had been a Windows NT server that used NT Access Control Lists (ACLs) on the contents of the `PUBLIC` share. In this case, each file/directory could be owned by a different user or group and have its own set of permissions. Although it is not a technical impossibility, the current implementation of Linux's `smbfs` is unable to display that level of information.

When a CIFS server attempts to enforce its local access control mechanisms on objects, it knows only of the user that originally connected to the share. In the case of an `smbfs` filesystem, this means that all access is performed under the guise of the user that was used to mount the filesystem (`user1` is one example), regardless of the identity of the user actually reading or writing the file.

To make this more clear, imagine that we had mounted the filesystem using the `fmask` and `dmask` options to grant access to members of the Accounting group (`gid=520`).

```
root# mount -t smbfs -o username=user1,password=secret,\
> fmask=660,dmask=770,gid=520 //cleopatra/public /mnt/smb
```

Now if we view the contents of the root of the share, the permissions have been set to allow access by members of the Linux group `acct`. However, this only affects how the Linux kernel enforces local access to the mounted filesystem. Figure 12.2 illustrates how

this operation is translated into a CIFS command and transmitted to the server. Whenever a user on the Linux client attempts to read or write a `file/directory`, it is the request sent under the context of the connected user credentials to the CIFS server that ultimately determines whether or not the access is granted or denied.

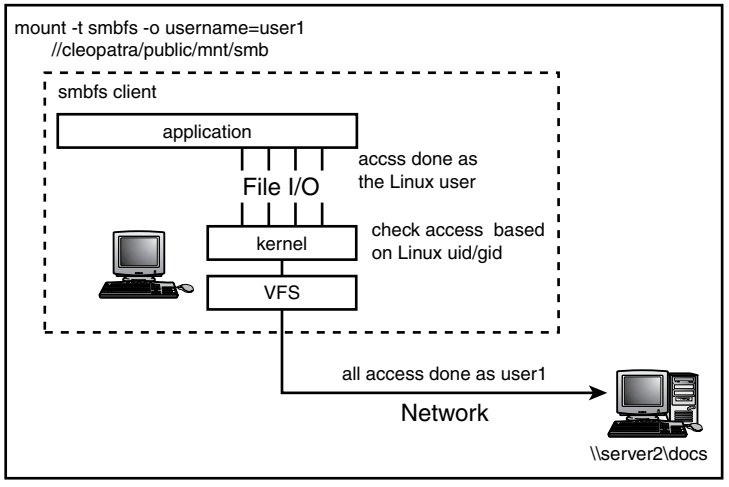


FIGURE 12.2
Accessing a file on an `smbfs` *filesystem.*

Allowing Normal Users to Mount `smbfs` Filesystems

Normally, only the root user is allowed to mount and unmount filesystems on the Linux client. Given the user-centered nature of CIFS connections, there are foreseeable circumstances under which normal users would need to initiate a connection for themselves. Assuming that the binaries are owned by the root account, this can be done by enabling the `setuid` on `smbmnt` and `smbumount`.

```
root# chmod u+s /usr/bin/smbmnt
root# chmod u+s /usr/bin/smbumount
```

Normal users can now mount an `smbfs` filesystem by executing the `smbmount` utility. `smbmount` uses the same options presented in Table 12.1. This example mounts the same file share from `CLEOPATRA` as in the previous section, but as a normal user.

```
$ smbmount -o username=user1,password=secret //cleopatra/public ~/public
```

The `smbumount` command is used to remove a connection to a CIFS server.

```
$ smbumount ~/public
```

smbclient

After seeing how a CIFS file share can be mounted as a normal filesystem on a Linux client, you may wonder what the advantage is of `smbclient's` primitive, FTP-like interface. Though `smbclient` does not support the level of interactivity that a filesystem offers, there are several advantages that it has over `smbfs`.

- `smbclient` is supported on all of the same platforms as Samba itself. The `smbfs` covered so far is available to Linux only, although there are similar implementations for other platforms.
- `smbclient` has the capability to send files to remote printers as well as the capability to access file shares.
- `smbclient` has the capability to enumerate shares and to obtain the browse list and known master browsers from the server. We have already seen some use of this as a means of testing our server.

There are quite a few options and internal commands provided by `smbclient`. Rather than describe them one by one, we will look at several situations and how `smbclient` can be used to solve a given problem.

Before addressing a specific task, such as retrieving a remote file, we will first examine some of the tool's common command line parameters. The general syntax used when invoking `smbclient` is

```
$ smbclient sharename [password] [options]
```

If we were trying to connect to `\\cleopatra\public` again, but this time were using `smbclient` instead of `smbmount`, the command would appear as

```
$ smbclient //cleopatra/public -Uuser1%secret
added interface ip=192.168.1.200 bcast=192.168.1.255 nmask=255.255.255.0
Domain=[NILE] OS=[Windows NT 4.0] Server=[NT Lan Manager 4.0]
smb: \>
```

The `-U` switch specifies the username to be used in the `SMBsessetup&X` request. The optional `%password` syntax provides a means of passing all the information necessary to establish the connection from the command line. The prompt `smb: \>` indicates that the connection was successful and that `smbclient` is ready to receive the next command.

The complete list of `smbclient's` options and internal commands is given in Tables 12.2 and 12.3. These are here for reference. Feel free to skim over them at first and continue on to the example sections that follow.

TABLE 12.2 `smbclient` Command Line Options

Option	*Description*
`sharename`	The UNC path of the share to connect to in the form of `//server/share`.
`password`	The `password` to use with the specified `username` when connecting to the remote server.
`-A <filename>`	Specify a file that contains the `username` and `password` to be used when connecting to the remote server. The format of the file is identical to the one used by the `smbmount` tool and it also carries with it the same warnings regarding insecure permissions. See the `smbclient(1)` man pages for more details.
`-d <debug level>`	Set the debug level at which to log messages. This option is analogous to the `log level smb.conf` parameter.
`-E`	Write all messages to standard error instead of standard output.
`-h`	Display the list of available command line options.
`-I <ip address>`	Specify the IP address of the remote server.
`-i <scope>`	Define the NetBIOS scope to be associated with the client's NetBIOS name.
`-L`	Enumerate the list of shares, workgroups, and master browsers known by the remote server.
`-l <log filename>`	Define the location of the file that should be used to record all debug messages.
`-M <netbios name>`	Send a Winpopup style message to the client with the given NetBIOS name.
`-N`	Perform the connection as an anonymous user (a NULL `username` and `password`).
`-n <netbios name>`	Define the NetBIOS name of the local host to use in the NetBIOS Session Request when connecting to the remote server.
`-O <socket options>`	Specify the TCP socket options to use as described in the `smb.conf(5)` man page.
`-p <port>`	Define the port on the remote machine to use in the connection. The default is to use port 139.
`-R <name resolve order>`	Specify the order of name services to try when resolving a NetBIOS name as described by the `name resolve order` parameter in the `smb.conf(5)` man page.
`-s smb.conf`	Define an alternative configuration to use in the place of the compile time default.

continues

TABLE 12.2 Continued

Option	*Description*
`-T <tar options>`	Instruct `smbclient` to run in "tarmode" for backing up and restoring files/directories on remote servers.
`-t <terminal code>`	Inform `smbclient` how to interpret filenames by specifying the Unix character set to use when converting filenames from a DOS codepage. See the `smbclient(1)` man page for examples.
`-U <user[%pass]>`	Define the `username` (and optionally the `password` as well) to be used by establishing a connection with the remote server.
`-W <workgroup>`	Define the `workgroup` name to be used when connecting to the remote CIFS server.

TABLE 12.3 `smbclient` Internal Commands

Command	*Description*
`? [command]`	Print the list of available commands. If the '`?`' character is followed by a command, the help message for that command is printed to standard output. A synonym for this command is `help`.
`! [shell command]`	Execute the specified shell command. If no command is specified, a local shell will be launched.
`blocksize <size>`	Define a valid blocksize, which must be greater than `0`.
`cd [directory name]`	Change to the directory on the remote file system.
`del <mask>`	Remove all files matching the specified `<mask>`. A synonym for this command is rm.
`dir <mask>`	List all files in the current directory matching the specified `<mask>`. A synonym for this command is `ls`.
`exit`	Quit the `smbclient` session. A synonym for this command is `quit`.
`get <remote file> [local file]`	Download the `<remote file>` to the local current directory. If a `[local file]` is given as well, it will be the name of the new local file.
`lcd [directory name]`	Change the local current directory to the specified path. If no `[directory name]` is defined, the command prints the current local directory path.
`lowercase`	Toggle the flag that controls whether or not downloaded file names are converted to lower case.
`mask <mask>`	Define a wildcard mask to be used for all `get` and `mget` operations.

continues

TABLE 12.3 Continued

Command	*Description*
`mget <mask>`	Download all files matching the specified `<mask>`.
`mkdir <directory name>`	Create a new directory on the remote file system. A synonym for this command is `md`.
`mput <mask>`	Upload all files in the local current directory that match the specified `<mask>` to the current directory on the remote file system.
`print <filename>`	Spool the specified file to a printable service.
`printmode <graphics|text>`	Set the print mode to accommodate either graphics- or text-based files when submitting jobs for printing.
`prompt`	Toggle the flag that controls whether or not `smbclient` should prompt for filenames when issuing `mput` and `mget` commands.
`put <local file> [remote file]`	Upload the file from the current local directory to the current working directory on the remote file system.
`queue`	Display the list of print queue entries for a printable service.
`recurse`	Toggle the directory recursion flag for `mput` and `mget` commands. See the `smbclient(1)` man page for details.
`rmdir <directory name>`	Remove the specified empty directory from the remote file system. A synonym for this command is `rd`.
`setmode <file> <perm=[+|\-]rsha>`	Set/unset the specified DOS attributes on the given file.
`tar <c|x>[IXbgNa]`	Performs an operation identical to the `-T` command line switch. Refer to the `smbclient(1)` man page for full details.
`tarmode <full|inc|reset|noreset>`	Set the behavior of the `tar` command in relation to the archive bit on files. See the `smbclient(1)` man page for details.

In the next sections, we will see how to use `smbclient` to

- Upload and download files from a file share.
- Back up a complete directory.
- Send a file to a remote printer.

Using `smbclient` to `get` and `put` Files

One of the most common uses of `smbclient`, other than as a diagnostic tool, is to upload and download files from CIFS shares. In this case, the commands used by `smbclient` are

very similar to the ones supported by bare-bones FTP clients. Once again, we will connect to `\\cleopatra\public`, as with the account `user1`.

```
$ smbclient //cleopatra/public -Uuser1%secret
```

Once logged in, we can view the contents of the top-level directory by running the built-in `ls` command.

```
smb: \> ls
  .                                   D        0  Sun Sep  9 16:55:59 2001
  ..                                  D        0  Sat Sep 15 10:00:24 2001
  lj4                                 D        0  Wed Sep  5 11:20:47 2001
  afile                               A        0  Mon Aug 20 17:09:05 2001
  New Folder                          D        0  Mon Aug 27 18:17:35 2001
  lj2100                              D        0  Sun Sep  9 16:56:01 2001
  made by foo                         D        0  Mon Aug 27 14:24:33 2001
  group.cap                           A    57612  Thu Aug 23 00:18:52 2001
  setupprt.cmd                        A      202  Tue Sep  4 13:31:52 2001
                36204 blocks of size 262144. 3041 blocks available
```

Though certain pieces of information that we are accustomed to receiving from the `/bin/ls` command are absent (for example, permissions and ownership), `smbclient` does reveal some DOS/Windows-specific information, such as the archive attribute bit (`A`) on the `setupprt.cmd` file.

Next, we can change to the `"New Folder"` directory by issuing the `cd` command.

```
smb: \> cd "New Folder"
smb: \New Folder\>
```

In order to upload a file to the current directory, it is necessary to set the local current directory as well. In order to upload the local client's `/etc/hosts` file, first we need to use the `lcd` command to change to the local `/etc` directory.

```
smb: \New Folder\> lcd /etc
the local directory is now /etc
```

Finally, we can use the `put` command to upload the `hosts` file to the `"New Folder"` directory.

```
smb: \New Folder\> put hosts
putting file hosts as \New Folder\hosts (16.8 kb/s) (average 16.8 kb/s)
```

We'll use the `ls` command again to verify that the file was really placed in the current folder.

```
smb: \New Folder\> ls
  .                                   D        0  Sat Sep 15 13:07:21 2001
  ..                                  D        0  Sun Sep  9 16:55:59 2001
  hosts                               A      793  Sat Sep 15 13:07:21 2001
                36204 blocks of size 262144. 3041 blocks available
```

In order to download the `group.cap` file stored in the root of the `PUBLIC` share, we need to move up one level in the directory tree.

```
smb: \New Folder\> cd ..
smb: \>
```

Before retrieving the file, we must first change our local directory to `/tmp` because we do not have write permission to the local `/etc` directory.

```
smb: \> lcd /tmp
the local directory is now /tmp
```

Finally, we can save the desired file to our local disk by issuing the `get` command, followed by the filename to download.

```
smb: \> get group.cap
getting file group.cap of size 57612 as group.cap (150.0 kb/s)
    (average 150.0 kb/s)
```

Since we now have a local copy of the file, we can remove the original by using the `rm` command.

```
smb: \> rm group.cap
```

Although there are many other internal `smbclient` commands available, which are described in Table 12.3, one particular command begs a quick mention.

The `translate` command is similar to the FTP `ascii` command. It informs `smbclient` whether or not it should perform any `LF <-> CR/LF` conversions when retrieving or uploading files. `smbclient's` default behavior is to not perform any automatic conversion of files. If you wish to enable this feature (or disable it again), the `translate` command toggles the setting on and off.

```
smb: \> translate
CR/LF<->LF and print text translation now on
```



The `translate` command should be used only when working with text files, not executable files.

Backing Up a Directory

All Unix systems include a version of the `tar` command for archiving files. `smbclient` also supports a variant of this, using either the `-T` command line option or the `tar` internal command. We will explore the internal `tar` command since both options are

equivalent, and this will provide the opportunity to also examine how to automate `smbclient` operations.

In its most basic form, the `smbclient` can back up either a file or a directory (`c`) to a single file compatible with the Unix `tar(1)` command or extract items from a `tar` file (`x`). Our first example backs up a directory named `\samba-updates`, located on the share `\\cleopatra\public`. The `-c` command line switch is used to provide a list of commands to execute unattended once connected to the share. The file `backup.tar` is created in the local working directory.

```
$ smbclient //cleopatra/public -Uuser1%secret -c "tar c backup.tar samba-
updates"
added interface ip=192.168.1.200 bcast=192.168.1.255 nmask=255.255.255.0
Domain=[NILE] OS=[Windows NT 4.0] Server=[NT Lan Manager 4.0]
                directory \samba-updates\
     50732 (16514.3 kb/s) \samba-updates\network-arch.jpg
     12800 (12500.0 kb/s) \samba-updates\network-arch.ppt
      5899 ( 5760.7 kb/s) \samba-updates\network-arch.ag
    406016 (15860.0 kb/s) \samba-updates\ForSambaHOWTO.doc
tar: dumped 5 files and directories
Total bytes written: 477184
```

The files can be restored again by using the `x tar` option to extract the files.

```
$ smbclient //cleopatra/public -Uuser1%secret -c "tar x backup.tar"
added interface ip=192.168.1.200 bcast=192.168.1.255 nmask=255.255.255.0
Domain=[NILE] OS=[Windows NT 4.0] Server=[NT Lan Manager 4.0]
restore directory \samba-updates\
restore tar file \samba-updates\network-arch.jpg of size 50732 bytes
restore tar file \samba-updates\network-arch.ppt of size 12800 bytes
restore tar file \samba-updates\network-arch.ag of size 5899 bytes
restore tar file \samba-updates\ForSambaHOWTO.doc of size 406016 bytes
tar: restored 5 files and directories
```

Samba also includes a wrapper script specifically designed to make backing up and restoring files easier. The `smbtar` shell script is installed in `/usr/local/samba/bin/` when you execute `make install`. For more details on this, see the `smbtar(1)` man page.

Printing a File

Printing a file to a CIFS printer is very similar to uploading a file to a disk share. First, we will connect to the `COLOR` printer share held by the server `CLEOPATRA` as an anonymous user.

```
$ smbclient //cleopatra/color -Uuser1%secret
added interface ip=192.168.1.200 bcast=192.168.1.255 nmask=255.255.255.0
Domain=[NILE] OS=[Windows NT 4.0] Server=[NT Lan Manager 4.0]
smb: \>
```

Next, we will change to the local directory where the file is stored (for example, `~/Documents`). `smbclient` is not able to handle things like tilde (~) expansion or environment variables for directory paths, so we must enter the absolute path to where we want to go.

```
smb: \> lcd /home/pogo/gcarter/Documents
the local directory is now /home/pogo/gcarter/Documents
```

We will send the file to the printer by using the `print` command.

```
smb: \> print resume.ps
putting file resume.ps as resume.ps (303.0 kb/s) (average 303.0 kb/s)
```

Because `smbclient` does not support the concept of filtering the job before sending it to the remote printer, the file should already be a format that can be understood by the printer. Common formats include Postscript or PCL. This also means that you must have some knowledge about the printer to which you are connected.

We can then verify that the file has been submitted to the printer by executing the `queue` command.

```
smb: \> queue
2        243594       Remote Downlevel Document
```

This does not give us a lot of information about the file in the print queue. In this case, the displayed filename is also incorrect. However, we can verify the file size to ensure that our file was spooled to the printer.

The `smbprint` Script

The `smbprint` utility is a simple shell script included with Samba that allows a Unix system to print to printers on other CIFS/SMB servers. `smbprint` uses `smbclient`'s command-line facilities and its capability to handle STDIN to process the file to be printed all in one simple pipeline. For example,

```
(
     echo translate
     echo print -
     cat
) | smbclient "//$server/$service" -U $user%$pass
```

This syntax is not exactly the same as what `smbprint` uses, but it illustrates the same principles.

1. In a subshell, two `echo` statements tell `smbclient` to translate all input and to print from STDIN (-). You execute these commands in a subshell (surrounded by parentheses) so you can send a stream of commands into `smbprint` via STDIN.
2. The `cat` command copies from STDIN to STDOUT, but STDOUT is connected to the next stage of the pipeline, and STDIN is connected to the file to be printed by

`lpd`. This command is also executed in the subshell that the previous two are executed in.

3. `smbclient` runs, connects to the server specified in `$server` on the service specified in `$service` (where `$server` and `$service` are set up by `smbprint`) using the other parameters passed to it, and processes STDIN.
4. However, because STDIN is connected to the subshell mentioned previously, it receives the `translate` command, then the `print -` command, and finally, the file to be printed, all on STDIN.

Printing from Unix to Windows

It is often useful to be able to print from your Unix servers to a printer that is connected to a Windows client. This can be achieved by using the appropriate version of the `smbprint` tool as a print filter. We will look at how to do this for Berkeley-style printing (LPD, PLP, and LPRNG) as well as System V-style printing.

Samba comes with a small shell script called `smbprint` (for the BSD version) and `smbprint.sysv` (for System V systems). As previously described, this program accepts print jobs and queues them to printers shared on remote Windows systems. The steps for configuring this are different depending on whether your Unix system uses BSD-style printing or System V-style printing.

In either case, the printer must be shared on the Windows machine. If the Windows host has the "File and Printer Sharing for Microsoft Networks" service installed (Windows NT/2000 clients have this ability by default), simply selecting the sharing tab from the printer Windows printer properties dialog and enabling the feature is enough—often all that is required. Figure 12.3 shows how to share a printer from a Windows 98 system.

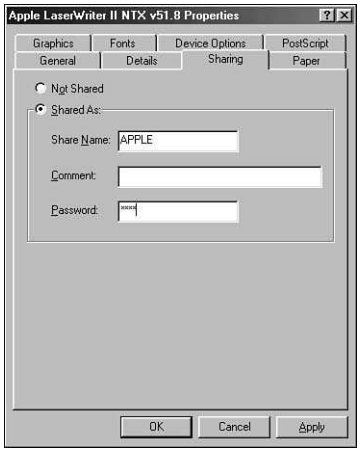


FIGURE 12.3
Sharing a printer named APPLE from a Windows 98 host.

Printing to Windows with BSD-Style Printing

BSD-style printing uses LPD and is controlled from a file called `/etc/printcap`. All printers are defined in the `printcap` file. It is a good idea to refer to the `printcap` man page for the full details on configuring remote LPD printers.

To specify that a particular printer should send its print jobs to a shared printer on a remote Windows 9x/ME system, perform the following steps:

1. Define the printer; that is, build a `printcap` entry either manually or with your favorite tool.
2. In the spool directory created for the printer, usually `/var/spool/<printer-name>`, create a file called `.config` (make sure that the name starts with a period) and place the following lines in the file:

   ```
   server=<server name>
   service=<printer share name>
   password=<password>
   ```

3. Change (or add) the input filter in the `printcap` entry for your printer to

   ```
   :if=<directory path to smbprint>/smbprint:
   ```

4. Restart the printer.

For a printer named `MYPRINTER` provided by the remote server `HAWK` that required no password for access, the `.config` file would look like

```
server=HAWK
service=MYPRINTER
password=""
```



For printers located on Windows NT systems, it is possible to use the Windows TCP/IP Printing service and install LPD on NT/2000.

Printing to Windows with System V-Style Printing

System V-style printing is different from BSD-style printing and uses lp. Samba supports printing to Windows machines from System V systems using the script `smbprint.sysv` located in the `examples/printing` directory within the Samba source tree.

This script is a modified version of the BSD-style script. To use it, make a modified version of the script to specify the Windows server, service, and service password. These are located in a block in the program:

```
server=admin
service=hplj2
password=""
```

Change each of these to the correct values, and then install the script as an interface script for your queue and start printing, using

```
root# lpadmin -punixprintername -v/dev/null -i./smbprint.sysv
root# enable unixprintername
root# accept unixprintername
```

For example, if the printer's name is `hawk_print`, the remote server is `HAWK`, the printer share is called `MYPRINTER`, and no password is required to access the remote share, the changes to `smbprint.sysv` required would look like

```
server=HAWK
service=MYPRINTER
password=""
```

The commands required to set up and enable this printer would be

```
root# lpadmin -phawk_print -v /dev/null -i./smbprint.sysv
root# enable hawk_print
root# accept hawk_print
```

Resolving Hostnames Using the NetBIOS Name Service

Unix network administrators often complain about NetBIOS name resolution and Windows clients because of the dynamic nature of client (and server) names. The root of the complaint is that, unless the NetBIOS host has been configured in DNS, certain standard Unix tools, such as `ping`, will fail to contact a machine using its NetBIOS name.

Samba includes an NSS module that can be used to query the NetBIOS Name Service on a network to resolve a hostname. Despite its name, `libnss_wins`, this library will also resolve hostnames using broadcast mechanisms if a WINS server cannot be contacted. WINS is fully covered in Hour 18, so we won't go into it here.

Samba does not build the `libnss_wins` module by default, since it is only available on Linux. However, only a single extra step is required to create the shared library in the `source/nsswitch` directory. From the `source/` subdirectory, execute the following command

```
$ make nsswitch/libnss_wins.so
```



The `libnss_wins.so` filename is specific to Linux. In the current 2.2 releases, this module will fail to compile on other platforms, such as Solaris.

Once the library has been built, we must copy it to the /lib directory and set the ownership and permissions as shown here.

```
root# cp -p nsswitch/libnss_win.so /lib/libnss_wins.so.2
root# chown root /lib/libnss_wins.so.2
root# chgrp root /lib/libnss_wins.so.2
root# chmod 755 /lib/libnss_wins.so.2
```

The `libnss_wins` NSS module uses the NetBIOS name resolution settings from `smb.conf` just like `smbd` and `nmbd`. This will include the WINS server settings to be covered in Hour 18. For the moment, we will simplify the name resolution settings by setting the `name resolve order` parameter in `smb.conf`.

```
[global]
     ...
     name resolve order = bcast
```

The `libnss_wins` library expects the configuration file to be located in the compile default location (i.e. `$prefix/lib`) since there are no means of passing a command line parameter to the module.

The final configuration step is to add the `wins` service name to the `hosts` database line in `/etc/nsswitch.conf`.

```
hosts:          files dns wins
```

This line means that the local `/etc/hosts` file should be consulted first, followed by the DNS servers specified in `/etc/resolv.conf`, and finally Samba's `name resolve order` setting will be used for querying the NetBIOS Name Service.

Assuming that the NetBIOS host `CLEAOPATRA` is located on our local network, we can now `ping` the machine using its name even if the machine has not been added to our DNS zone.

```
$ ping cleopatra
PING cleopatra (192.168.1.140): 56 data bytes
64 bytes from 192.168.1.140: icmp_seq=0 ttl=128 time=66.551 ms
64 bytes from 192.168.1.140: icmp_seq=1 ttl=128 time=0.800 ms
```

This step can fail if NetBIOS name resolution has not been properly configured. The process of troubleshooting a failure such as this will be the topic of the next hour when we discuss various methods of troubleshooting a Samba installation.

Summary

During this hour we have examined two client tools provided with Samba for Unix hosts. Although the `smbfs` implementation covered here was specific to Linux, other versions exist, both commercial and Open Source. The ability to mount a CIFS file system provides support for standard Unix tools such as `vi` (or `emacs`), `find`, `rm`, `cd` and `mkdir`.

If a CIFS file system is not available for your particular Unix system, Samba provides an FTP-like tool named `smbclient`, which supports connecting to both file services and shared printers. There have been many scripts developed around `smbclient` to automate daily tasks such as printing, backups, and general network management. The `smbtar` and `smbprint` scripts included with Samba are examples of this, and they can act as a starting point for writing your own programs to manage remote CIFS resources.

Finally, we learned how the NetBIOS name service can be queried by standard Unix programs to resolving hostnames. The `libnss_wins` NSS module is already packaged by many Linux vendors.

Q&A

Q When I restore files to a Windows 95 machine, `smbclient` sometimes fails after telling me that it could not update the creation time on a file. What do I do?

A Windows 95 and Windows NT behave differently when asked to change the creation date on a file. Windows NT performs the requested operation without any problems, whereas Windows 95 performs the requested action and returns an error response. The version of `smbclient` in Samba 2.0 contains a fix for this problem (it ignores the response from the server when changing the creation date). Make sure that you are not using an extremely old version of `smbclient`.

Q Is there a limit on the length of a filename that can be backed up by `smbclient`?

A In Samba 2.2 there is currently a limit that restricts access to absolute pathnames that are less than 1024 characters. We are planning on removing the limitation in Samba 3.0.

Q How can I specify multiple operations to be executed sequentially by `smbclient` when using the `-c` option from the command line?

A Multiple commands should be separated by a '`;`'. For example, `-c "cd some-directory; get somefile"`.



HOUR 13

Troubleshooting Techniques

I find it interesting that this hour is the first hour after the half-way point of this book—if you are counting hours. It is also the last hour before "Part IV—Going Production."

Why have I waited so long to introduce techniques for troubleshooting Samba? I believe that troubleshooting is a chicken-and-egg type of problem. When you need to know how to troubleshoot a software package, you don't have enough information to know what to do on your own. However, once you have this information, you don't necessarily need a troubleshooting guide because you can pinpoint the problems based on your own experiences. The problem with writing about methods of tracking down and solving broken Samba configurations is that you must have a general understanding of how things should work before you can understand why the techniques are valid. But, as I said before, once you have this knowledge, you can usually figure things out for yourself. It is a vicious cycle, really.

The approach we will take in this hour is to identify a collection of best practices for Samba administrators. These guidelines will be in the form of common error messages and their meanings as well as techniques and avenues for obtaining help when you are absolutely stuck. In the tradition of Unix and Open Source/Free Software, this hour will help you to stand on the shoulders of those who have already been in the position of fixing broken servers and have succeeded.

Egypt, Samba, and Bugs

Wouldn't it be horrible to replace the engine in your car, only to find out later that it was simply out of gas? This may seem like a silly question, but I have seen too many network administrators waste time because they did not check for basic problems first. With this in mind, Figure 13.1 presents a pyramid for analyzing Samba problems.

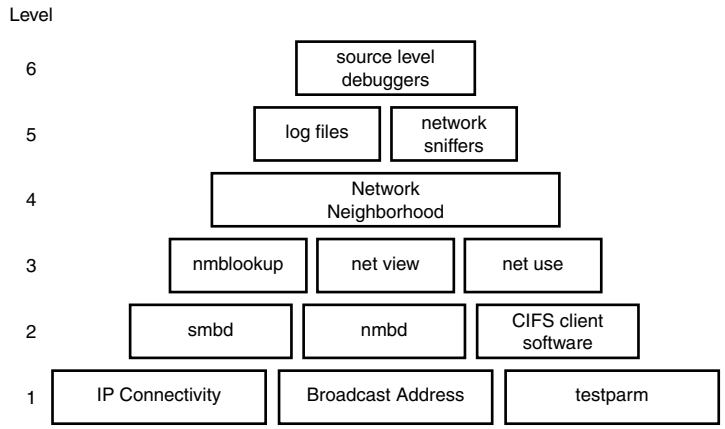


FIGURE 13.1
Analyzing problems with Samba from the bottom up.

The topics presented at the bottom of the pyramid are fundamental to the ones at the upper levels. It is no wonder why Windows clients cannot access a file share on your Samba box if the server is unplugged from the network. Though this representation of troubleshooting requirements doesn't always hold to be true (you can always view log files), it is better to start from the bottom when diagnosing a problem. As we climb higher, involvement with Samba internals grows, ending with the heights (or depths, depending on your point of view) of stepping through the source code line by line using a debugger.

Available Tools at Our Disposal

When searching for the root of a problem, there are three basic resources available to us:

- Existing documentation, such as this book, HOWTO's distributed with Samba itself, and mailing list archives
- Interactive communication with other administrators and users, via means such as IRC, newsgroups, and mailing lists
- Software tools, such as the utilities included with Samba and network monitoring programs

We will spend the majority of the hour focusing on personal troubleshooting efforts. This does not imply that the first two items in this list are not as important. Extra documentation and the advice of experienced administrators can be invaluable. This book could not contain the wealth of knowledge possessed by the Samba community. Who knows? You might even have a chance to return the favor to someone else one day.

Documentation

Much work has gone into cleaning up the documentation for Samba 2.2, although there is still much to be done. One of the major changes was the conversion from a markup language called YODL (*Yet oneOther Document Language*) to SGML/DocBook. One of the products of this is the Samba-HOWTO-Collection book. This collection of tutorials and explanations is available in PDF and HTML format, both of which are included with the Samba distribution. The HTML version is linked from the main page when logging into SWAT.

There are many other individual text files stored in the `docs/textdocs` directory that have yet to be updated and converted. One that receives a great deal of use is the `DIAGNOSIS.txt` file. This file is the original troubleshooting guide for Samba. Although it is fairly old now, it is still the first guide many administrators use when trying to track down a problem.

People

The Samba community of people can be one of the best resources for tracking down problems. You must remember, though, that all the people whom you might contact on the various mailings and newsgroups help because they want to and not because they are paid to. This includes many of those who develop Samba, commonly referred to as the Samba Team. This means that the ultimate burden for fixing your problems still falls on your own shoulders.

When posting to or answering questions on any mailing list or newsgroup, you should follow common Internet etiquette (or Netiquette). If you do not, you will find that people will be less than helpful. However, if you are considerate in your postings, someone will normally respond.

You can find out more information about the Samba mailing lists and how to join them at `http://lists.samba.org/`. Some of the available mailing lists are

- `samba@samba.org`—This is the main Samba mailing list for general information related to installing, configuring, and maintaining Samba servers.
- `samba-technical@samba.org`—This is the mailing list for discussions regarding the development of Samba. If you feel like pitching in, join the list, open up `vi`, and start working through the source code.
- `samba-bugs@samba.org`—This address is not a mailing list, but rather an address for reporting actual bugs in the Samba applications. This is not an address to be used for general help questions.

Many people have found the interactive nature of IRC channels to be more conducive for debugging problems in real time. There are two popular channels hosted by `irc.openprojects.net`. The `#samba-technical` channel often contains Samba Team members and other developers discussing current plans and the future of Samba. The `#samba` channel can be used when seeking help with configuration problems.

The Usenet newsgroup `comp.protocols.smb` is another good source of information about configuring and testing Samba.

The Test Environment

In order to properly address the troubleshooting methods presented in the remainder of the hour, we must make some statements about environment. These tests assume that

- The Samba server named `POGO` is located at IP address `192.168.1.75` using a netmask of `255.255.255.0`. These are specific details that relate to the underlying operating system on which Samba is running.
- Both `smbd` and `nmbd` are configured to start as daemons. Experience has proven that this configuration is much less problematic than having them launched by `[x]inetd` upon demand.
- Our Windows client (95, 98, ME, NT, 2000 or XP) is named `WIN-CLIENT`.
- The Windows system has only the CIFS client (i.e. “Client for Microsoft Networks” or `Workstation` service), a network adapter, and the TCP/IP protocol installed. The client is using address `192.168.1.135` with a netmask of

255.255.255.0. You can refer to Hours 10 and 11 for details on configuring Windows clients.

- Both WIN-CLIENT and POGO are located on the same logical IP network, meaning that a broadcast packet from one machine can be seen by the other.
- Both WIN-CLIENT and POGO are members of the workgroup STY-SAMBA.

The Samba server is using the following smb.conf:

```
[global]
     netbios name = POGO
     workgroup = STY-SAMBA
     security = user
     encrypt passwords = yes

[public]
     path = /tmp
     read only = no
```

With these components in place, we are ready to begin our troubleshooting journey.

Level 1: Beginning the Climb up the Pyramid

At the base of the pyramid of Figure 13.1, three components are presented as having foundational importance. These should be the first items we check if a problem arises.

- General TCP/IP connectivity
- Matching network masks and broadcast addresses among servers and clients
- A working smb.conf

Pinging the Server and Client

One of the basic tools for verifying TCP/IP connectivity is the ping command. At the risk of oversimplifying the ICMP protocol, ping sends a request to a host and asks, "Are you alive?" If the host does not respond, the machine sending the ping request assumes that it is not connected to the network or not currently available (for example, powered off).

First, we will attempt to ping the client from our Unix server.

```
$ ping win-client
PING win-client (192.168.1.135) from 192.168.1.74 : 56(84) bytes of data.
64 bytes from win-client (192.168.1.135): icmp_seq=0 ttl=255 time=2.138 msec
64 bytes from win-client (192.168.1.135): icmp_seq=1 ttl=255 time=2.181 msec
64 bytes from win-client (192.168.1.135): icmp_seq=2 ttl=255 time=2.263 msec

--- pogo.plainjoe.org ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 2.138/2.194/2.263/0.051 ms
```

The output from `ping` will appear slightly different depending upon the operating system installed, but success or failure should be obvious. The previous example was from a RedHat 7.1 host.

One of the fundamental services of TCP/IP networks is the Domain Name Service (DNS). If the server is unable to resolve the hostname to an address, you will see a message similar to

```
$ ping win-client
ping: unknown host win-client
```

If this occurs, the first step is to try the `ping` command again, but using the IP address of the client.

```
$ ping 192.168.1.135
```

If this succeeds, we can point a finger at the DNS configuration. The most common errors are

- The DNS configuration on the server (`/etc/resolv.conf`) is broken
- The `win-client` name does not contain an entry in the DNS zone
- The DNS server is currently unreachable

If we are unable to `ping` the IP address of the client, the next step is to verify that the network card of both the client and the server is installed correctly and functioning properly. You should also ensure that all the network cabling and intermediate components, such as hubs and switches, are connected properly.



Describing how to debug hardware components of either Windows or Unix systems can be a very detailed subject. I am of the opinion that experience is really the best teacher in this case. If you suspect hardware failures, try to find someone who is already familiar with your configuration and ask for advice. Books and other documentation are always helpful, but they are rarely as quick to provide an answer as an experienced administrator.

The `ping` tool is also available on Windows clients. If you cannot find the `ping.exe` executable on the Windows client, make sure that the TCP/IP protocol is listed in the installed network components, which are displayed by the Network Control Panel applet.

Windows' `ping` command produces output similar to the Linux `ping` utility.

```
C:\WINDOWS>ping pogo

Pinging pogo [192.168.1.75] with 32 bytes of data:

Reply from 192.168.1.75: bytes=32 time=19ms TTL=255
Reply from 192.168.1.75: bytes=32 time=3ms TTL=255
Reply from 192.168.1.75: bytes=32 time=15ms TTL=255
Reply from 192.168.1.75: bytes=32 time=6ms TTL=255

Ping statistics for 192.168.1.75:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum =  19ms, Average =  10ms
```

The process of diagnosing a failed `ping` on the Windows client is the same as the one previously described for a Unix server.

Comparing Network Broadcast Addresses

It is possible to `ping` one host from another and still have a misconfigured netmask and broadcast address for the host's network interface. Why would a broken broadcast address matter to Samba?

Remember that our discussion of NetBIOS so far has covered only broadcast-based name registration and resolution. The NetBIOS name services are crucial to features such as the Network Neighborhood and connecting to a shared directory or printer. In Hour 18, "WINS and NetBIOS Name Services," we will see how the NetBIOS name space can be extended beyond a single network segment. Until then we must rely on using broadcast packets to locate NetBIOS clients and servers.

Both `POGO` and `WIN-CLIENT` should be using a network mask of `255.255.255.0` and a broadcast address of `192.168.255`. We can verify this information on our Unix server by running the `ifconfig` command with a single argument of the network adapter's name.

```
$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:04:5A:0C:1C:19
          inet addr:192.168.1.75  Bcast:192.168.255.255  Mask:255.255.0.0
          inet6 addr: fe80::204:5aff:fe0c:1c19/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68006 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100783 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:12186135 (11.6 Mb)  TX bytes:121642120 (116.0 Mb)
          Interrupt:3 Base address:0x100
```

Our broadcast address is `192.168.255.255`, which is incorrect. When run as root, `ifconfig` can be used to set the broadcast address.

```
root# ifconfig eth0 192.168.1.75 netmask 255.255.255.0 broadcast 192.168.1.255
```



Network settings made from a command line are almost always lost when the network interface is reset or the system is rebooted. Refer to your server's documentation for the correct procedure to make this change persistent.

Viewing the current TCP/IP settings on a Windows NT/2000 client can be done with the `ipconfig.exe` tool. The `/all` option instructs `ipconfig` to display detailed information about all installed adapters. This listing is from a Windows NT 4.0 client named `CAESAR`, which obtains its IP settings via DHCP.

```
C:\WINNT\>ipconfig /all

Windows NT IP Configuration

        Host Name . . . . . . . . . : caesar.plainjoe.org
        DNS Servers . . . . . . . . : 192.168.1.1
        Node Type . . . . . . . . . : Broadcast
        NetBIOS Scope ID. . . . . . :
        IP Routing Enabled. . . . . : No
        WINS Proxy Enabled. . . . . : No
        NetBIOS Resolution Uses DNS : No

Ethernet adapter AMDPCN1:

        Description . . . . . . . . : AMD PCNET Family Ethernet Adapter
        Physical Address. . . . . . : 00-50-56-91-01-4A
        DHCP Enabled. . . . . . . . : Yes
        IP Address. . . . . . . . . : 192.168.1.134
        Subnet Mask . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . : 192.168.1.1
        DHCP Server . . . . . . . . : 192.168.1.1
        Lease Obtained. . . . . . . : Thursday, September 20, 2001 1:53:08 PM
        Lease Expires . . . . . . . : Thursday, September 20, 2001 2:03:08 PM
```

Windows 9x/ME systems do not include a command line tool for viewing local TCP/IP settings. These hosts use a graphical program named `winipcfg.exe`. The IP configuration for a Windows 98 client is shown in Figure 13.2.

If either the network mask or the broadcast address is incorrect on this client, refer to the instructions in Hours 10 and 11 on how to set the correct value. If the client obtained its information via DHCP, you should refer to your DHCP server's documentation for possible solutions.

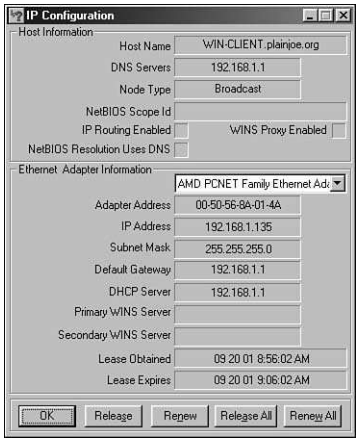


FIGURE 13.2
Viewing the Windows IP configuration using the `winipcfg.exe` *utility.*

Verifying the `smb.conf` Settings

Because Samba uses an extremely large amount of parameters in `smb.conf`, developers have provided a command line tool for verifying the syntax of a configuration file. The utility, named `testparm`, has already been described in Hour 4, "Starting Your Feet to Dance." It is being mentioned here again so that we do not overlook simple `smb.conf` errors when hunting for the source of a problem.

`testparm` can be instructed to parse a specific configuration file by using the `-s` switch. It is a good idea to run the check on a new configuration file before putting it into production.

```
$ testparm -s /usr/local/samba/lib/smb.conf.new
Load smb config files from /usr/local/samba/lib/smb.conf.new
Processing section "[public]"
Loaded services file OK.
# Global parameters
[global]
        coding system =
        client code page = 850
        code page directory = /usr/local/samba/lib/codepages
<...remaining output deleted...>
```

After reviewing the specified configuration file, `testparm` continues to print a version of `smb.conf` that contains all parameter values, including default ones. This can help to verify that `smbd` and `nmbd` are using the values that you expect.



Because default values can change between releases, it is important to use the version of `testparm` that matches the `smbd` and `nmbd` daemons installed on your server.

Level 2: Local Server and Client Software

The second level of Figure 13.1 turns to the server and client software. Our goal is to make sure that both machines are running and responding to NetBIOS and CIFS requests. For the most part, we are concerned with an isolated host. It is not until the third layer that the server and client will begin to carry on a conversation.

smbd

For this test, `smbd` must be running, so first we check whether this is the case by using the `ps` command. The actual `ps` arguments on your system may be different than in this example from a Linux server.

```
$ ps -ef | grep smbd
root     28592     1  0 12:37 ?  00:00:00 /usr/local/samba/bin/smbd -D
```



Having a running `nmbd` is not a requirement of the tests in this section.

After verifying that `smbd` is running, or after launching it, if necessary, our next test is to enumerate the list of shares and browse masters known by the server. In Hour 4, `smbclient`'s capability to browse a server was first introduced as a means of testing our new Samba server. The `-L <server>` option instructs `smbclient` to list the available resources on the server. The `-N` flag (anonymous login) is used to temporarily avoid any potential problems with authentication. This step should be executed while logged on to the Samba server locally (that is, not from another Unix host on the network).

```
$ smbclient -L pogo -N
added interface ip=192.168.1.75 bcast=192.168.1.255 nmask=255.255.255.0
Anonymous login successful
Domain=[STY-SAMBA] OS=[Unix] Server=[Samba 2.2.2]

        Sharename      Type      Comment
        ---------      ----      -------
        public         Disk
        IPC$           IPC       IPC Service (Samba 2.2.2)
```

```
ADMIN$          Disk      IPC Service (Samba 2.2.2)

Server               Comment
---------            -------
POGO                 Samba 2.2.2

Workgroup            Master
---------            -------
STY-SAMBA            POGO
```

Two common problems can result in a failure of this test. The first error,

```
error connecting to 192.168.1.75:139 (Connection refused)
Connection to <server> failed
```

is caused by `smbd` not running or not being able to bind to port 139. An inability to bind to the correct port can be caused by configuring `smbd` to start from `[x]inetd` (possibly left over from a previous Samba installation) and then attempting to launch the server as a daemon. The most common means of correcting this is to ensure that `smbd` can actually start. Because `smbd` does not print error messages to the console window, it is a good idea to view the last few lines of the associated log file (see Level 5 for more information on log files).

The second error message often seen by administrators is

```
session request to <server> failed (Not listening for calling name)
```

When connecting locally using `smbclient`, this error is almost always a result of a misconfigured `hosts allow` or `hosts deny` parameter in `smb.conf`. The server is running by rejecting the NetBIOS session setup. These two parameters are covered in Hour 17, "Security Tips," when we focus on Samba network security concerns.

From the description of the NetBIOS Name Service in Hour 2, it sounds like we used the wrong NetBIOS name when connecting to the server. However, this is not the case here. This error cannot be caused by a broken `nmbd` installation because `nmbd` does not even have to be running currently.

Assuming that we can successfully enumerate shares, we can next test Samba's ability to authenticate users. The details of connecting to a disk share or printer service using `smbclient` were covered thoroughly in the previous hour when we explored CIFS clients that are available for Unix hosts. In this test, we will attempt to connect to the `[public]` share as the account named "`user1`" with a password of "`secret`".

```
$ smbclient //pogo/public -U user1%secret
added interface ip=192.168.1.75 bcast=192.168.1.255 nmask=255.255.255.0
Domain=[STY-SAMBA] OS=[Unix] Server=[Samba 2.2.2]
smb: \>
```

If smbd is able to successfully authenticate the login name/password pair and that user is authorized to access the requested share, we are greeted with an smb: \> prompt.

If Samba is not able to validate the user's credentials, we are informed that

```
session setup failed: ERRSRV - ERRbadpw (Bad password - name/password
pair in a Tree Connect or Session Setup are invalid.)
```

There can be many reasons for this, such as a misspelled username or password, a missing smbpasswd entry for the user in the case of encrypt passwords = yes, or an invalid guest account if we are allowing non-authenticated access.

If the user was correctly authenticated, but could not access the request service, smbclient reports that the

```
tree connect failed: ERRDOS - ERRnosuchshare (You specified an
invalid share name)
```

This can be caused by a misspelled share name, permissions on the share which restrict the user in question from accessing that directory or printer, or a bad path statement in the share's definition as given by smb.conf.

nmbd

To test nmbd, we again use the ps command to ensure that it is running.

```
$ ps -ef | grep nmbd
root     29054     1  0 15:53 ?  00:00:00 /usr/local/samba/bin/bin/nmbd -D
```

If nmbd does not appear in the list reported by ps, it should be started as root using the normal means (/usr/local/samba/bin/nmbd -D).

Samba's nmblookup tool was briefly mentioned in Hour 4. Samba servers have a special NetBIOS name, __SAMBA__, to which they will always respond. By querying the server for this name, we can verify that nmbd is working correctly. The -U flag is used to specify the address to which the query should be sent.

```
$ ./nmblookup -U 127.0.0.1 __SAMBA__
querying __SAMBA__ on 127.0.0.1
192.168.1.75 __SAMBA__<00>
```

If nmbd had not been running, the query would have resulted in the following error message.

```
name_query failed to find name __SAMBA__
```

This can also be caused by not including the loopback interface name for the interfaces parameter in smb.conf and setting bind interfaces only = yes. Both of these parameters are discussed during Hour 23 in the context of capacity planning and system tuning.

Next we will check to see if nmbd was able to successfully register the name POGO.

```
$ nmblookup -U 127.0.0.1 POGO
querying POGO on 127.0.0.1
192.168.1.75 POGO<00>
```

Any error messages, such as "name query failed", are most likely caused by the same conditions that would cause the query for the __SAMBA__ name to fail. Another possible reason for a failure is that the server was unable to register its NetBIOS name. If this is the case, you can locate the host that currently owns the name by sending a name query request to the broadcast address of the local subnet.

```
$ nmblookup -B 192.168.1.255 POGO
querying POGO on 192.168.1.255
192.168.1.98 POGO<00>
```

In this example, the name POGO has been registered by a host at address 192.168.1.98 and not our Samba server. Obviously, the way to correct this problem is to rename the rogue machine.

Windows' NetBIOS Interface

The NetBIOS interface on the Windows machine must also be verified, just as we checked the Samba installation on the server. Because the CIFS client and NetBIOS interface are intimately linked together on Microsoft clients, it is enough to check that the latter component is functioning correctly.

Windows' NetBIOS name query tool, nbtstat.exe, contains a few extra features beyond those available to nmblookup. One of these (-n) is the ability to ask the NetBIOS interface what names it has successfully registered.

```
C:\WINDOWS> nbtstat -n

Node IpAddress: [192.168.1.135] Scope Id: []
            NetBIOS Local Name Table

   Name               Type         Status
---------------------------------------------
WIN-CLIENT     <00>  UNIQUE      Registered
STY-SAMBA      <00>  GROUP       Registered
WIN-CLIENT     <03>  UNIQUE      Registered
```

If the "Client for Microsoft Networks" has not been installed, nbtstat.exe will report (assuming that the tool is installed on the system at all)

```
Failed to access NBT driver 1
```

A more subtle error is when the Windows client reports that it has registered a workgroup name, but not its unique workstation name as shown here.

```
Name               Type         Status
---------------------------------------------
STY-SAMBA      <00>  GROUP       Registered
```

This is often caused by a machine existing on the network with a duplicate NetBIOS name. The Windows client needs a unique name to use when establishing a NetBIOS Session with a server. Until the client is able to successfully register a workstation name, it will be unable to do things such as browse the Network Neighborhood or map a network drive.

Level 3: Remote Access to Shares

So far, we have ensured that both machines are accessible on the network and that the NetBIOS and CIFS software components are working locally. At this level of the pyramid, we move beyond the local server and client to test how well the two can communicate with each other.

Name Resolution

We will again turn to the `nmblookup` and `nbtstat.exe` programs for help in verifying that the server can resolve the name of the client and vice versa. This test will come in two stages. The first part will issue a broadcast name resolution request to test the responses of the server and the client. This is done by passing the network's broadcast address to `nmblookup` (`-B 192.168.1.255`) when querying for a name. This brings network communication into play. We will first attempt to resolve the server's name by running

```
$ nmblookup -B 192.168.1.255 pogo
querying pogo on 192.168.1.255
192.168.1.75 pogo<00>
```

Next we will query for the client's name using the same broadcast address.

```
$ nmblookup -B 192.168.1.255 win-client
querying win-client on 192.168.1.255
192.168.1.135 win-client<00>
```

If all has gone well up to this point, this test should seldom fail. However, if either step does result in an error, verify that the broadcast address on each client is set to the same value. You should also check for any `interfaces` or `bind interface only` settings that would prevent the Samba host from responding to queries arriving from this particular subnet.

Next we will perform a NetBIOS Node Status Lookup request from the server to the client and from the client to the server. This step sends a directed packet to the IP address given, requesting a list of all unique and group NetBIOS names registered by the host. We will begin by querying Samba from the Windows machine.

```
C:\WINDOWS\> nbtstat -A 192.168.1.75

       NetBIOS Remote Machine Name Table

   Name               Type         Status
---------------------------------------------
POGO            <00>  UNIQUE      Registered
POGO            <03>  UNIQUE      Registered
POGO            <20>  UNIQUE      Registered
..__MSBROWSE__.<01>  GROUP       Registered
STY-SAMBA       <00>  GROUP       Registered
STY-SAMBA       <1D>  UNIQUE      Registered
STY-SAMBA       <1E>  GROUP       Registered

MAC Address = 00-00-00-00-00-00
```



The `nmbd` daemon always reports a MAC address of `00-00-00-00-00-00` to node status requests.

We can perform the same operation on the Samba server to gain information about the client. The options for performing a Node Status Request with `nmblookup` are exactly the same as those used by Windows' `nbtstat.exe` tool.

```
$ nmblookup -A 192.168.1.135
Looking up status of 192.168.1.135
        WIN-CLIENT      <00> -         B <ACTIVE>
        STY-SAMBA       <00> - <GROUP> B <ACTIVE>
        WIN-CLIENT      <03> -         B <ACTIVE>
```

If either of these requests fails, you should back up to the IP connectivity tests of Level 1 and the NetBIOS interface checks for `nmbd` and the Windows client described in Level 2.

Enumerating Shares from the Windows Client

We have already used the `smbclient` tool to enumerate the list of file and printer services on our Samba server in Level 2 of the pyramid. During this section, we will perform the same test except that the request will come from the remote Windows client.

The `net.exe` command is Microsoft's Swiss army knife for its CIFS clients. This tool provides an equivalent version of the "`smbclient -L`" command. The `view` option can be used to browse the contents of a workgroup or, when given the name of a specific server (for example, `\\POGO`), enumerate the shared resources on that host.

Unless you have enabled the special guest settings described in Hour 7, "Security Levels and Passwords," it will be necessary to make sure the Samba server can successfully authenticate you. When executing `net view \\POGO`, the Windows client will attempt to connect to the server using the login name and password of the currently logged on user. If a connection already exists, such as a mapped network drive, the `net view` command will use that session. Do not be surprised if the preceding command results in an "`Access Denied`" error message or indicates that your password is incorrect.

A successful view of `POGO` appears as

```
C:\WINDOWS\>net view \\pogo
Shared resources at \\pogo

Samba [2.2.2]

Share name   Type         Used as  Comment

-------------------------------------------------------------------------------
public       Disk
The command completed successfully.
```

Connecting to a Share Remotely

This step is often more of a goal for administrators than it is a test. Here, we will test the inter-operability of the Windows client with our server.

Before continuing, you should remember that there are several issues concerning Windows clients and clear text passwords that require special attention. These were thoroughly covered in Hours 10 and 11. We have conveniently avoided those problems here by enabling encrypted passwords in `smb.conf`.

Assuming that we are logged in to the Windows console with a valid username and password, the following command will connect the `[public]` share on `POGO` to drive `P:` on the local client.

```
C:\WINDOWS\> net use p: \\pogo\public
The command completed successfully.
```

Windows 9x/ME will not allow you to use a login name when accessing a CIFS server that is different from the one specified when logging on to the Windows console. Windows NT 4.0/2000/XP supports an optional `/user:<username>` switch for setting the

username to be used in the `SMBsessetup&X` request. The following variation, which connects to the server as `user1`, works only under Windows NT/2000/XP systems.

```
C:\WINNT\>net use \\pogo\public /user:user1
The password or user name is invalid for \\pogo\public.

Type the password for \\pogo\public:
The command completed successfully.
```

There are many more things that can break with authentication. Often, these problems can be located and solved only by digging through Samba's log files, which will be discussed later in this hour.

Level 4: Network Browsing

Network browsing (that is, the Network Neighborhood) is a rather complicated dance. So much so that we will spend two hours discussing how it works and how to integrate Samba with other CIFS clients and servers. If you have reached Level 4 of the pyramid and the Network Neighborhood is still empty or only half working, I would suggest that you look at Hours 18, 19, and 20. These hours will take you through the steps necessary to configure Samba as a good neighbor both in its local network segment and with machines located on a remote network.

Level 5: A Wealth of Information—Log Files and Packets

There are times when debugging subtle problems, which no form of diagnostic tool seems to give enough information to, tracks down the real issue. The first four levels of Figure 13.1 can be viewed as general diagnostics steps used to run a sanity check on the server when first installing or upgrading Samba. At Level 5, we have crossed over from the general diagnostics phase into the realm of hard-core Samba administration. Sooner or later, everyone hits a problem that requires cranking up the debug level, scanning the log files, and monitoring network traffic.

Samba's Log Files

Table 13.1 (originally presented as Table 5.2) describes the information recorded at each log level. The actual division of information is not this clean throughout all of Samba, so take the categories as a rule of thumb and not a firm design.

TABLE 13.1 Information Recorded at the Various Log Levels

Level	*Description*
0	Critical failures such as failing to open a log file, dropping a connection, or receiving an unknown CIFS command
1	Connection and session information
2–4	System administration debugging information
5–9	Moderate developer debugging data
10	Full developer debugging information

In Samba 2.0, the debug level of a running process could be incremented by sending it the `USR1` signal and decremented using the `USR2` signal. With the introduction of Samba's internal messaging system in 2.2, this responsibility has been handed over to the `smbcontrol` tool.

To query the current log level of an `smbd` (for example, `pid 1234`), we would send the `debuglevel` message to the process by executing the `smbcontrol` command as root.

```
root# smbcontrol 1234 debuglevel
Current debug level of PID 1234 is 0
```

If we needed to increase this to debug level 10, we would send the process a `debug` message such as

```
root# smbcontrol 1234 debug 10
root# smbcontrol 1234 debuglevel
Current debug level of PID 1234 is 10
```



It is often helpful to set `debug timestamp = no` at higher log levels. This prevents `smbd` from adding a timestamp header to each log entry, which can become distracting.

The next question is, "What do we do with all of this information that `smbd`, `nmbd`, or some other Samba tool has just recorded?"

Here is one example of how log files can help to locate the source of a problem. In this case, we are attempting to connect to a disk share from a Windows client. However, `smbd` will never accept the password that we enter for the connection. When testing the server using `smbclient` we receive the error

```
$ smbclient //pogo/public -U testuser%test
session setup failed: ERRSRV - ERRbadpw (Bad password - name/password
pair in a Tree Connect or Session Setup are invalid.)
```

We are sure that the `testuser` account has a valid entry in `smbpasswd` and that the password is set to the string "`test`". After making another attempt to connect the share with

```
log level = 10
log file = /usr/local/samba/var/log.%m
```

added to the `[global]` section of `smb.conf`, we notice the following lines in the `log.pogo`:

```
pdb_getsampwnam: search by name: testuser
startsmbfilepwent_internal: opening file /usr/local/samba/private/smbpasswd
getsmbfilepwent: returning passwd entry for user root, uid 0
getsmbfilepwent: returning passwd entry for user jerry, uid 786
getsmbfilepwent: returning passwd entry for user guest1, uid 782
getsmbfilepwent: returning passwd entry for user testuser, uid 791
endsmbfilepwent_internal: closed password file.
pdb_getsampwnam: found by name: testuser
build_sam_account: smbpasswd database is corrupt!  username testuser
     not in unix passwd database!
Couldn't find user 'testuser' in passdb.
```

The final line of this excerpt gives us a clue to our problem. Samba could not locate a Unix account for the `testuser` account. The reason for this is that someone has commented out its entry in the `/etc/passwd` file.

```
#testuser:x:791:100::/dev/null:/bin/false
```

After removing the hash mark (`#`) from the beginning of the account entry, we try one more time to connect using `smbclient`. This time, however, we are met with success.

```
$ smbclient //pogo/public -U testuser%test
Domain=[STY-SAMBA] OS=[Unix] Server=[Samba 2.2.2]
smb: \>
```

This is just one example of how Samba's log files can tell you where they are sick. Though the amount of data logged at high levels (for example, 10) can be intimidating, here are some key phrases to `grep` for when searching for a problem.

- `fail`
- `error`
- `unsuccessful`
- `corrupt`
- `unknown`



Unless you know the exact string you are searching for, use `grep`'s `-i` option to define the search string to be case insensitive (for example, `grep -i fail log.smbd`).

Monitoring Network Traffic

Many networks use shared media, such as Ethernet, to connect computers together. This is similar to a conference call, where all the connected participants can hear everything spoken by every other person on the line. Many conference calls have a passcode or special access number because of the sensitive nature of the conversation. This prevents uninvited people from randomly listening in. Networks work in much the same way. If anyone who wanted to view all network packets as they flew by on the wire were allowed to do so, invariably a password or credit card number would be seen.



Many large networks now use Ethernet switches in the place of hubs. A switch will allow you only to see packets sent to and from your host and any packets sent to your broadcast address.

Packet sniffers, tools that allow administrators to view network traffic as it passes by, can also be used as a diagnostic tool for determining why hosts or services cannot communicate with each other. We will quickly examine two such programs: Ethereal and Microsoft's Network Monitor. The first is freely available, while the second is commercial software. Our goal is to understand what these tools do and what type of information we can gain from them.

Ethereal

Ethereal is a GTK+ based program available for Unix and Windows clients from `http://www.ethereal.com/`. Its main window is displayed in Figure 13.3. One of the main advantages of Ethereal is its ability to decode packets into a more human-readable form.

Tethereal, a command line version of `Ethereal`, is also distributed with its GUI based sibling. To get a better idea of what `ethereal` (and `tethereal`) can tell us, we will watch the traffic between `smbclient` and our Samba server when connecting to a share.

```
$ smbclient //pogo/public -U user1%secret
```

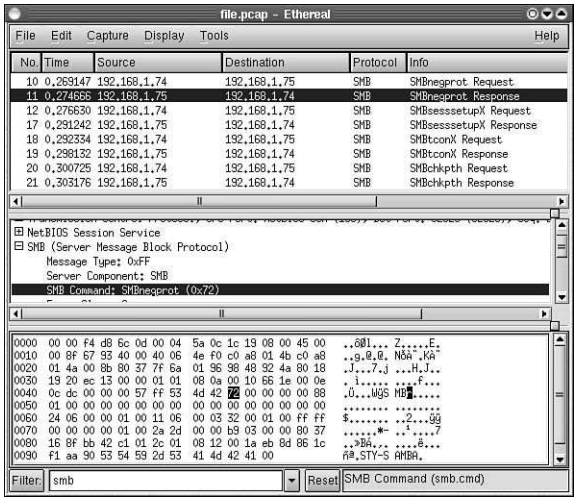


FIGURE 13.3
Ethereal's main window for displaying captured packets.

To capture the network traffic transmitted between `smbclient` and `smbd` running on the same host, we will instruct `tethereal` to watch the loopback network interface (`-i lo`) for all packets destined for `port 139`.

```
root# tethereal -i lo port 139
  0.268257        pogo -> pogo        SMB SMBnegprot Request
  0.269917        pogo -> pogo        SMB SMBnegprot Response
  0.272619        pogo -> pogo        SMB SMBsesssetupX Request
  0.280524        pogo -> pogo        SMB SMBsesssetupX Response
  0.281527        pogo -> pogo        SMB SMBtconX Request
  0.283225        pogo -> pogo        SMB SMBtconX Response
```

These packets illustrate the steps required to establish a connection to a CIFS share initially presented in Figure 2.8. It is possible to get `tethereal` to decode the packets and print a tree based view (-V) of each one. This is identical to the view provided by Ethereal. In this example, we are examining a portion of the Negotiate Protocol (`SMBnegprot`) response sent from `smbd` to `smbclient`.

```
root# tethereal -i lo -V port 139
<...output deleted...>
SMB (Server Message Block Protocol)
    Message Type: 0xFF
    Server Component: SMB
    SMB Command: SMBnegprot (0x72)
    Error Class: Success
    Reserved: 0
    Error Code: No Error
```

```
<...output deleted...>
Security Mode: 0x03
    .... ...1 = Security  = User
    .... ..1. = Passwords = Encrypted
    .... .0.. = Security signatures not enabled
    .... 0... = Security signatures not required
<...output deleted...>
```

Can you guess what the values of the `security` and `encrypt passwords` parameters are based on in this listing? The "`Security Mode`" flag is an 8-bit field used to represent this information. The least significant bit (`0x01`) is set when the server is operating in user-mode security (that is, `security = [user|server|domain]`). The next bit (`0x02`) is 1 when the server supports NTLM (that is, `encrypt passwords = yes`).

Microsoft's Network Monitor

Microsoft packages a packet-tracing tool with the Windows NT/2000 Server CD-ROM and with the System Management Software (SMS) CD-ROM called Network Monitor (also known as `netmon`). `netmon` is composed of two parts: the tool itself and a local or remote agent to which it connects. Both must be installed for the software to function correctly. Figure 13.4 displays the installation window shown by a Windows 2000 Server.

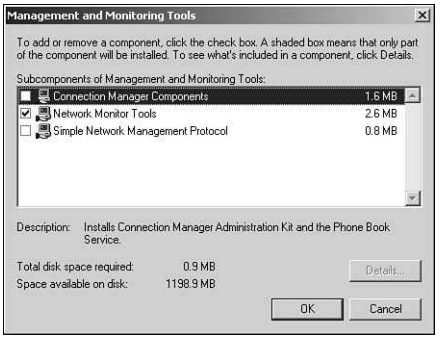
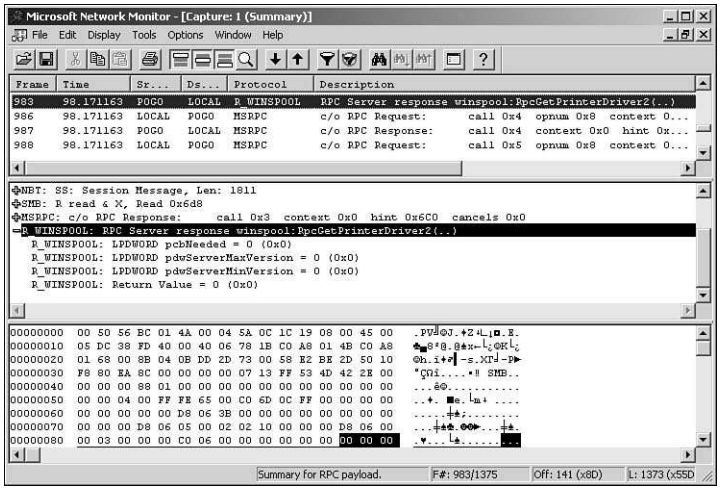


FIGURE 13.4
Installing the Network Monitor tools on Windows 2000.

There are two distinct versions of `netmon`, neither of which is freely available like Ethereal. The version included with the Windows NT/2000 Server CD-ROM allows only for viewing packets sent to and from the local machine, similar to an Ethernet switch. The version included with the SMS CD-ROM enables the network interface to be put into promiscuous mode, with which all packets on the shared media can be seen.

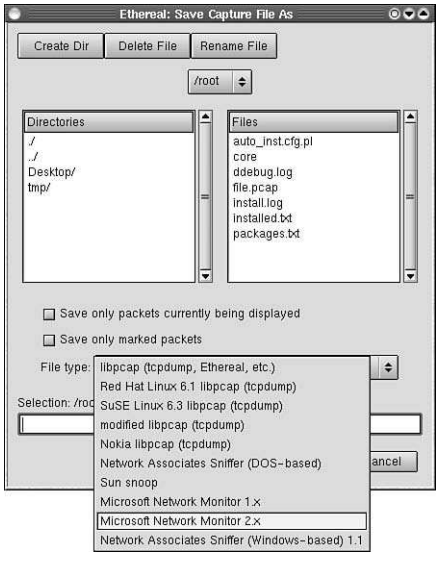
FIGURE 13.5
Decoding a printing MS-RPC response using `netmon`.



As can be seen in Figure 13.5, `netmon` has the ability to decode packets, much like Ethereal. In some cases Microsoft's tool is able to do a better job because their engineers added parsers to decode undocumented protocols such as MS-RPC.

Ethereal has the ability to read and write Network Monitor's CAP file format (see Figure 13.6). So though you may not have the full version of `netmon` available without purchasing SMS, it is possible to take advantage of its decoding capabilities by capturing the traffic using Ethereal and converting the packet trace to Network Monitor's own format.

FIGURE 13.6
Saving a network trace captured by Ethereal to `netmon`*'s CAP format.*



Level 6: Samba Internals

One of the advantages of Open Source/Free Software is the availability of the source code. Many people, however, never make use of this. During this section, we will quickly look at means of tracking down bugs in the Samba source code using a symbolic debugger. You will need to be comfortable with programming C and somewhat familiar with Unix development tools such as `gcc` and `gdb` to get the most possible out of this section. If you do not feel quite up to speed with your programming skills or have no interest in fixing bugs in the source code, this section can be skipped.

For any type of source level debugging, it is imperative that Samba be compiled to include debugging symbols. With this extra information, we can match the currently running instruction with the exact line in the source code that generated that instruction.

Enabling debugging symbols in the Samba binaries can be done by setting the `--enable-debug` flag when running the `./configure` script. This switch is compatible with all other `./configure` options, so we can use it in conjunction with Samba's support for PAM, for example.

```
root# ./configure --enable-debug  --with-pam
```

The rest of Samba's installation process remains the same.



The `--enable-debug` option sets the `-g` flag when compiling. This will normally result in much larger binaries (as much as 4 to 8 times in size), which is something to consider when installing Samba in tight places with a minimal amount of free disk space.

There are two types of bugs we will be concerned with here: those bugs that make Samba crash, and those that make Samba do the wrong thing.

Fortunately, the first class of bugs is not found very often in stable Samba releases. If Samba does encounter a fatal error such as a segmentation fault (attempting to access a region of memory that was off limits, generally due to an invalid pointer), a message similar to the following will be written to the `smbd` log file.

```
============================================================
INTERNAL ERROR: Signal 11 in pid 16124 (2.2.0)
Please read the file BUGS.txt in the distribution
============================================================
```

Before shutting down, however, `smbd` will also execute a global `panic action` command if one is defined. By default, the `panic action` parameter has no value. Because

Windows clients often reconnect transparently to the user, these crashes can easily be missed. An obvious `panic action` setting would be to immediately attach to the faulty `smbd` with `gdb` and examine where the problem occurred. The following command will launch an `xterm` on the local server's display and attach to the crashed `smbd`.

```
panic action = /usr/X11R6/bin/xterm -display :0 \
  -T Panic -n Panic -e /bin/sh -c 'gdb /usr/local/samba/bin/smbd %d'
```

Few servers run `X` in order to conserve resources. In this case, we can have the `smbd` sleep and wait for us to manually attach.

```
panic action = "/bin/sleep 99999"
```

We can then attach to the errant `smbd` at our leisure.



The `panic action` parameter is provided as an option for developers. All of the latest Samba code is available online for access via CVS (Concurrent Versions System). More information about Samba's CVS repository can be found at `http://samba.org/samba/cvs.html`. CVS clients for many platforms can be downloaded from `http://cvshome.org`.

The second class of bugs can be much harder to track down. The reason for this is that the CIFS protocol is very complex and messy. Locating exactly where the problem lies in Samba's source code can be compared to looking for a needle in a haystack. One tool that can help you to navigate through the functions and data structures is the Cygnus Source Navigator. This Integrated Development Environment (IDE) has been released under the GPL and can be downloaded from `http://sources.redhat.com/sourcenav/`. Source Navigator is an excellent tool for understanding any piece of software for which the source code is available.

Summary

Troubleshooting any problem is somewhere between an art and a science. This hour has provided a methodology for tracking down problems with Samba installations through a process of elimination. The steps have been divided up into layers, each of which depends on the successful passing of the tests of the previous level. The troubleshooting layers shown in Figure 13.1 from the bottom up are

- Level 1—General network connectivity tests and a working `smb.conf`
- Level 2—Local client and server software
- Level 3—Remote access to shares

- Level 4—Network browsing
- Level 5—Logging and monitoring network traffic
- Level 6—Source level debugging

Various mailing lists, IRC channels, and Usenet newsgroups can put you in touch with people who can help you with the art of troubleshooting. Remember to approach the problem from all possible angles, not simply focusing on the piece that is broken.

Q&A

Q Are there archives for the different Samba mailing lists that I can search to see whether anyone has ever asked my question before?

A Yes. There is a searchable archive for all the mailing lists served by `samba.org`. See `http://samba.org/samba/archives.html` for details.

Q Where can I find out information about commercial support for Samba?

A The main Samba Web site includes a page containing a list of companies that offer commercial support for Samba. Look under the "Support" page. Vendors are grouped by country.

Q What information should I include when reporting a problem or bug in Samba?

A At the very minimum, you should include

- The server's operating system and relevant version numbers (for example, kernel 2.4.9 running on a SuSE 7.2 system, Solaris 8 x86, and so forth)
- The version of Samba (or branch tag and date of the cvs snapshot) you are running
- The operating system version, including hot fixes and Service Packs, of the clients that are experiencing the problem
- A good description of the problem and the means to reproduce it, if possible

You should also be prepared to make your `smb.conf` and log files available if necessary.

Q Is there a current list of known bugs available for released versions of Samba?

A The Samba developers maintain a list of known bugs and their status at `http://samba.org/samba/buglist.html`.

New Terms

packet sniffer This is a common name for a class of network tools, either software or hardware, that are able to display the raw data being transmitted across a network. These utilities are also called network tracers or packet tracers. Some also provide the capability to parse the packets and display the information in a format more readable to humans.



PART IV
Going Production

Hour

14 Replacing a Windows NT File and Print Server—Lock, Stock & Barrel

15 Server-Side Automation

16 Managing User Accounts and Single Sign-On

17 Security Tips

18 WINS and NetBIOS Name Services

19 Local Subnet Browsing

20 Cross Subnet Browsing

21 Domain Control for Windows 95/98/ME

22 Domain Control for Windows NT 4.0/2000

23 Capacity Planning and System Tuning

24 Exploring the Samba Community and Looking Down the Road



HOUR 14

Replacing a Windows NT File and Print Server—Lock, Stock & Barrel

I have grown to hate meetings like these. I start going over my slide presentation in my head once more. If only I had a network connection under the table, or a wireless card, I could be doing something useful now, like checking my e-mail or something.

I can tell that my boss is getting ready to introduce me soon. ". . . and now with the cost analysis of replacing the server, here is our resident network expert." My boss always likes to throw in that *expert* line. I take another sip of coffee as I make my way to the head of the room to stand by the projector. I press the spacebar to bring my notebook out of sleep mode as I speak. "What we want to look at today are some figures that compare the costs of the services that we offer to our users on the network," I begin. I can hear my notebook's hard disk spinning up, and the first slide appears as if on cue. . .

". . . so the bottom line is this. By using a combination of Linux and Samba running on commodity PC hardware, we can replace the existing file server with a newer machine that is twice as fast for about half the cost. Second, we will gain a substantial cost savings in terms of client connection licensing fees. And finally, the change will be transparent to the end user." I breathe a silent sigh of relief as I sit back down, only to find my coffee cold by now.

"If this solution is as good as it sounds, why didn't we do this the first time?" one of the department heads asks.

I shrug my shoulders a little, remembering the person who installed the last batch of Windows NT servers for the company. "Times change," I explain. "Regardless of the rationale for the plan that got us to here, the solution I presented is the best one for us today and one that I believe will serve us well in to the future."

"Well done," my boss says as the two of us walk back to the office from the meeting. "I'll have Mike get out the purchase orders for the new hardware by the end of the day."

"She's always overly optimistic about those purchase orders," I smile and think to myself. "Sounds good," I reply as I turn the corner to the lab and begin walking away. I begin to go over in my mind the things I need to do to replace the Windows NT file server with a Linux box. "Now where did I put that coffee cup?" I mutter. . .

So far, we have explored many of Samba's capabilities, such as authentication, file, and print services. During this hour we will look at how these features can work together to provide a full replacement for an NT 4.0 file and print server in a Windows domain. Our case study will walk through the steps for replacing a Windows NT 4.0 member server of a Windows 2000 domain. The server will provide home directories for domain users, a file service used by various Windows NT groups as a point of collaboration, and a shared network printer. We will also cover some more advanced Samba topics, such as remote management via native Windows NT administration tools and the implementation of the Microsoft Distributed File System. It will be a very full hour indeed!

The Existing Network

Before beginning to address the configuration details of our new Samba server, we must gather the requirements that must be met by the new machine.

- The Samba server should appear in the same domain (GLASS) and use the same NetBIOS name (BEVELED) as the existing server in order to minimize confusion for the users.

- All Domain users should be able to access the new shares on the Samba server without requiring a synchronized UNIX account on the machine. This means that the existing NT domain account should provide access to the new server's shared resources.
- The access control mechanism for files should be kept the same so that a user who has access to a file on the existing server should have access to the same file on the Samba server. Also, a user who does not have access to a file on the existing server should not be able to access that file under the new configuration.

Satisfying the first requirement is easy to do by simply joining the Samba server to the domain using the same name as the current server. The second requirement will be met using a software package new to Samba 2.2.2 named Winbind, which will be covered later during this hour. In order to fulfill the third requirement, we will need to compile Samba to include support for file system ACLs. Because of Linux's proliferation in the server room today, I will cover how to implement POSIX ACLs on a Linux 2.4 system.

There are three basic services that must be moved from the existing Windows NT server onto the new Samba box. This list is small enough that we can examine each service individually.

- `[users]`—This share contains the home directories for the users in the domain. Each user account possesses a directory immediately below the root of the share. For example, user `jdoe`'s home directory would be located at `\\BEVLELED\USERS\jdoe`.
- `[docs]`—A common disk share for group collaboration. All users can create directories within the share, but once a file/directory has been created, access is controlled via the standard NTFS ACLs.
- `[hp4]`—A network printer available for use by all users in the domain.

Finally, Figure 14.1 illustrates the current setup. We will use a single domain model with a sole Primary Domain Controller (PDC) to handle all user authentications. The number of client machines is not important for our purposes here. The scenario used during the hour will use three machines: (1) the PDC to perform the authentication, (2) the file/print server, and (3) one Windows NT or Windows 2000 client machine.

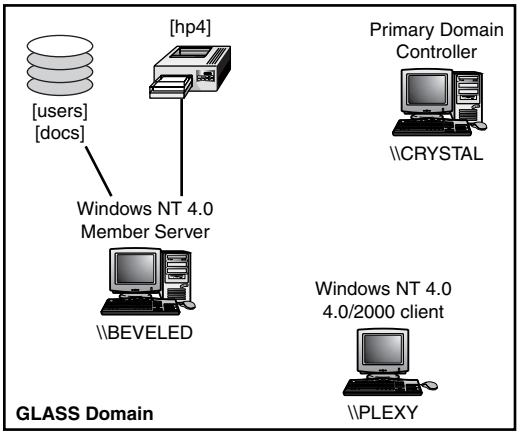


FIGURE 14.1
Network Topology of the GLASS *domain.*

Samba and ACLs

Out of concern for portability among modern Unix systems, Samba developers have chosen to implement a mapping between Windows NT ACLs and the POSIX ACLs described in the POSIX 1003.1e Draft Standard (revision 17). Although this standard was never completed, it contains enough detail to be implemented by vendors. Figure 14.2 illustrates how Samba is able to support converting NT ACLs into file system ACLs from various Unix vendors. The Windows NT ACL is converted into a POSIX ACL for internal use. This POSIX ACL is then converted into an ACL that is understood by the operating system's libraries to be written onto disk.

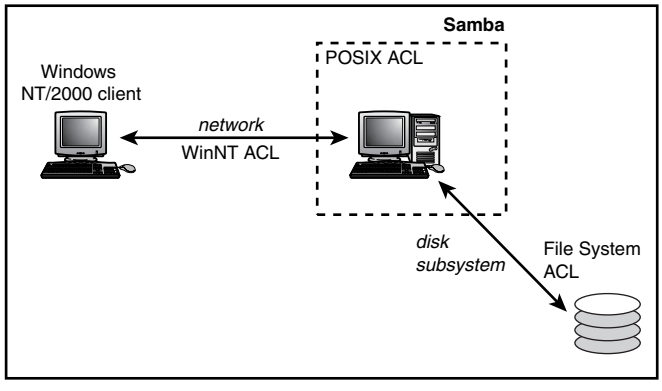


FIGURE 14.2
Converting a Windows NT ACL into a POSIX ACL and out to a specific File System ACL (for example, Solaris, Linux, AIX, and so forth).

Currently, the Linux kernel does not include support for file system access control lists. If you are using a server OS other than GNU/Linux, feel free to skip this section entirely and pick up with the heading "`--with-acl-support`". Be aware that it is very difficult to replace a Windows NT file server with a Samba server if the underlying Unix OS does not contain support for some type of file system ACL that is recognized by Samba. It is a good idea to make sure that your system is supported. The list of supported systems with ACLs in Samba 2.2.2 is

- Solaris
- True64
- AIX
- UnixWare
- SGI's XFS file system for IRIX and Linux
- The POSIX ACL patch for the Linux 2.2 and 2.4 kernels

There are two up and coming candidates vying for the title of standard Linux ACL. The first is SGI's XFS, which has been released for the Linux 2.4 kernel. This high-performance, journaling file system includes support for native POSIX ACLs, among a list of other features. The main disadvantage of XFS is that it requires the file system to be created from scratch. There is no means of converting an existing `ext2` file system to XFS transparently. More information on the Linux XFS project can be found at

`http://oss.sgi.com/projects/xfs/.`

The second possible candidate is the kernel patch written by Andreas Gruenbacher. Gruenbacher's code is affectionately known as the POSIX ACL patch by Samba developers and can be obtained from `http://acl.bestbits.at/`. This addition to the Linux kernel adds the ability to store extended attributes and access control lists on existing `ext2` file systems. For this reason, it is less intrusive to most systems than XFS and will be the ACL package of choice for our server.

Configuring the Linux Server

We will begin building our new server from the bottom up. For the sake of reference, the system used for all examples is running the Linux system with a 2.4.9 kernel created from the pristine source code and version 2.2.2 of the GNU C library.

The first step to installing ACL support on our Linux system is to obtain the necessary patches from `http://acl.bestbits.at/download.html`. Although this list is subject to

change with future kernel releases, or may even become obsolete if ACL support is accepted into the kernel, the current requirements are

- The Extended Attribute (EA) kernel patch
- The Access Control List (ACL) kernel patch
- The ACL utilities and development files
- Release 1.23 or later of the e2fsprogs package

In addition, there is an optional patch described on Gruenbacher's Web site for the GNU File Utilities, such as `cp`, `ls`, and `mv`, to display and preserve ACLs on files and directories. The main items we will be concerned with in this section are the kernel patches and the ACL tools.



When using the Linux kernel POSIX ACL patches for the Linux kernel, be aware that the `chmod` command will behave slightly differently. The user and other permission sets will be manipulated as normal. Named user permissions will not be modified by the `chmod` command, however. In addition, rather than changing the group owner or named group permission sets, `chmod` will modify the group access mask, which is described later in this chapter. This behavior is not standard on all Unix systems, so be sure to read the documentation for your system to understand how the `chmod` tool interacts with file system ACLs.

First, we must download the appropriate kernel source from `http://www.kernel.org/`. In our case, this will be `linux-2.4.9.tar.gz`. Once we have obtained the compressed tar archive, we can extract the kernel source into the `/usr/local/src` directory with



Make sure to create the `/usr/local/src` directory first if does not currently exist.

```
$ cd /usr/local/src
$ tar zxf linux-2.4.9.tar.gz
$ mv linux linux-2.4.9
```

The last command renames the top-level directory to help remind us which release we are using.



You can find a simple guide to patching and compiling the Linux kernel at the Linux Documentation Project (`http://www.linuxdoc.org`). Search for a file named Kernel-HOWTO.

Next we will uncompress the extended attribute and ACL patches and add them to the kernel source tree.

```
$ gunzip linux-2.4.9ea-0.7.16.patch.gz
$ gunzip linux-2.4.9acl-0.7.17.patch.gz
$ cd linux-2.4.9
$ patch -p1 < ../ linux-2.4.9ea-0.7.16.patch
patching file Documentation/Configure.help
patching file arch/alpha/kernel/entry.S
<...output deleted...>
$ patch -p1 < ../ linux-2.4.9acl-0.7.17.patch
patching file Documentation/Configure.help
patching file fs/Config.in
<...output deleted...>
```

Before compiling the kernel, however, the EA and ACL features must be enabled in the File Systems section of the kernel configuration file. Figure 14.3 shows the options displayed by executing `make xconfig` from the top level of the kernel source directory. This screen can be reached by selecting the "File Systems" button from the main menu.

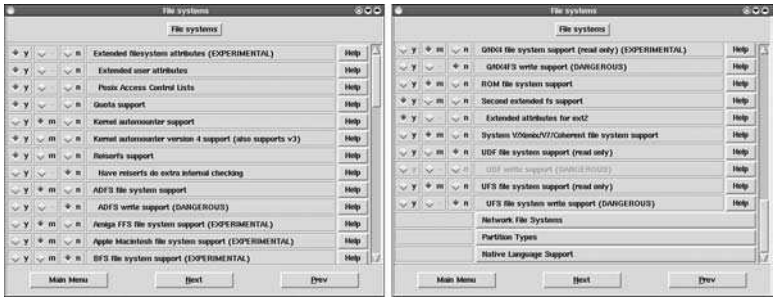


FIGURE 14.3
Enabling EA and ACL support for the `ext2` *file system in the Linux kernel.*



`make xconfig` requires that the X window system be running on the local machine (or on the remote machine if logged in to the Samba server across the network). There are other configuration tools included with the kernel source, such as `make menuconfig` and `make config`. The differences between these are described in the Kernel-HOWTO.

After saving the new kernel configuration, the kernel and associated modules can be built and installed as root, according to the directions in the Kernel-HOWTO.

Once we have a working kernel with ACL support and have booted with that kernel, it is time to build and install the ACL tools themselves. After unpacking the `acl-<version>.tar.gz` archive into the same directory where the kernel source was stored (i.e. `/usr/local/src/`),

```
$ pwd
/usr/local/src
$ tar zxf acl-0.7.16.tar.gz
```

building the tools is a simple process of running the `configure` script and executing `make`.

```
$ cd acl-0.7.16
$ env KERNEL_SOURCE=../linux-2.4.9 ./configure
$ make
$ /bin/su
Password:
root# make install
```

The "`env KERNEL_SOURCE=../linux-2.4.9`" string is used to inform the `./configure` script where to locate the kernel source tree (if it is located somewhere besides the default directory `/usr/src/linux`).

Executing a `make install` as the root user installs the ACL tools in subdirectories (that is, `bin`, `lib`, and `include`) below the default install root of `/usr/local`. We can install them here by running the following command as root.

Verifying that the kernel's ACL support is functioning correctly involves becoming familiar with two of the ACL tools, `getfacl` and `setfacl`, both of which are installed in `/usr/local/bin`. These utilities are used to print and set Access Control Entries (ACE), respectively, within the ACL.

First, remember that the POSIX ACL support applies only to `ext2` file systems, not `vfat`, `reiserfs`, or `ntfs`. It is important to make sure that the directories Samba will be sharing with users are located on an `ext2` file system.

When in doubt, an easy method of determining the file system type is to ask the kernel how the disk partition is mounted. Executing the `mount` command with no arguments displays the currently mounted disk partitions as well as the file system type and any additional parameters used. In this example, we can see that the `/export` directory is in fact the root of an `ext2` file system.

```
$ mount
/dev/sda3 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
/dev/sda1 on /boot type ext2 (rw,check=none)
devpts on /dev/pts type devpts (rw)
/dev/sdb2 on /export type ext2 (rw,check=none)
```

Next we will create a directory used for testing purposes named `tmp`. This assumes that we have `write` permission to the `/export` directory.

```
$ mkdir /export/tmp
```

We can use the `getfacl` tool to view the initial ACL placed on this directory.

```
$ getfacl /export/tmp
# file: /export/tmp
# owner: gcarter
# group: users
user::rwx
group::r-x
other::r-x
```

We can add an entry for an account named `guest1` to this ACL by running the `setfacl` command shown here.

```
$ setfacl -m user:guest1:r-x
```

The resulting ACL is

```
# file: /export/tmp
# owner: gcarter
# group: users
user::rwx
user:guest1:r-x
group::r-x
mask::r-x
other::r-x
```

The full syntax of the `getfacl` and `setfacl` commands is described in the respective man pages for the tools. Our purpose here is not to learn about manipulating ACLs using these tools. Rather, our goal is to ensure that the ACL support in the kernel is working correctly so that Windows users can set and view ACLs on Samba file shares. With this in mind, there are a few points of interest to remember about POSIX ACLs.

- They do not add any permission bits. Each entry contains only read (`r`), write (`w`), and execute (`x`).

- The owner of a file/directory is represented by the `user:: <permission>` entry. The group owner is represented by the `group::<permission>` entry. The world-permission set is listed as the `other:: <permission>` entry. The two semicolons are significant here, because "named" users and groups are represented by either `user:<name>:<permission>` or `group:<name>:<permission>`.
- Access to an object is checked first by searching for an explicit entry for that user. If one is found, then the access granted by that entry is used and the check is completed. If no explicit user entry is located, search for all groups of which the user is a member. The resulting permission, in this case, is the bitwise `OR` of the permissions of each matched group. If the user is not a member of any group listed in the ACL, then the `other::` entry is used.

  Assume that a file had the following ACL:

```
# file: tmp
# owner: gcarter
# group: users
user::rwx
user:guest1:r-x
group::r-x
group:pppusers:-wx
group:guest:r-x
mask::rwx
other::r-x
```

  The access granted to `guest1` would be `r-x` because it is explicitly stated. The access granted to a user who belonged to both the `guest` and `pppusers` group would be `rwx`.
- An ACL mask is applied as bitwise `AND` with each group entry (name and owner). The result ensures that groups are given the most restrictive set. For example, an entry of `group:engr:rwx` and a `mask::r-x` would result in only `r-x` access being granted to members of that group.
- Directories can have "default" entries, which assume the role of a `umask`. When a new file or directory is created with the parent directory possessing the default ACEs, these entries are inherited onto the requested permission set.
- The `security mask` and `force security mode` group of parameters presented in Hour 8 are only applied to the owner, group, and world permission sets. These parameters have no effect on named users or named groups that may exist in a POSIX ACL.

There is much more to POSIX ACLs than the information presented in this section. The `acl(5)`, `setfacl(1)`, and `getfacl(1)` man pages contain a very good explanation of the interaction of the various components. These documents should be considered required reading for any administrator maintaining a Linux/Samba server with POSIX ACLs.

We will now turn our attention to how Windows users can view and manipulate these ACLs via the Windows Explorer.

`--with-acl-support`

In order for Samba to be able to manipulate anything beyond the standard set of Unix permissions (`user/group/other`), the `./configure` script must be run to include the `--with-acl-support` option.

```
$ ./configure --with-acl-support
<...output deleted...>
checking whether to support ACLs... checking for acl_get_file in -lacl... yes
checking for ACL support... yes
Using posix ACLs
<...output deleted...>
```

If there are any errors at this stage, verify that the header files and library have been installed from the ACL tools package and can be located by Samba. It may be helpful to view the last couple of hundred lines of the `config.log` file to determine the exact cause of failure.

Assuming that the configure script completes successfully, the Samba binaries can be built and installed using the same methods described in previous hours (that is, `make && make install`).

The Replacement Process

To replace the Windows NT server with a Linux/Samba box, we will go through three steps:

1. Join the Samba server to the Windows domain and configure all user/group account mappings between the Unix system and the Windows domain.
2. Create the new shares in Samba's `smb.conf` file to replace the existing ones on the Windows NT server, and move the files and print spools from the Windows NT Server to the Linux/Samba box.
3. Test the new server.

Step 1: Joining the Domain. . .

In times past it was necessary to manually create a user or group entry for each domain account beforehand, so that `smbd` could obtain a `uid` (or `gid`) for the Windows domain user (or group). The need for mapping a Windows domain account onto a Unix `uid`/`gid` has not been removed. However, with the first stable release of the Winbind package in Samba 2.2.2, management of this mapping has become much easier.

Winbind is a name for a collection of three pieces of software.

- The `winbindd` daemon handles mapping Windows NT SIDs to Unix `uids`/`gids` and vice versa. A SID is the Windows NT version of a `uid` or `gid`, except that it is guaranteed to be unique across multiple domains. If you are familar with NIS/YP on Unix servers, consider an SID to be like prefixing a `uid` with an NIS domain name. A Windows NT SID can be broken up in the machine/domain SID and the Relative Identifier (RID) that composes the last 32 bits of the SID.
- The `libnss_winbind.so` Name Service Switch (NSS) library.
- The `pam_winbind.so` Pluggable Authentication Module (PAM) library.

The three of these combined can provide a unified solution for transparently authenticating Unix users against a Windows Domain Controller. In the case of `smbd`, only the first two items are of critical importance. The PAM module will be covered more in Hour 16, "Managing User Accounts and Single Sign-on."

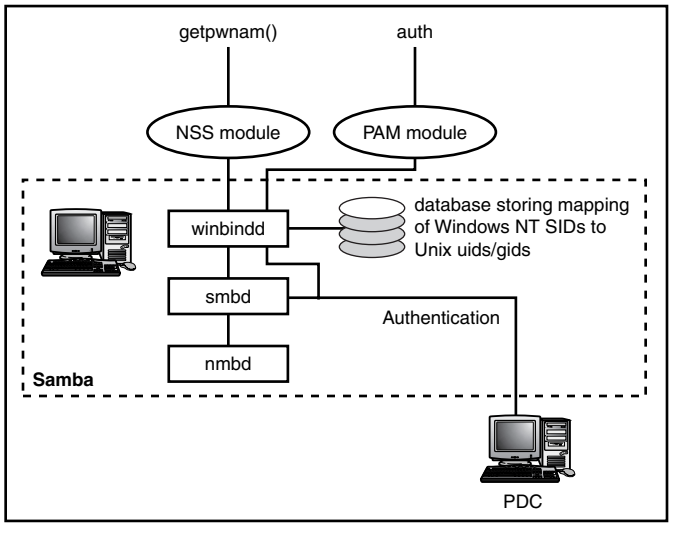


FIGURE 14.4
The Architecture of Winbind and its interaction with `smbd`.

Figure 14.4 illustrates how the pieces fit together. In the actual implementation, the division between modules is not quite as clean as I have presented it here, but the main points should be clear. This Samba server is composed of `winbindd`, `smbd`, and `nmbd`. The purpose of `winbindd` is to maintain the database of SID to `uid`/`gid` mappings so that normal Unix programs, including Samba, can resolve a user name by going through the Winbind NSS module. Technically, `smbd` does not rely on the C library's `getpwnam()` call to resolve a domain user to a `uid`. Rather, it calls an internal lookup routine that contacts the `winbindd` daemon directly. In both cases, the end result is the same—a Windows NT domain user or group is converted to a Unix `uid` or `gid`.

Remember that although it is the PDC that performs the authentication, determining authorization on a Unix system is done by checking the user's `uid` and list of `gids`. How can the kernel know if you should be able to write to a file unless it can check for your `uid` in the object's permission set? Ultimately, winbind lowers the management cost for this requirement, handling the `uids` and `gids` for you.

The `winbindd` server does not perform any authentication per say. It does contain a few hooks used to support the PAM module, which can authenticate users of any PAM-aware application, such as `OpenSSH` (`http://www.openssh.org`) or a `proFTPD` server (`http://www.proftpd.net`).

All the pieces of winbind are built by default in Samba 2.2.2 with the exception of the PAM module, which will be discussed more in Hour 16. Winbind was developed for Linux systems but should also work without much difficulty on Solaris. Its state on other operating systems is somewhat of an unknown at this time, due to testing requirements. The `winbindd` daemon is installed in `/usr/local/samba/bin`, and the NSS module should be copied to `/lib`.



It is possible to not build the Winbind components by specifying `--without-winbind` when running the `configure` script.

Because winbind is intimately tied with Samba's `security = domain`, our first step in configuring it is to join our Samba server to the Windows domain. Beginning with an empty `smb.conf` file, we will add the following parameters:

```
## new smb.conf for Samba member server of a
## Windows domain
##
## Sam's Teach Yourself Samba in 24 Hours
## Gerald Carter <jerry@samba.org>
```

```
[global]
     netbios name      = BEVELED
     workgroup         = GLASS
     security          = domain
     encrypt passwords = yes
     password server   = *
```

We have not covered WINS and non-broadcast NetBIOS name resolution yet. In some circumstances, you may find it necessary to also define a `wins` server that should be queried for name registration and resolution requests. This will be covered in detail in Hour 18, "WINS and NetBIOS Name Services." For now, we will assume that the Windows NT server we are replacing, all Windows clients, and the PDC for the domain are all located on a single network and can reach each other with broadcast packets.

The `netbios name` and `workgroup` parameter values have been previously defined by the existing Windows NT server. The `security = domain` and `encrypt passwords = yes` settings are a prerequisite for acting as a member server.

The `password server` parameter is used to inform `smbd` of the names of domain controllers which can be contacted to authenticate the user. By setting this to the wildcard character (`*`), `smbd` will attempt to locate a DC on its own, using the same name resolution methods as a normal Windows NT member server (that is, resolve the `GLASS<1c>` NetBIOS name to a list of IP addresses). This could have been set to a list of names such as

```
password server = PDC BDC1 BDC2
```

In this case, `smbd` would attempt to contact the first server in the list and would continue only to the next server if the previous one was unavailable.

We are now ready to join the Samba host to the `GLASS` domain. The `smbpasswd` utility can wear many hats, from managing user accounts to changing passwords on remote CIFS servers. We can now use it to create a computer account in the domain for our server. To do this, we must first remove the existing machine trust account from the domain as shown in Figure 14.5.

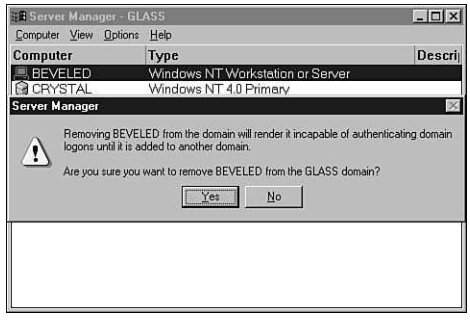


FIGURE 14.5
Removing the existing account for `BEVELED` *from the* `GLASS` *domain.*



The requirement of removing the machine account from the domain prior to joining the domain with `smbpasswd` is considered by many to be a bug. Expect this requirement to be removed in future versions of Samba.

With no account for `BEVELED` in the domain, we can rejoin the domain using the `smbpasswd` tool by specifying the domain to join with the `-j` flag. The `-r` option accepts the same arguments as the `password server` parameter. The `-U` switch should be followed by the `username%password` combination for an account that has the appropriate level of privilege to add a machine to the domain (for example, a member of the "`Domain Admins`" group). This command must be executed as root because `smbpasswd` will store the new machine trust account password in the `secrets.tdb` file, which is located in the `private/` directory and is writable only by root.

```
root# smbpasswd -j GLASS -r '*' -UAdministrator%secret
Joined domain GLASS
```



Samba 2.0 required that the computer account for the Samba server being created beforehand by running the Windows NT Server Manager on the PDC. This method, though still supported, is not recommended for use in Samba 2.2.2 and later because it leaves a window of opportunity for a malicious user to join any machine to the domain simply by using the name of the new computer account.

We are almost ready to launch the Samba daemons. Prior to this, we must add some required parameters for the `winbindd` daemon. This server process was previously described as handling the mapping between Windows NT SIDs and Unix `uids/gids`. In order for this to occur, we must allocate a set of `uids` and `gids` for which `winbindd` will be authoritative. This is done using the `winbind uid` and `winbind gid` global `smb.conf` configuration settings.

```
winbind uid = 1000-50000
winbind gid = 1000-50000
```

These lines set aside a range of uids and gids that `winbindd` can draw from as necessary. Whenever `winbindd` encounters an SID that has not previously been mapping to a Unix id, it will grab the next available uid or gid as necessary and store the mapping in a local database.

Given the finite range of numbers in a Windows NT RID, in theory it is possible to overflow the available uids and gids with domain and multiple trust relationships. However, in practice this does not occur because it would take an extremely large and complex domain with a large number of users all accessing this same Samba server. A good rule of thumb is to beware of any of the following circumstances:

- More user accounts in the Windows domain that available ids in the `winbind uid` range. The same is true for groups although it is rare to see the number of groups in a domain out number user accounts.
- An extremely high number of trust relationships with the Windows domain of which the Samba server is a member. In this case, every user in the trusted domain has the potential to access the Samba server. The number of user accounts spread across all trusted domains can easily exhaust the pool of available uids or gids.
- The frequent addition and deletion of user and/or group accounts from the Windows domain. This can exhaust a Unix uid/gid range because once an id is allocated by winbind, it is not used again. To do so would introduce a security problem that would make it potentially possible for one user to gain access to files that he or she did not own. This is the same reason why Windows NT/2000 does not reuse a RID when a user account or group is deleted.

This does not mean that Winbind will not function properly in these environments. It simply means that things should be monitored closely to ensure that Winbind is not overwhelmed.

There are also two additional parameters that administrators often include, whether or not they are needed by their site. These are the `template homedir` and `template shell` settings. Their purpose is to fill in information needed by `getpwnam()` calls, which return an `/etc/passwd` style structure. The default values for each parameter are

```
template homedir = /home/%D/%U
template shell = /bin/false
```

Of course, these can be replaced with whatever value is more appropriate for your server. The `%D smb.conf` variable is expanded to the domain as gleaned from the account `DOMAIN\user`.

After adding the winbind-related parameters, we are finally ready to test our winbind and current Samba configurations. The first step is to verify that the `libnss_winbind.so` library is installed in `/lib` and that the file permissions are set to `755`. There should also exist a soft link from `libnss_winbind.so.2` to `libnss_winbind.so`.

```
root# ls -l /lib/libnss_winbind*
-rwxr-xr-x    1 root  root   57103 Sep 14 22:26 /lib/libnss_winbind.so
lrwxrwxrwx    1 root  root      22 Sep 18 00:41 /lib/libnss_winbind.so.2 ->
/lib/libnss_winbind.so
```

Next, the NSS configuration file (`/etc/nsswitch.conf`) must be edited to add the `winbind` lookup module to the password and group databases. A portion of the file is shown here. Notice that there is no need to add the `winbind` keyword to the `shadow` map.

```
passwd: files winbind
shadow: files
group:  files winbind
```



See the `nsswitch.conf`(5) man page for more information on Name Service Switch.

After launching the `winbindd` daemon as root,

```
root# /usr/local/samba/bin/winbindd
```

it should now be possible to enumerate all Unix and Windows domain user accounts by executing

```
$ getent passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
<...deleted...>
testuser:x:791:100::/dev/null:/bin/false
kristi:x:792:100::/home/pogo/kristi:/bin/bash
GLASS\Administrator:x:1000:1000::/home/GLASS/administrator:/bin/false
GLASS\foo:x:1001:1000::/home/GLASS/foo:/bin/false
GLASS\foo2:x:1002:1000::/home/GLASS/foo2:/bin/false
GLASS\foo3:x:1003:1000::/home/GLASS/foo3:/bin/false
GLASS\Guest:x:1004:1000::/home/GLASS/guest:/bin/false
GLASS\user1:x:1005:1000::/home/GLASS/user1:/bin/false
```

In the same way, we can enumerate all groups on the Samba server by passing the group argument to the `getent` command.

```
$ getent group
root:x:0:root
bin:x:1:root,bin,daemon
<...deleted...>
```

```
guest:x:1000:
sapdb:x:61:sapdb
GLASS\Domain Admins:x:1001:GLASS\Administrator
GLASS\Domain Guests:x:1002:GLASS\Guest
GLASS\Domain Users:x:1000:GLASS\Administrator,GLASS\foo2,GLASS\user1,GLASS\foo3
GLASS\testgroup:x:1003:GLASS\foo
GLASS\testgroup2:x:1004:GLASS\foo
```

To convince ourselves some more, we can experiment by using domain accounts in conjunction with the standard Unix tools, such as `chown`.

```
root# touch /tmp/afile
root# chown 'GLASS\user1' /tmp/afile
root# ls -l /tmp/afile
-rw-r--r--    1 GLASS\us root            0 Sep 18 09:09 /tmp/afile
```

The Linux `ls` command will display only up to eight characters of the user's name, which is why we see only "`GLASS\us`" listed.

Because the backslash is traditionally used as an escape character on Unix systems, entering a name user as `DOMAIN\user` requires that the string be surrounded in single quotes to prevent the shell from viewing '`\`' as a special character. If this is too much of an annoyance, the separator character used to display domain accounts and groups can be changed by setting the `winbind separator` global parameter. A common choice is to define `winbind separator = +`.

Once we are convinced that the Winbind components are functioning correctly, we can finish configuring the file and printer shares for `smbd`.

Step 2: Creating the New Shares and Transferring the Data

We have already performed most of the work necessary to copy the files containing the `USERS` and `DOCS` file share from the old Windows NT server to the new Samba server. However, we have yet to define these services in `smb.conf`. It is now time to define these two share as follows:

```
## disk share for storing user home directories
[users]
   path = /export/smb/users
   read only = no
   create mask = 0600
   directory mask = 0700
```

```
## public disk share used by all users
[docs]
   path = /export/smb/docs
   read only = no
   create mask = 0660
   directory mask = 0770
```

The `create` and `directory masks` were chosen because of the expected use of the share. Of course, the default ACLs on directories will still be inherited, as previously explained. There is a subtle issue with the permissions and ownership of the top-level directory of each share that begs further explanation.

It is assumed that all "`Domain Admins`" should be able to create directories in the top level of the `[users]` share. However, normal users should be restricted to creating objects only in their own respective folders. To enforce this, we will give the `Domain Admins` group ownership and write access to the directory `/export/smb/users`.

```
root# chgrp 'GLASS\Domain Admins' /export/smb/users
root# chmod 775 /export/smb/users
```

The `[docs]` service has a different requirement however. Any user or group should be able to create folders and set permissions as necessary in this share. Although we could make the share group writable by the "`Domain Users`" group, it is just as easy to make it world writable to ensure that any user will be able to write to the share. The sticky bit (the '1' in '1777') is used to prevent a user from removing a file that he does not own even though standard Unix semantics would allow this if the user possessed write access to the parent directory containing the file.

```
root# chmod 1777 /export/smb/docs
```

The `create` and `directory masks` ensure that all files/directories will initially be accessible to members of the same group. This can be manually changed by a user if desired.

When transferring the existing data from these two services on the Windows NT server to the new Samba file shares, we must be very careful to preserve the existing ACLs. The easiest way to do this is to use a tool such as the `scopy.exe` utility included with the Windows NT 4.0 Resource Kit.

Before continuing, we should change the name of the old Windows NT server and rejoin it to the domain. Any name is fine (for example, `FOO`), as its domain membership will be short lived. Once the server has been rejoined to the domain, we can log in using a Domain Administrator account (for example, `GLASS\Administrator`).

Here we are copying everything in the local `c:\users` directory to `\\BEVELED\USERS`.

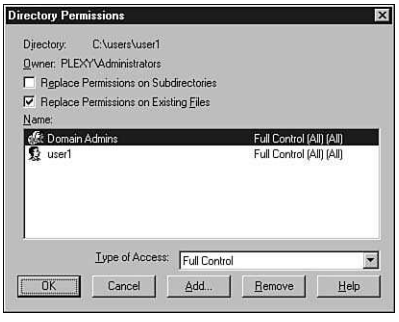
```
C:\users>scopy *.* \\beveled\users /s
C:\users\foo2 -> \\beveled\users\foo2
C:\users\foo2\AUTOEXEC.BAT -> \\beveled\users\foo2\AUTOEXEC.BAT
C:\users\foo2\testing -> \\beveled\users\foo2\testing
C:\users\user1 -> \\beveled\users\user1
C:\users\user1\09fig09a.bmp -> \\beveled\users\user1\09fig09a.bmp
File(s) copied: 5
```



In Supplement 4 of the Windows NT 4.0 Server Resource Kit, this tool was supposedly replaced by the standard `xcopy.exe` command, which is included as part of the Windows NT 4.0 operating system. Personally, I have had trouble getting `xcopy.exe` to copy secure information on files/directories.

Figure 14.6 shows the permissions on the home directory for `user1` as displayed by the Security Tab from the Windows NT Explorer.

FIGURE 14.6
Original permissions on `C:\USERS\user1` *as shown by the Windows NT 4.0 Explorer.*



After copying the directory to the Samba server that uses POSIX ACLs, the Windows NT ACL is converted to

```
# file: user1
# owner: GLASS\Administrator
# group: GLASS\Domain Users
user::rwx
user:GLASS\user1:rwx
group::---
group:GLASS\Domain Admins:rwx
mask::rwx
other::---
default:user::rwx
default:user:GLASS\user1:rwx
```

```
default:group::---
default:group:GLASS\Domain Admins:rwx
default:mask::rwx
default:other::---
```

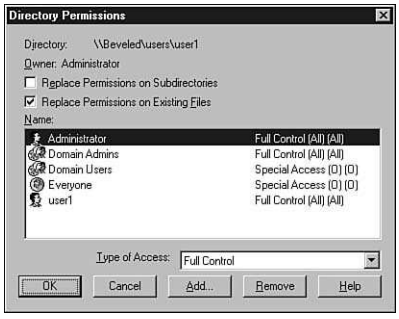


FIGURE 14.7
Permissions on the new `\\BEVELED\USERS\user1` *stored on a Linux/Samba server using POSIX ACLs.*

Figure 14.7 shows how this ACL appears in the Windows NT security dialog. This brings up a couple of points that can be confusing.

- Just as with non-ACL aware file systems, Samba reports no permissions (that is, `---`) using the Take Ownership (`0`) bit because it looks like 0 bits set. This is simply a convention for reporting the permissions and does not give the specified user or group any additional access.
- The `rwx` permission set is reported as being equivalent to Full Control (ALL) access.

A Quick Note About Printers

After copying all the file data to the Samba server, it is time to turn to the details of moving the `[hp4]` printer from the old NT server to our Samba host. Unfortunately, there are no simple means to move a printer queue from one Windows NT server to another. Consequently, this means that there is also no easy way to move a print queue from a Windows NT host to a Samba server.

Our only option is to recreate the printer on the Samba box and manually reset the permissions. This also means that we will need to manually install the necessary drivers as well. All of this was presented in Hour 9, "Samba—The Print Server," so I will not cover it again here.

Step 3: Testing the Server

We have already verified parts of the server's new configuration. However, as any seasoned network administrator will tell you, there is a big difference between performing tests and moving to a production environment. Is it any wonder that so many IT professionals have short fingernails? It is impossible to describe all the possible things a user

might do while logged in to the network, but we will make sure that a few important things work for a normal user.

- Logging in to the domain, a user should continue to mount his or her home directory without any intervention.
- The new server should appear in the Network Neighborhood.
- A user should be able to open a document from the `[docs]` share and print it to the `[hp4]` printer.

The logon script window, shown in Figure 14.8, is displayed after you log in as `user1` and mounts the `[docs]` service to drive `S`—so far, so good.



FIGURE 14.8
Logon script run when logging on to the `GLASS` *domain as* `user1`.

Next we browse the Network Neighborhood to make sure that the new `BEVELED` server appears as it should. The result from our successful test is shown in Figure 14.9.

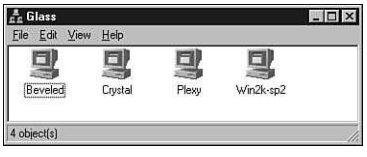


FIGURE 14.9
Our New Samba server displayed in the Network Neighborhood.

And finally we attempt to open a text file from `\\BEVELED\DOCS (S:)` in `wordpad.exe` and print it (see Figure 14.10).

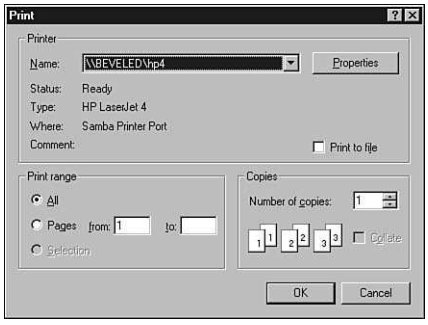


FIGURE 14.10
Printing a text file stored in `\\BEVELED\DOCS` *to* `\\BEVELED\HP4`.

Managing Shares on a Samba Member Server

In the case where a Samba server is deployed in an environment made up mostly of Windows NT clients, servers, and administrators, traditional Unix-centric means of managing the Samba host, such as a text editor or SWAT, may be viewed with some anxiety due to the differences between Unix tools and the standard Windows NT Administration tools. These methods, loved by Unix admins, are more flexible for Samba's means, but it is also necessary to support the native NT utilities, such as Server Manager or the Microsoft Management Console (MMC), in order to blend in with other servers.

As a member server, there is no real need for user management tools because the majority of user accounts of concern are maintained by the domain controllers. In contrast to this, tools that allow an NT admin to create, modify, and delete file shares can be used on a daily basis. We have already seen some precedence for this with the Windows NT Add Printer Wizard covered in Hour 9.

Samba provides three global `smb.conf` parameters to support modifying `smb.conf` through MS-RPC functions. These are

- `add share command`
- `change share command`
- `delete share command`

Each of these parameters accepts the name of a program or script that should modify `smb.conf` to reflect the requested changes. In order to execute any of these commands, `smbd` requires that the user requesting the change be connected as the root account. Being a member of the "`Domain Admins`" group is not enough in this case. A common means of

doing this is to connect to the \\server\ipc$ share prior to connecting to the server using the management tools.

```
C:\WINNT\ > net use \\beveled\ipc$ /user:root
```

Remember that, as a member server, smbd will first ask a DC to authenticate the user. If this fails, smbd will attempt to look up the user in a local smbpasswd file. Hour 7, "Security Levels and Passwords," covered how to add a user to Samba's smbpasswd file when using encrypted passwords. Refer to that hour if you need a refresher in order to add root to the smbpasswd file.

The add share command and change share command scripts are each passed the following four parameters:

- The absolute path to the current smb.conf
- The name of the share in question
- The absolute path to an existing directory on disk, which will act as the root of the share
- A comment for the share

The delete share command requires only two arguments:

- The absolute path to the current smb.conf
- The name of the share in question

Samba 2.2.2 includes a perl script written to illustrate how a minimal change share script would work. The script, named modify_samba_config.pl, is stored in the examples/misc/ directory of the Samba source archives. We will add it to our smb.conf as

```
## provide hooks for responding to Server Manager/MMC
add share command    = /usr/local/samba/lib/modify_samba_config.pl
delete share command = /usr/local/samba/lib/modify_samba_config.pl
change share command = /usr/local/samba/lib/modify_samba_config.pl
```

Before testing this configuration, we will instruct smbd to tell our connection to re-read its configuration file by sending it the hangup signal (that is, kill -HUP <pid>).

Figure 14.11 shows how the Samba server appears in the Computer Management Plug-in to the MMC running on a Windows 2000 client. The only part of this plug-in that is currently supported is the branch that deals with shared directories. Any attempt to use other branches, such as the "Event Viewer" or "Services and Applications," will result in an error.

FIGURE 14.11
Viewing the shares on a Samba server from the Windows 2000 MMC.

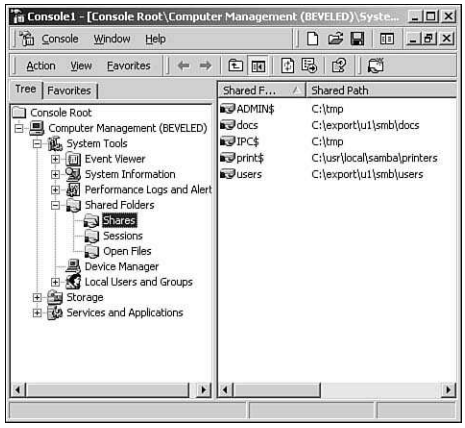


Figure 14.12 displays the MMC dialog used to create a new shared directory. The three available fields in the window correspond directly to three of the four parameters passed to the `add share command` and `change share command` scripts.

FIGURE 14.12
Creating a new share on a Samba server using the Windows 2000 MMC.



The one parameter that is less than intuitive is the path to the folder to be shared. The MMC GUI requires that this path be formatted as a DOS absolute path, including beginning with a driver letter (for example, `C:`). Samba strips this beginning drive letter and converts the remaining path to a Unix one. For example, the following string

```
c:\export\smb\public
```

would be internally converted to

```
/export/smb/public
```

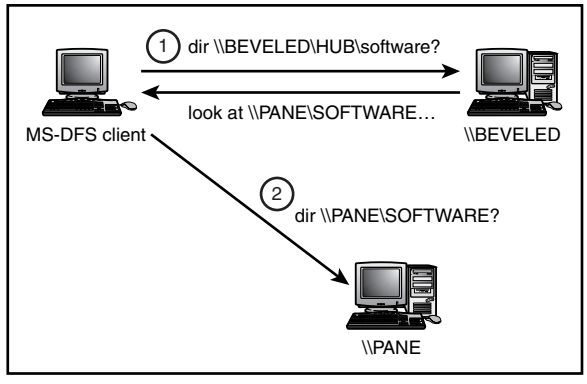
before passing it to the add or change share hooks.

To determine if the command completed successfully, `smbd` reparses the `smb.conf` after executing the hook. If the share has not been added/deleted/modified, then `smbd` returns an error to the client.

Microsoft's Distributed File System

The final topic we will cover in this hour is Microsoft's Distributed File system (MS-DFS). MS-DFS has been around for quite a while, but has been pushed to the forefront with the release of Windows 2000 and XP. In simplest terms, MS-DFS can be described as an automount system for Windows. Figure 14.13 illustrates how it works.

FIGURE 14.13
Implementing the Microsoft Distributed File System.



The client connects to a top-level share on the server. When attempting to access the "software" directory, the server responds that this directory is really located on the server `PANE` and informs the client of the UNC path to access the share (that is, `\\PANE\SOFTWARE)`.

All of this occurs without the user's knowledge and is handled entirely by the operating system. This means that files can be moved from server to server, and as long as the referral link in the DFS root is updated, the client will automatically be able to locate the necessary data. There is, however, an underlying assumption that the user logged on to the client machine has a valid account with the same username and password on both `BEVELED` and `PANE`. This implies that MS-DFS is more suited to use in a domain environment, as opposed to a workgroup setting.

All releases of Windows NT/2000/XP and Windows 98/ME include a built-in MS-DFS client with the "Client for Microsoft Networks." There is an add-on client available for Windows 95 that can be downloaded from Microsoft at `http://microsoft.com/ntserver/nts/downloads/winfeatures/NTSDistrFile/default.asp`.

I have already mentioned one of the advantages of MS-DFS—the capability to move data to a new location and have this shift be completely transparent to users. Other advantages include

- The ability to specify the list servers that house exact copies of the same files can provide the possibility of load balancing.
- Users can access data on multiple servers without having to have the complete alphabet displayed in the "My Computer" window.

Configuring Samba to support Microsoft's DFS begins with enabling the `--with-msdfs` option to the `./configure` script when Samba is compiled. In future releases, this will probably be enabled by default, but in Samba 2.2.2, unless you specifically ask for it, you will not get it.

Once Samba has been compiled to support MS-DFS, there are two additional `smb.conf` parameters available. The `host msdfs` global `smb.conf` flag must be enabled so that `smbd` will report its new capability to clients when they establish an initial connection to the server (that is, during the protocol negotiation phase of the connection setup).

```
host msdfs = yes
```

The second available setting is used to tag an individual file share as an MS-DFS root share. This means that the share can contain DFS referrals and is used to inform `smbd` that it needs to make an extra check when accessing files that are symbolic links.

```
## A file share for storing MS-DFS referrals to other
## servers and shares
[hub]
     path = /export/smb/hub
     msdfs root = yes
```

MS-DFS referrals within the top level of this share are actually a special feature of symbolic links on the Unix file system. These symlinks are of the form

```
name -> msdfs:server\share[,server\share]
```

The `name` is the directory name displayed to clients viewing the contents of the share. The keyword `msdfs:` is used to denote that the symlink is storing information on an MS-DFS referral and not just pointing to a non-existent directory. The UNC path of the referral follows `msdfs:` with the server name and share name separated by a single backslash character (`\`). It is possible to encode multiple referrals by separating each server\share path with a comma. This list is then returned to the client, which selects one to use.



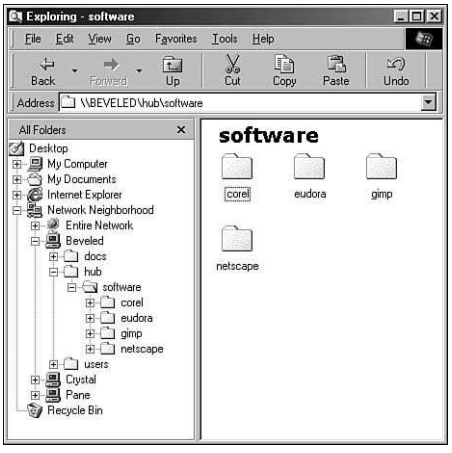
MS-DFS referrals stored on a Samba file service are not necessarily limited to the top level of the share. This is a fact that I am going to wave my hands over and ignore for our discussion here.

Here we will create an MS-DFS referral to `\\PANE\SOFTWARE` as shown in Figure 14.13.

```
root# cd /export/smb/hub
root# ln -s msdfs:pane\\software software
```

From the user's point of view, this link is displayed just as any other directory, as is illustrated by the Windows Explorer in Figure 14.14.

FIGURE 14.14
Browsing a DFS link using the Windows Explorer on a Windows 98 client.





More information on MS-DFS can be found at `http://microsoft.com/ntserver/nts/downloads/winfeatures/NTSDistrFile/AdminGuide.asp`.

Summary

During this hour, we have looked at how the various features of Samba can be combined to provide a viable alternative to a Windows NT/2000 file and print server.

Samba meets the three requirements necessary that I set forth:

- The Samba server should appear in the same workgroup and use the same NetBIOS machine name as the existing server in order to minimize confusion for

the users. The `netbios name` and `workgroup` `smb.conf` parameters enable us to assume the identity of the previous server after we have been joined to the domain as a new member server (that is, `security = domain`).

- All Domain users should be able to access the new shares on the Samba server without requiring a synchronized UNIX account on the machine. The existing Windows NT domain controller will perform all logon validation, and Winbind's NSS module automatically handles the issues regarding mapping Windows NT SIDs to Unix `uids/gids`.
- The access control mechanism to files should be kept the same so that a user who has access to a file on the existing server should have access to the same file on the Samba server. Also, a user who does not have access to a file on the existing server should not be able to access that file under the new configuration.

  Samba 2.2's new capability to manipulate POSIX ACLs means that we can simply copy the data from an existing NTFS partition to a Samba file share using the appropriate tools and maintain the security settings on objects. Although this is not a complete emulation of NT ACLs, it is a best effort that often captures the intent of users and administrators.

We also saw how Samba can be managed using standard Windows NT administration tools such as the Windows NT 4.0 Server Manager and the Windows 2000 MMC. Finally, we examined how to integrate a Samba server into a deployment of Microsoft's Distributed File system in order to provide better manageability and accessibility to network storage.

Q&A

Q Winbind is not available on my platform. Is there any way I can better manage Unix user accounts than creating them by hand?

A Samba supports two parameters, `add user script` and `delete user script`, for adding and creating users as needed. The details of these settings are described in the `smb.conf`(5) man page.

Q Why is `security = domain` better than `security = server`?

A There are three reasons why `security = domain` is better. The first is because this method enables the Samba server to participate in domain trust relationships. This is impossible with server-level security. The second reason is that, under server-level security, each `smbd` process must keep an open connection with the authentication server. This can drain a Windows NT PDC quickly. Under domain-level

security, this connection is maintained only long enough to perform the validation, thus conserving valuable resources. The final reason is that, as a domain member, the Samba server has access to much more information about user accounts, which can be used to automate the creation and deletion of user accounts upon demand.

Q Samba doesn't seem to support the file systems ACLs provided on my OS. What can I do to get my platform supported?

A Each server file system ACL implementation that Samba supports requires a separate set of back end functions to handle writing the POSIX ACL out to disk. There are two possibilities for you in this case. The first is to ask for help on the samba or samba-technical mailing list (`http://lists.samba.org`) to see if someone is already working on this. The second option is to implement the support for yourself. This option requires some motivation and a working knowledge of programming in C. If you choose this route, coordinate your work with others on the samba-technical mailing list so that they can benefit as well.

New Terms

ACL An Access Control List is an attribute associated with files, directories, and printers that allows for restricting or granting the capability to manipulate a given object.



HOUR 15

Server-Side Automation

Automation is the process of designing a solution that can be carried out without human intervention. It is closely related to scalability. Our goal for this hour is to develop methods that automate Samba's capability to handle connections from various client machines and users. Even if there are not large numbers of users, computers, or servers on your network, automation can still be of benefit because it leaves time for other things.

What Is Server-Side Automation?

Server-side automation in Samba is my term for events that the system administrator has configured to occur on the server as a direct result of a client connecting to a service, usually either a directory or printer.

Thinking back to previous hours, we have already seen two examples of server-side automation. The `[homes]` service is an example of Samba's built-in support for server-side automation? When a user attempts to connect to his home directory (for example, `\\pogo\joe`)—assuming the `[homes]` service has been defined—Samba attempts to locate the share name, first in

smb.conf and then in the local passwd file. If the name is found in /etc/passwd, Samba automatically creates a copy of the [homes] share and renames it to the name of the connecting user. This occurs transparently to the user and without any human intervention. If you create an account for a new user on the Unix box, you don't need to change anything in the smb.conf file to enable that user to access his home directory. That's what is referred to as server-side automation. The solution is scalable and self-maintaining.

The [printers] service from Hour 9, "Samba—The Print Server," is another example of Samba's built-in automation. We do not need to explicitly define every printer available on the system for the printer to be available. Samba will obtain its list of valid printer names from the file defined by the printcap name option. If we create another printer, Samba will be aware of it automatically.

One of the features we will use extensively in automating Samba is the set of runtime variables available in smb.conf. We first examined these in Hour 5, "Exploring Samba's smb.conf File." Table 5.1 describes the complete list if you need to refer back. Variables provide the foundation for individualizing connections. Some of the more commonly used variables are %u, %U, %g, %G, %m, %L, and %d.

preexec and postexec Scripts

Samba provides four service parameters that enable commands to be executed on the server when a client connects to or disconnects from a service. These are

- preexec (also known as exec)
- postexec
- root preexec
- root postexec

The preexec and postexec Parameters

We will examine the preexec and postexec parameters first. Both parameters take a series of commands or a script name that is executed on the server under the context of the connected user. We can understand better the point at which a preexec script executes by examining output from the net use command on a Windows box.

```
C:\users\jerry>net use s: \\pogo\my_share
The command was completed successfully.
```

The preexec command executes between the time that the net use command runs and the time that Windows reports that The command was completed successfully. The same is true for postexec commands. The Windows client sends the disconnect request

(`net use s: /d`). Samba then executes the `postexec` command and tells the Windows machine that the service was disconnected:

```
C:\users\jerry>net use s: /d
The command was completed successfully.
```

There are many things that can be done using the `preexec` and `postexec` commands. As a practical example, consider this scenario. More than once, I have had a user call to report that his Unix shell account didn't seem to be working correctly. After a few questions, it became clear that the user had accessed his home directory from a PC and deleted several files that didn't seem important. A friend of mine jokingly refers to these as, "Those pesky dot files!" By now, you can probably imagine which files the user deleted (that is, `.login`, `.bashrc`, `.cshrc`, or `.logout`). Here is an example that will help alleviate some of the help desk calls for this particular problem. The dot files are copied to the user's home directory upon connecting to the [homes] share. This script might make more sense run as a `postexec` command to prevent the user from deleting the files while logged onto a Windows client. That is left to your discretion as an administrator.

```
[homes]
      ## copy all .* files for user if they do not exist
      preexec = /usr/local/bin/fix_dot_files.sh %H
```

The script itself is very simple as shown in Listing 15.1.

LISTING 15.1 Bourne Shell Script to Copy Default Initialization Files

```
#!/bin/sh

## Define the directory which will contain initial .*
## files for users.  This might be /etc/skel instead on your system
SKELDIR=/usr/local/etc/skel


EXPORT SKELDIR
home=$1

## Copy over any files that do not exist
if [ ! -f $home/.login ]; then
   cp -p $SKELDIR/.login $home
fi

if [ ! -f $home/.logout ]; then
   cp -p $SKELDIR/.logout $home
fi

if [ ! -f $home/.profile ]; then
```

continues

LISTING 15.1 Continued

```
   cp -p $SKELDIR/.profile $home
fi

if [ ! -f $home/.cshrc ]; then
   cp -p $SKELDIR/.cshrc $home
fi
```



There is an associated `preexec close` Boolean parameter that determines whether `smbd` should examine the return value from the `preexec` command. If this `preexec close` is enabled, then the `preexec` command must return a value of 0 or else `smbd` will immediately close the connection to the share.

This next example might be a little silly and could get to be annoying for the user. The `preexec` command sends a copy of the `~/todo` file to the user on connection using a WinPopup message. It would probably be better to wrap this command in a script that first checks for the existence of the file:

```
[homes]
     preexec = cat %H/todo | /usr/local/samba/bin/smbclient -M %m
```

The `-M <machine name>` option informs `smbclient` that a WinPopup-style message should be sent to the host `<machine name>`.

The `postexec` parameter is similar to the `preexec` parameter except the value executes when the user closes a connection to the service. Here's an example that removes all files in the user's `~/tmp/` directory.

```
[homes]
      postexec = /bin/rm -rf %H/tmp/*
```

In this way, a user can store all the temporary files needed during a session in his home directory, thus slightly increasing security in an environment where many people use a machine, such as a student computer lab. I'll leave it up to you whether it is a good idea to delete these files automatically on your server.



All of the presented `preexec` and `postexec` examples have used the `[homes]` share. The use of the `preexec` and `postexec` options are not limited to the special share, but can be used with any file or printer share.

The `root preexec` and `root postexec` Parameters

There are a few differences between the `root preexec` and `root postexec` parameters and the normal `preexec` and `postexec` parameters. The obvious one is that the root variants are executed as the superuser account on the server. The second difference, which relates only to the `root preexec` parameter, is subtler. A normal `preexec` command will be executed after Samba has determined whether or not a user has permission to access the actual directory shared by the service. The `root preexec` command is executed before this check has been made, which means it has the potential to affect the outcome of the access query.

This can be very helpful for creating directories, setting ownership of files, or mounting and unmounting file devices, such as CD-ROMs or floppy disks.



There is a corresponding `root preexec close` parameter that acts analogously to `preexec close`.

One of my responsibilities when I was employed as a system administrator was to set up, update, and maintain several student-accessible Windows 9x/NT labs. We decided to build the labs to be as self-sufficient as possible so we could keep service disruptions to a minimum and localize SMB traffic. All students are issued Unix accounts that the lab server obtains information about via NIS+ tables. All students have home directory space reserved for their Unix accounts. The home directory the students use for the lab, however, is separate from their Unix home space and local to the lab's server.

Rather than bothering to create a home directory for each user on the lab server at the time of account creation, we chose to create it the first time the user logged into a machine in the lab. Here's how it worked:

```
[homes]
     comment = PC Lab home directories
     root preexec = /usr/local/samba/bin/buildhome %u %g
     path = /export/home/%u
     valid user = %S
     create mask = 0600
     force create mode = 0600
     directory mask = 0700
     force directory mode = 0700
```

You will notice that the path is not the user's home directory from `/etc/passwd` (that is, `%H`). The path is set to a local disk, `/export/home`, and each user has a directory there. Initially the disk is empty.

Now the `root preexec` script comes into play. The script checks for the existence of the directory name `/export/home/<username>` and creates it if it doesn't exist. Here is the source for a simple `buildhome` script:

```
#!/bin/sh
##
## Simple script to create a user's home directory
##
umask 077
user=$1
group=$2

# Does the user's home directory exist ( export/home/$user )?
if [ ! -d /export/home/$user ]; then
   mkdir /export/home/$user
   chown $user /export/home/$user
   chgrp $group /export/home/$user
fi
```

Here's another fun example of something you can do with `postexec` scripts. I needed a method to access the SCSI CD-ROM on my Linux workstation from a Windows NT client located nearby. I set up Samba on the Linux host to share the CD-ROM and to mount/unmount the CD-ROM as needed. Here is the `smb.conf` service entry:

```
[cdrom]
        path = /cdrom
        root preexec = mount /cdrom
        root postexec = umount /cdrom && eject /cdrom
        read only = yes
        public = yes
```

Now I can insert a CD-ROM and execute

```
net use f: \\mymachine\cdrom
```

on the Windows NT client to make the CD-ROM available. When I disconnect from the CD-ROM

```
net use f: /d
```

the CD-ROM ejects automatically.



Programs or scripts run using Samba's `root preexec` or `root postexec` options should be treated with the same caution as `setuid` scripts. These scripts should in no way be modifiable by normal users. The examples in this chapter do not perform any checks of arguments passed in by `smbd`. It is always a good idea to include sanity checks in the scripts to avoid common security holes like symlink attacks and race conditions. For more information on secure programming, refer to books such as *Linux Programming Unleashed* by Kurt Wall (SAMS).

15

The `message command` Parameter

In the previous section, we saw how we could use the `preexec` parameter and the `smbclient` utility to send WinPopup-style messages from `smbd`. We did not explain how we could receive those same types of messages.

The `message command` parameter allows an administrator to define an action to be taken when `smbd` receives a WinPopup-style message. From the discussion of NetBIOS names in Hour 2, "Windows Networking Concepts," you know that names with the `<03>` resource tag represent the messenger server. The WinPopup messages are sent to this name. Figure 15.1 shows the WinPopup Windows 95 utility preparing to send a message to the Samba server named BILBO.



FIGURE 15.1
Sending a WinPopup message from a Windows 95 client.

Samba's default action is to discard WinPopup-style messages. However, many possibilities can be used to read the message once it has been received by `smbd`. The following setting will display the message sent by the WinPopup client (see Figure 15.1) in an `xterm` on the Samba server (see Figure 15.2).

```
message command = /bin/bash -c '/usr/X11R6/bin/xterm -T \
     "WinPopup Message" -e /usr/bin/vim %s; rm %s' &
```

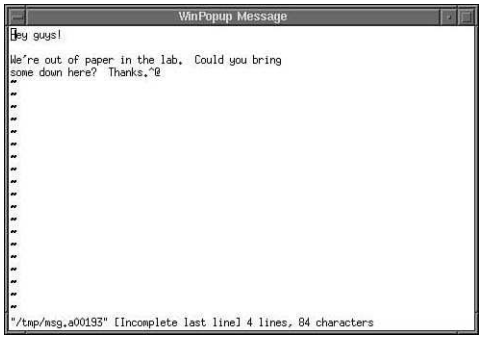


FIGURE 15.2
Displaying a received WinPopup message in the vi editor in an `xterm`.

Other options are using `sendmail` to deliver the message to a user's mailbox or sending the message to the `syslogd` daemon so it will be displayed on the system console.

Be aware that the WinPopup message is delivered as the default `guest account` on `smb.conf` (usually the `nobody` account). The command can contain additional variables besides the standard ones previously discussed. These are listed in Table 15.1.

TABLE 15.1 Additional Variables Available for the `message command` Parameter Value

Variable	*Description*
`%s`	The name of the file containing the message body.
`%t`	The destination name to which the message was sent. This is normally the name of the server.
`%f`	The name of the client who sent the message.

There are a few items to be aware of when setting a message command:

- Unless the commands you specify are in the default search path for the shell that is executed, you need to use absolute paths to the binaries.
- You must explicitly remove the received message or else it remains after the `message command` has completed.
- The `message command` should return immediately, or else the sending client can hang until a timeout period occurs. Notice that our example executed the command in the background by specifying the `&` job control character.

The `include` Parameter

If you have ever written a computer program in C, you will be familiar with the `#include filename` preprocessor directive. This directive tells the preprocessor to include the entire text of the given file lexically into the source code at that point. Samba's `include` parameter performs the same function.

The parameter's value is he absolute path to a file whose contents will replace the current occurrence of the `include` line. If Samba cannot open the file specified, the include parameter has no effect.

Let's use this sample `smb.conf` file:

```
; smb.conf
[global]
     netbios name      = POGO
     workgroup         = STY-SAMBA
```

```
     security         = user
     include          = /usr/local/samba/lib/shares.conf
```

Here are the contents of `/usr/local/samba/lib/shares.conf`:

```
; shares.conf
[foo]
     comment = example disk share
     path = /export/smb/foo
[homes]
     read only = no
     valid user = %S
```

The resulting file after parsing would be

```
; smb.conf
 [global]
     netbios name     = POGO
     workgroup        = STY-SAMBA
     security         = user
; shares.conf
[foo]
     comment = example disk share
     path = /export/smb/foo
[homes]
     read only = no
     valid user = %S
```

What difference does this make and why would you want to do something like this?

Some of the most powerful features of the `include` parameter are made apparent when using it in conjunction with `smb.conf` (or environment) variables. A frequent combination is to make use of the `%L` variable to design a Samba server that exhibits different behavior based on the called name used in client's NetBIOS Session setup request. I have used this technique more than once to consolidate multiple smaller servers into a single larger server. Windows clients often cache UNC network paths that contain the server name. By maintaining the old server name, clients are still able to connect to familiar services and can be weaned to use the new server name over time.

In Hour 5, the following `smb.conf` excerpt was presented to discuss the `netbios aliases` option.

```
[global]
   netbios name = CAESAR
   workgroup = ROME
   netbios aliases = ITALY GREECE
```

This story was left unfinished to a certain degree. Imagine for a moment that the Samba (or Windows NT) servers Italy and Greece had the `smb.conf` files shown in Listing 15.2 and 15.3.

LISTING 15.2 `smb.conf` for `\\ITALY`

```
[global]
    netbios name = ITALY
    workgroup = ROME
    security = user
    encrypt passwords = yes
    smb passwd file = /etc/smbpasswd

[athens]
    path = /export/athens
```

LISTING 15.3 `smb.conf` for `\\GREECE`

```
[global]
    netbios name = GREECE
    workgroup = ROME
    security = user
    encrypt passwords = no
    password level = 4

[homes]
   read only = no

[venice]
    path = /export/athens
```

To combine the servers `\\ITALY` and `\\GREECE` into a single server with multiple personalities, a few conditions must hold true.

- Users who possessed an account (in `/etc/passwd` and possibly the `smbpasswd` file) on the old servers must also possess an account on the new server.
- All file shares must be copied from the old servers onto the new server.
- Users should not notice that one server exists in the place of the previous two. In other words, a user who validated using a clear-text login should still be able to do so with the new server, but should not be able to access any shares on the new server to which she did not previously have access.

Let's develop our new `smb.conf` for `\\CAESAR`.

```
[global]
    netbios name = CAESAR
    workgroup = ROME
    netbios aliases = ITALY GREECE
    include = /etc/smb-%L.conf
```

The new `include` line will add the contents one of the following files

- `smb-caesar.conf`
- `smb-italy.conf`
- `smb-greece.conf`

depending on what the client used in its NetBIOS session request. For the moment, we'll assume the `smb-caesar.conf` file is empty. The contents of the new `smb-italy.conf` and `smb-greece.conf` files are shown next.

LISTING 15.4 `smb-italy.conf`

```
## smb-italy.conf
    security = user
    encrypt passwords = yes
    smb passwd file = /etc/smbpasswd

[venice]
    path = /export/venice
## end of smb-italy.conf
```

LISTING 15.5 `smb-greece.conf`

```
## smb-greece.conf
    security = user
    encrypt passwords = no
    password level = 4

[homes]
    read only = no

[athens]
    path = /export/athens
## end of smb-italy.conf
```

In this way we are able to support two different CIFS servers, one using clear-text passwords and one using encrypted passwords, on a single physical machine. Our example used a single workgroup for all servers because there is no means to configure a single Samba 2.2 server as a member of multiple workgroups.

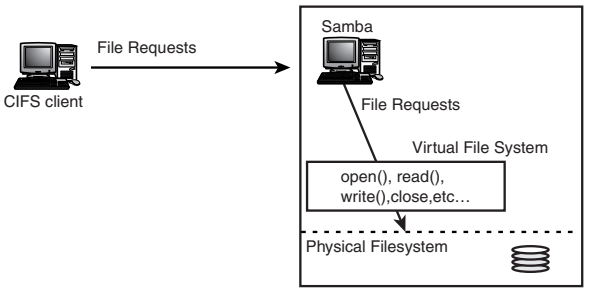
The `smb.conf` file also supports a `config file` parameter that replaces the current `smb.conf` with the filename specified by the parameter's value (for example, `config file = /etc/other-smb.conf`). Although this may appear similar to the `include` parameter, it is not as flexible (nor as powerful).

Samba's Virtual File System for Shares

The last section in this hour describes a feature new to Samba 2.2 and points to some future possibilities of its use. One of the GNU `autoconf` options mentioned briefly in Hour 3, "Obtaining the Latest Release of Samba," was the `--with-vfs` switch.

A *virtual file system (VFS) module* is a shared library that can be loaded by `smbd` to replace all file and directories operations in a file service. Figure 15.3 illustrates how a VFS module interacts between `smbd` and the exported file system.

FIGURE 15.3
An illustration of Samba's virtual file system (VFS) interface.



Samba's VFS provides a flexible means of overriding default behavior for one or two operations and allowing `smbd` to handle the reaming VFS options normally. A few examples of what a custom VFS module could implement include

- A Samba recycle bin for a share by redefining the VFS `unlink` operation
- A file auditing system by logging uses of the VFS `open`, `read`, `write`, and `close` operations
- A replacement for the `preexec` and `postexec` options by defining `connect` and `disconnect` functions

In order to load a VFS, the path to the shared library is specified using the `vfs object` parameter, such as used here

```
[foo]
   path = /export/foo
   vfs object = /usr/local/samba/lib/libsomevfs.so
```



Samba 2.2 does not currently ship with any production ready VFS modules, but expect to see continuing support with future Samba releases. If you are interested in writing your own VFS module, see the documentation in the `examples/vfs/` directory of the Samba source distribution and the current interface definition in `source/include/vfs.h`. You may also wish to join the `samba-technical` mailing list (see `http://lists.samba.org/` for details) and coordinate with other developers.

Summary

Samba provides many tools for automating activities on the server side of CIFS connections. By combining the use of the various `preexec` and `postexec` parameters, `smb.conf` variables, and the `include` directive, a server can be configured to exhibit dynamic behavior to meet the needs of clients. Using the `netbios aliases` option in conjunction with the `include` parameter can offer a means of emulating multiple Samba servers using a single machine. Samba's VFS interface provides a programmatic means of overriding file and directory operations that access the physical file system.

Q&A

Q **Can a share have both a `root preexec` and a `preexec` setting?**

A Yes, it is possible to configure a service to have both a `root preexec` and a `preexec` setting or to have both a `root postexec` and a `postexec` setting.

Q **Can the `%U`, `%u`, `%G`, and `%g` variables be used in the `include` parameter?**

A Yes. This would enable you to specify settings on an individual or group basis. An example would be to include a departmental share by adding `include = %G.conf` to the configuration, where `%G.conf` would contain the definition for some share.

New Terms

automation The process of performing a job or task without human intervention.

server-side automation Those things that occur on the server's end of a network connection that are automated. Examples would be dynamic share creation or reconfiguration based on the client connecting.

Virtual File System (VFS) An interface in Samba which is used to abstract the functions for performing file I/O operations such as open(), close(), read() and write().



HOUR 16

Managing User Accounts and Single Sign-On

It is impossible to discuss a Samba installation without migrating to the topic of user authentication at some point. We have already spent a good deal of time in Hour 7 covering the various models used by Samba to validate user connection requests. During this hour, we will explore how to coordinate this authentication between Unix and Windows when a user has an account in both realms.

Some of these problems were alluded to in Hour 7 when we were introduced to the `username map` `smb.conf` parameter. This option was used to point to a file that contained a mapping of Windows usernames such as "`Gerald Carter`" to a Unix account name, such as `jerry`. Not only do we have to deal with differences in login names, but also we must potentially deal with two sets of passwords for users. In order to support Microsoft's challenge/response authentication, Samba requires that the system administrator maintain an `smbpasswd` file that contains a list of login names and password hashes. There is no automatic or default correlation between the

passwords in `/etc/passwd` and those in `smbpasswd`. In fact, it is impossible to generically convert one to the other. The question to answer this hour is, "How can we automatically synchronize entries stored in these two files?"

An alternative to synchronizing the password entries in `/etc/passwd` and Samba's `smbpasswd` file is to establish a single sign-on model, which allows Unix services to use a Windows Domain Controller to authenticate users or vice-versa. During Hour 14, "Replacing a Windows NT File and Print Server—Lock, Stock & Barrel," we examined two pieces of the Winbind package—the NSS module and the `winbindd` daemon. During this hour, we will add a PAM module to these two components. Doing so will allow us to perform such feats as logging on to a Unix server via SSH or Telnet while using a Windows Domain User account.

On the flip side, configuring Samba as the Primary Domain Controller provides a means of validating Windows services, such as connecting to Windows NT member servers against a Unix server. The full story on Samba's domain controlling features will be presented in Hours 21 and 22.

Password Synchronization Between `/etc/passwd` and `smbpasswd`

We will begin our discussion with a topic that has been the bane of Samba administrators since the beginning—synchronizing passwords between the local system password database and Samba's `smbpasswd` file. The basic idea is to capture all password changes sent from CIFS clients and to update the password for the corresponding Unix account as well.

Our examples will assume that `encrypt passwords = yes` has been set in `smb.conf`. In particular, password changing from Windows 9x/ME clients can be rather difficult to support if password encryption is not used. The reason is that the clients transmit password strings in all uppercase. Any experienced administrator will immediately wave a red flag at case insensitive strings because it makes the password easier to guess by limiting the number of valid characters available for use.

Figure 16.1 gives the chronological flow of events for a user password update.

1. First the client sends the password change request.
2. Samba verifies that the client knows the old password hash by using the copy stored in the `smbpasswd` as the decryption key for the new password. If the server is unable to decrypt the new password, it means that the client did not know the old password hash.

3. If the `unix password sync` parameter has been enabled in `smb.conf`, `smbd` will attempt to change the Unix password first. If this step fails, an `error` message is returned to the client.
4. If the Unix password was successfully changed, or if `unix password sync = no`, `smbd` will then change the `smbpasswd` entry.

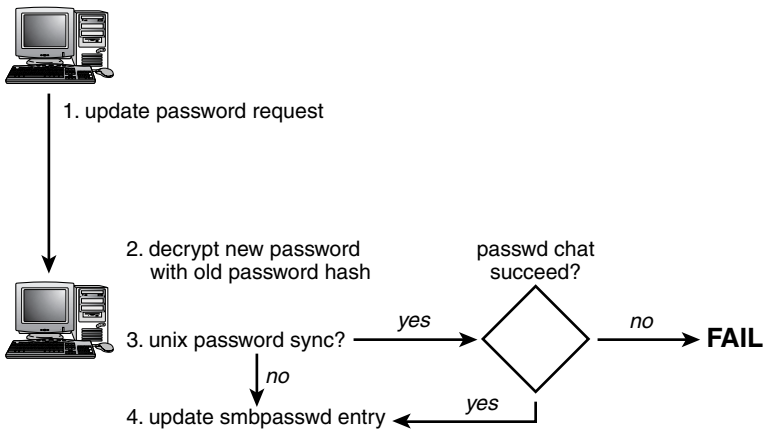


FIGURE 16.1 *Synchronizing passwords via Samba.*

Table 16.1 presents several global `smb.conf` options related to password changing. Since the `unix password sync` parameter has already been mentioned, we will begin there.

TABLE 16.1 Password Changing Parameters

Name	*Default*	*Description*
`passwd chat`	`"*new*password* %n\\n *new*password* %n\\n *changed*"`	Defines the conversation between `smbd` and the `passwd program`.
`passwd chat debug`	`no`	Should the chat script be run in debug mode? If enabled, and if the `log level` is set to a value greater than 100, the clear text of new passwords will be written to the process's log file.
`passwd program`	`/bin/passwd`	Specify the program to be called when changing a Unix user's password.
`unix password sync`	`no`	Should `smbd` attempt to update the user's `/etc/passwd` entry as well as the hash in the `smbpasswd` file?
`min passwd length`	`5`	Defines the minimum number of characters allowed in a new password.

To begin our password synchronization changes to `smb.conf`, we must enable the `unix password sync` parameter, or else the remaining options in Table 16.1 are ignored. At the same time, we must specify the program used to change Unix account credentials. Because the `passwd program` is executed as root, the `%u` variable is used to specify the name of the user whose password should be updated.

```
[global]
     <...>
     ## enable password synchronization feature
     unix password sync = yes

     ## use the 'passwd' program to change credentials
     passwd program = /usr/bin/passwd %u
```



Without the `%u` appended to the `/usr/bin/passwd` string, this example would change root's password *without your knowledge*. Even worse, it would change it to the password string sent by a user (and we know how hard it can be to crack passwords like "steve123").

The next step is the most site-specific value to define. The `passwd chat` option defines the expected conversation between `smbd` and the `passwd program` for a successful password change. There are two new variables unique to the `passwd chat` script that need to be introduced.

- `%o`—The old password string. This string is not known when using encrypted passwords.
- `%n`—The clear text of the new password.

To develop a working `passwd chat` script, we must know what to expect from a successful password change. The easiest way to capture this is to run the `passwd program` by hand. Here we are changing the password as root for a user named `jerry`.

```
root# /usr/bin/passwd jerry
Changing password for user jerry
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

Samba allows us to use certain control characters, such as `\n` (line feed), `\r` (carriage return), `\t` (tab), and `\s` (white space), plus a wildcard (`*`) to match any sequence of characters. From the password changing session already shown, we can shorten this conversation down to

```
     passwd chat = *New*d: %n\n*Re*d: %n\n*success*
```

Spend a few minutes staring at the `passwd chat` value and the output from the `passwd program` to convince yourself that this chat script will work.



For the password chat to succeed without knowing the old password, the root user on the system must be able to change the password for the user specified by the command line argument to the `passwd program`. If the Samba server is using NIS/YP, it must be an NIS master. It is best to verify that the super-user can reset the password for an arbitrary user (and without the current password for the account) before configuring a `passwd chat` to be used by `smbd`.

Finding the correct conversation string between `smbd` and your `passwd program` can be tricky. If you find yourself stumped, setting the following three options in the global section of `smb.conf` can be helpful.

```
passwd chat debug = yes
log level = 100
log file = /var/log/log.%m
```

This will record the plain text of the `passwd chat` interaction in Samba's log file. This includes passwords as well as response strings. Use these parameters only when troubleshooting a password change problem and then immediately disable `passwd chat debug` and set the `log level` to a more appropriate setting.

Samba `--with-pam`

Samba's support for Pluggable Authentication Modules (PAM) was briefly mentioned in Hour 7 when we discussed user authentication. Many Unix administrators are still unfamiliar with PAM, in spite of its existence since the mid 1990's. Vipin Samar and Roland J. Schemers, III, both from SunSoft, Inc., defined the PAM interface in the Open Software Foundation's RFC 86.0 (1995). Figure 16.2 gives a brief overview of PAM and how it interacts with a particular Unix server (for example, an `ssh` server).

In this diagram, the user attempts to log on to the server using an `ssh` client. The `sshd` daemon asks the PAM facility a series of questions such as "Do these credentials match?" and "Has this account been disabled?" These are only examples of what could be asked. The answer to each query depends on the PAM modules configured for each layer. The `auth` level is the only one required in this case. The remaining levels, `account` and `session`, could be configured to always return true.

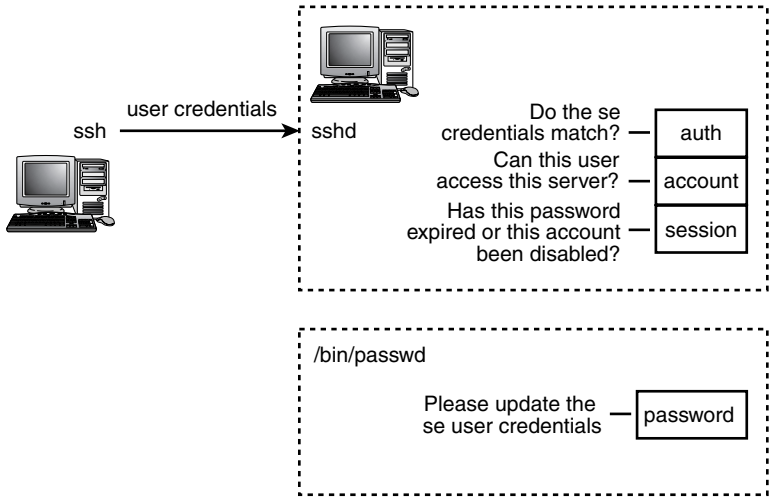


FIGURE 16.2
Features provided by PAM (`auth`*,* `account`*,* `session`*, and* `password`*).*

In the lower box, the `/bin/passwd` utility is used to update the user credentials stored behind the PAM interface, which could be propagated to multiple locations or formats (for example, `NIS`, `LDAP`, `/etc/passwd`, and so forth). This last point illustrates how PAM abstracts the account storage and authentication mechanisms used by applications. These PAM-clients should never need to worry about how to access account information or how to store it. That is the responsibility of the PAM interface.

PAM in 5 Minutes

There are two common formats for configuring PAM services. The Linux-PAM implementation uses a one-file-per-service approach and places all of these configurations in the `/etc/pam.d/` directory. Other operating systems, such as Solaris, use a single configuration file called `/etc/pam.conf`. The only real difference between these two variations is that the latter prefaces each module-type with a service name. The following excerpt defines the authentication settings for the login service on a Solaris host.

```
## excerpt from /etc/pam.conf
login   auth required   /usr/lib/security/pam_unix.so
```

Using the Linux-PAM configuration, an equivalent of this setting would be stored in a file named `/etc/pam.d/login` and appear as

```
## excerpt from /etc/pam.d/login
auth required   /usr/lib/security/pam_unix.so
```

The examples that will follow in this hour use this Linux-PAM format, which can be generalized to

```
## /etc/pam.d/<service-name>
module-type    control-flag     module-path     arguments
```

The `module-type` flags are described in Table 16.2. A particular PAM module is not obligated to support all four module types. Winbind's PAM library supports only the `auth`, `account`, and `password` module types.

TABLE 16.2 PAM Module-Types

Module-Type	`pam_winbind`	*Description*
`auth`	`yes`	The module is able to prove that a user is who he or she claims to be. The library can also grant certain credentials, such as group membership, to the user.
`account`	`yes`	A module will normally use this type for enforcing certain non-authentication-based account management, such as restricting logins based on host or time of day.
`session`	`no`	The module provides mechanisms for performing account management for this session such as writing log entries, creating home directories, or copying initialization files.
`password`	`yes`	The library provides a mechanism to update the user's credentials for a particular authentication module.

For each module-type, there are four control flags. These control flags define whether or not the module must return success for the PAM facility to return success to the calling application. Table 16.3 gives a brief overview of each flag.

TABLE 16.3 PAM Control-Flags

Module-Type	*Description*
`required`	This module must succeed in order for the module-type facility to return a success message to the calling application. This failure will not be returned to the caller until all modules of this module type have been processed.
`requisite`	This flag is similar to the `required` flag except that a failure is immediately reported to the calling application upon encountering a `requisite` module.

continues

TABLE 16.3 Continued

Module-Type	*Description*
`sufficient`	This module is required for the success of the module-type facility. However, the success of a `sufficient` module is deemed enough to report success to the calling application, and no further modules in the module-type stack are processed.
`optional`	This module is `optional` to the success or failure of a module-type.

One of the more interesting recent developments with PAM is the use of a recursive PAM module named `pam_stack.so`. This library is used extensively by RedHat 7.x based distributions. Its purpose is to allow an administrator to define a consistent set of modules to be used for controlling access to system services. The following portion of the configuration file for the login service

```
## /etc/pam.d/login
auth       required     /lib/security/pam_securetty.so
auth       required     /lib/security/pam_nologin.so
auth       required     /lib/security/pam_stack.so service=system-auth
```

shows that

- Logons from root should be allowed only from a terminal listed in `/etc/securetty` (`pam_securetty.so`).
- All logons except root should be denied if a file named `/etc/nologin` exists (`pam_nologin.so`).
- The remainder of the authentication process should be performed by using the `auth` lines in `/etc/pam.d/system-auth` (`pam_stack.so`).

The authentication settings in the `system-auth` file for this example could be something like

```
## /etc/pam.d/system-auth
auth    sufficient  /lib/security/pam_pwdb.so likeauth nullok md5 shadow
auth    required    /lib/security/pam_deny.so
```

There is much more information regarding PAM available in your system's documentation (assuming it supports PAM at all). The best place to start is to view the PAM man page (that is, `man pam`). Linux systems frequently include additional documentation on specific modules in `/usr/share/doc/pam-XXX` where the `XXX` should be replaced with the version of Linux-PAM installed on the machine.

PAM Support in `smbd`

Samba 2.2 introduced improved support for PAM authentication, account management, session management, and password changing. All of this is available by simply adding the `--with-pam` parameter to the list of `./configure` options used when Samba is built.

```
root# ./configure --with-pam
<...output deleted...>
checking whether to use PAM password database... yes
checking for pam_get_data in -lpam... yes
<...output deleted...>

root# make
root# make install
```


One of the most common causes for the build process to fail is that the PAM development headers (for example, `/usr/include/security/pam_appl.h`) and libraries (for example, `/usr/lib/libpam.a`) are not present on the system. If this is the case, make sure that the required packages are installed and that the necessary files can be located by Samba's `Makefile`. On Linux systems, this often means that both the `pam` and `pam-devel` packages must be installed.



PAM support was broken in the Samba 2.2.0 release. If you plan to use PAM with `smbd`, it is recommended that you update to the latest version of Samba.

The `smbd` daemon uses the service name `samba` when locating PAM module settings. Here is a basic samba PAM configuration file used on a Linux server.

```
## /etc/pam.d/samba
auth        required    pam_nologin.so
auth        required    pam_pwdb.so nullok
account     required    pam_pwdb.so
session     required    pam_pwdb.so
password    required    pam_pwdb.so nullok
```

TABLE 16.4 `smb.conf` PAM Parameters

Name	*Default*	*Description*
obey pam restrictions	no	Should Samba enforce the account and session module-type requirements defined in `/etc/pam.d/samba`?
`pam passwd change`	`no`	Should `smbd` use the PAM facility for implementing Unix password changes controlled by `unix password sync`? If enabled, the PAM `password` settings in `/etc/pam.d/samba` will be used in the place of the `passwd program`.

Table 16.4 gives a quick description of each PAM-related option in `smb.conf`. Once Samba's compile-time PAM support has been enabled, there are three run-time questions that we must answer in order to understand how Samba should use this feature.

- Should `smbd` use the `auth` module-type for authenticating users (`encrypt passwords = no`)?
- Should Samba obey the `account` and `session` management module-types in its PAM configuration file (`obey pam restrictions = yes`)?
- Should `smbd` use the `password` module-type for changing credentials (`pam password change = yes`)?

Samba is able to use the `auth` module-type only if `encrypt passwords = no` (the default in 2.2 and all prior releases). For all PAM modules commonly distributed today, the invoking application must possess the clear text of the user's password in order to pass to the PAM facility. This requirement cannot be met when using NTLM to authenticate clients. Therefore, `smbd` will ignore the `auth` module-type in its PAM configuration file and always authenticate users against the `smbpasswd` file when using `encrypt passwords = yes`.

When attempting to change a user's system password in addition to the LanMan/NT hashes stored in `smbpasswd`, the clear text of the new password is known. This means that PAM modules can be used to update passwords for Unix accounts, but only if the old password is not required. Because `smbd` does not know the clear text of the original password, it executes the Unix password change as `root` in an effort to avoid being prompted for the old password. This does not always work. Some PAM modules require that the old password be known.

If the `pam password change` option is enabled, Samba will pass the `passwd chat` string to the PAM support libraries. This effectively replaces the `passwd program` parameter, but otherwise works exactly the same as the process described in the beginning of this hour.

If Samba should be instructed to obey the `account` and `session` module-types in `/etc/pam.d/samba`, then `obey pam restrictions` must be enabled. These checks will normally be made only in the case of a new CIFS session request or a domain logon request from an NT/2000/XP client. The default behavior is to ignore these two PAM module-types.

In order to integrate all aspects of PAM support in Samba, we must disable support for encrypted passwords and enable the remaining two PAM-related parameters.

```
[global]
     <...>
     ## honor the PAM auth module line
     encrypt paswords = no

     ## honor the PAM account and session module-types
     obey pam restrictions = yes

     ## honor the PAM password module-type
     ## must also set a valid "passwd chat" for this to work
     pam password change = yes
```


If `smbd` has been instructed to synchronize Unix passwords and `smbpasswd` entries, we can use the recommended configuration of `encrypt password = yes` while still utilizing PAM for `account`, `session`, and `password` management.

Unix Services in a Windows Domain—`pam_winbind`

Figure 16.3 was presented first in Hour 14 when we began to examine Winbind. You will remember that the Winbind package is composed of three components:

- The `winbindd` daemon, which manages the mappings between Windows NT SID's and Unix `uids/gids`. `winbindd` also provides some authentication glue for the PAM module so that it is not necessary to run `nmbd` and `smbd` to validate user credentials.
- An NSS module, which is used to generate `/etc/passwd` (and `/etc/group`) style entries from Windows domain accounts.
- A PAM module, which can be used by Unix services to validate logins and change passwords.

The first two items were covered in great detail in Hour 14. However, the PAM module was skipped during that hour.

The power of Winbind is the ability to implement scenarios such as the one shown in Figure 16.4. In this environment, Unix services, such as `sshd`, ask a Windows domain controller to authenticate users, as opposed to looking them up locally. This means that any account changes are seen immediately by Winbind clients. There is no account information to keep synchronized because there is only one account.

FIGURE 16.3
The Architecture of Winbind.

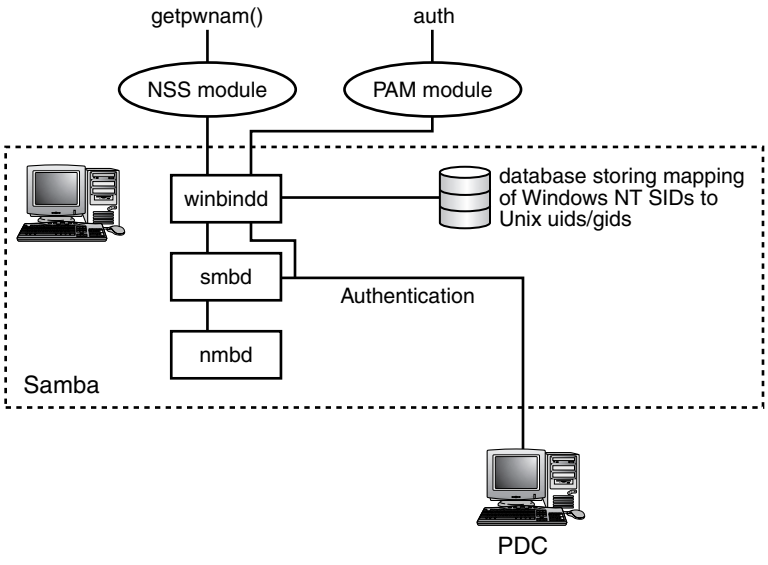
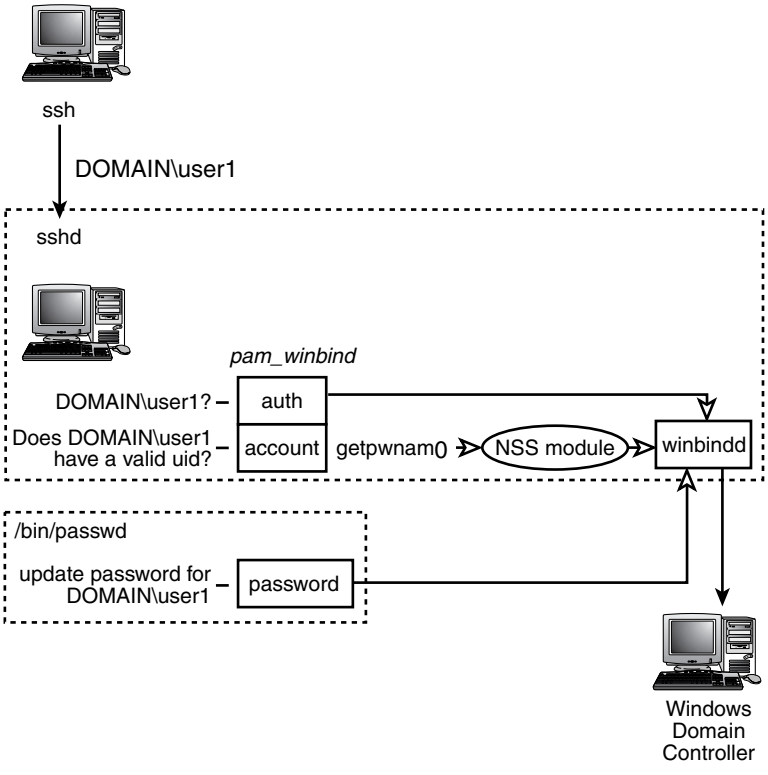


FIGURE 16.4
Using a Windows DC and Winbind to implement single sign-on for Unix and Windows users alike.



The first step to installing a usable Winbind PAM module on your system is to follow the instructions given in Hour 14 for joining Samba to the Windows domain. It is not necessary to run `smbd` and `nmbd` for our purposes here. The `winbindd` daemon alone will handle all the necessary communication with the Windows DC.

After joining the domain, installing the `libnss_winbind.so.2` library in `/lib`, adding it to the `passwd` and `group` database lines in `/etc/nsswitch.conf`, and launching `winbindd`, we are ready to focus on configuring the PAM component for use. The `make install` script used by Samba 2.2.2 does not copy the `pam_winbind` library to `/lib/security` for you. We must do this by hand by executing

```
root# cp /usr/local/src/samba-2.2.2/source/nsswitch/pam_winbind.so /lib/security
```

This assumes that the top level of the Samba source tree is located at `/usr/local/src/samba-2.2.2/`. Replace this with the appropriate path if your local Samba source tree is in a different folder.



Linux and Solaris use the `.so` extension for shared libraries. The actual name of the `pam_winbind` shared library may differ depending upon your server's operating system.

It is also a good idea to ensure that the ownership and permissions are set to prevent tampering from users.

```
root# chown root /lib/security/pam_winbind.so
root# chgrp root /lib/security/pam_winbind.so
root# chmod 755 /lib/security/pam_winbind.so
```

Now that the PAM module is installed correctly, we can begin to configure services to make use of it. The OpenSSH project is a free implementation of the SSH protocol first developed by Tatu Ylönen. This secure replacement for the traditional telnet server is becoming a common citizen on many networks (and for good reasons).

It is probably a good time to mention that not all Unix server applications include support for PAM. The first detail to check when experiencing problems with a PAM configuration is to verify that the application supports this interface at all. OpenSSH does include support for PAM, but it is not enabled by default. Quite often, binary distributions of OpenSSH available from vendors have been built to include PAM support. However, if it is necessary to build the `sshd` binary from source, simply add the `--with-pam` option to the configure script and verify that the parameter was recognized as shown here.

```
root# pwd
/usr/local/src/openssh-2.9.9p2
root# ./configure —with-pam
<...output deleted...>
OpenSSH has been configured with the following options:
                User binaries: /usr/local/bin
              System binaries: /usr/local/sbin
          Configuration files: /usr/local/etc
              Askpass program: /usr/local/libexec/ssh-askpass
                 Manual pages: /usr/local/man/manX
                     PID file: /var/run
       sshd default user PATH: /usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin
     Random number collection: Device (/dev/urandom)
                Manpage format: doc
                   PAM support: yes
<...output deleted...>
```

The last line of output indicates that OpenSSH was able to locate the files necessary to authenticate connections using PAM. The remainder of the installation process is as easy as

```
root# make
root# make install
```

On Linux systems, OpenSSH reads its PAM configuration information from `/etc/pam.d/sshd`. Winbind's PAM module supports only the `auth`, `account`, and `password` module-types.



Feel free to use another server software package for testing your `pam_winbind` installation. Just make sure that the software does support PAM before continuing.

First, we examine a simple `auth` module-type section. This example asks `winbindd` to authenticate the user against a Windows DC. If that succeeds, the user has passed the authentication test. If the user cannot be validated by the domain controller, the failure is ignored, but the user must have presented credentials that match those of a local Unix account.

```
## /etc/pam.d/sshd - OpenSSH configuration file
auth     sufficient     pam_winbind.so
auth     required       pam_unix.so use_first_pass
```

The `pam_winbind` library implements support for the `account` module-type by verifying that the system knows the username. This is done via a simple `getpwnam()` call, as shown in Figure 16.4. Assuming that the `libnss_winbind` module has been correctly installed and configured, this should always succeed.

```
account  sufficient     pam_winbind.so
account  required       pam_unix.so
```

Winbind does not currently contain support for the `session` module, so this section of `/etc/pam.d/sshd` is very simple. It consists of a single line containing the `pam_unix.so` library in order to support local session logging. Because Winbind's NSS module creates a Unix user from a Windows domain account, this step will log access from NT users as well as local Unix ones.

```
session  required       pam_unix.so     none
```


Once this file has been created and the `sshd` daemon has been started, we are ready to test the installation by connecting to the server as a domain user. The `winbind separator` has been set to the plus (+) character in `smb.conf`. We will use an account in the `BEDROCK` domain with the login name `jerry` for this test.

```
$ ssh localhost -l BEDROCK+jerry
BEDROCK+jerry@localhost's password:
Last login: Thu Sep 27 17:29:35 2001 from localhost
Have a lot of fun...
BEDROCK+jerry@pogo:~ > id
uid=10000(BEDROCK+jerry) gid=10002(BEDROCK+Domain Users)
groups=10002(BEDROCK+Domain Users)
```



To allow shell access to Windows domain users, the `template homedir` and `template shell` `smb.conf` parameters must be set to an existing directory and a valid shell, respectively.

Long delays or system hangs when trying to log in as a domain user can be caused by the server application attempting to iterate through all the user and group accounts in search of a match. For very large domains, this can be a problem. The `winbind enum users` and `winbind enum groups` options can be used to disable `winbindd` from responding to the C library `getpwent()` and `getgrent()` functions. By default, both of these parameters are enabled because certain applications can rely on being able to iterate through accounts. It is best to test the specific applications of concern when considering the use of either of these parameters.

`pam_winbind`'s password-changing support can be integrated with the standard `/bin/passwd` program by adding a line to the `/etc/pam.d/passwd` file similar to

```
password  required   pam_cracklib.so retry=3
## NEW : change the password for remote Windows domain users
password  sufficient pam_winbind.so use_authtok
## change the password for local Unix users
password  required   pam_pwdb.so nullok use_authtok md5 shadow use_first_pass
```



The `password` support in the `pam_winbind` library was broken in the 2.2.2 Samba release (which is the latest at the time of this writing). The Samba Team plans to correct this problem in the 2.2.3 release.

There are many more PAM-aware applications that can be used this way. `OpenSSH` and the `/bin/passwd` are just two examples.

Single Sign-On Samba and Unix—`pam_smbpass`

The sole purpose of Samba's `smbpasswd` file is to be able to authenticate Windows clients using their native network language (that is, NTLM). However, the main disadvantage of this solution is that Samba then requires that two different sets of account be maintained. One set is the standard `/etc/passwd` (or equivalent) used by Unix services such as `login`, and the second is the LanManager and NT password hashes used by `smbd` to implement the NTLM. I have spoken with many Samba administrators who have gone to great lengths and developed very clever solutions for synchronizing passwords in these two files.

From the previous section, we know how to configure Unix services to authenticate Windows domain user accounts. We will next explore how to reverse the scenario and implement a single sign-on mechanism that integrates Unix services and Samba's `smbpasswd` file. Steve Langasek's `pam_smbpass` library allows Unix services, such as OpenSSH, to authenticate users directly against Samba's `smbpasswd` file. This gives us the possibility of ignoring synchronization between `/etc/passwd` and `smbpasswd` completely and using only the latter for authentication needs. Figure 16.5 illustrates how this approach would work.



The `pam_smbpass` library will also support the experimental SAM databases discussed later in this hour. However, the compile options for the PAM module must match the storage format used. For example, it is not possible to use a version of `pam_smbpass` that was compiled to support LDAP with a `smbpasswd` file. More on this will be covered before the end of this hour.

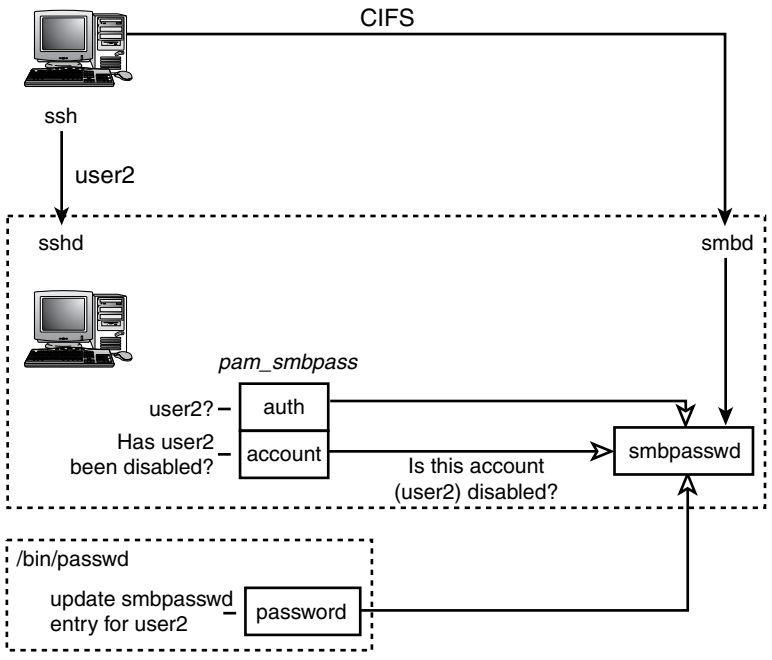


FIGURE 16.5
Using `smbpasswd` *as a central database of user passwords.*

Here, the `sshd` server authenticates the connection request from `user2` by looking up the user's password in the `smbpasswd` file. When connecting to a file or printer share using CIFS, `smbd` also validates `user2` against the `smbpasswd` file. When `user2` updates her password, either from a Windows client or with the `/bin/passwd` program, the new password is immediately seen by `sshd` and `smbd` because both daemons are pointing to the same set of accounts.

The `pam_smbpass` source files are distributed with Samba, but are built only if the `--with-pam_smbpass` option is given to the `configure` script. Because the PAM module has nothing to do with Samba's own internal authentication mechanisms and PAM support, including the `--with-pam` option is not a requirement.

```
$ ./configure --with-pam_smbpass
<...output deleted...>
checking whether to use pam_smbpass... yes
<...output deleted...>
```

After building the library, it must be manually copied to the `/lib/security` directory and have appropriate ownership/permissions assigned.

```
root# cp /usr/local/src/samba-2.2.2/source/nsswitch/pam_smbpass.so /lib/security
root# chown root /lib/security/pam_smbpass.so
root# chgrp root /lib/security/pam_smbpass.so
root# chmod 755 /lib/security/pam_smbpass.so
```

There is additional documentation for the module in the `source/pam_smbpass/` directory of the Samba source distribution. The main thing to remember is that `pam_smbpass` contains support only for the `auth`, `account`, and `password` module-types.

In order to configure the OpenSSH `sshd` server to authenticate users first against `smbpasswd` and only fall back to the local `/etc/passwd` file in the case of a failure (for example, a bad password or unknown user), the following two lines will be placed in `/etc/pam.d/sshd`:

```
## /etc/pam.d/sshd - OpenSSH configuration file
auth     sufficient     pam_smbpass.so
auth     required       pam_unix.so use_first_pass
```

`pam_smbpass`'s `account` module support is composed of checking for disabled accounts in `smbpasswd` denoted by the '`D`' in the account flags field (for example, `[UD ]`).

```
account    sufficient     pam_smbpass.so
account    required       pam_unix.so
```

At the beginning of this hour, we explored how to catch password change requests from CIFS clients and synchronize the user's Unix password entry with `smbpasswd`. The `pam_smbpass` library allows us to catch password updates in the reverse direction. When a user executes a PAM-aware version of `/bin/passwd` from a Unix shell, the `pam_smbpass` library can also ensure that the `smbpasswd` entry for that user is updated. In fact, because we are relying upon `smbpasswd` for all authentication requests, the `/etc/pam.d/passwd` configuration file can be boiled down to

```
## /etc/pam.d/passwd
<...>
password   requisite   pam_cracklib.so retry=3
password   sufficient  pam_smbpass.so use_authtok use_first_pass
password   required    pam_unix.so shadow md5 use_authtok try_first_pass
```

The tailing `pam_unix.so` line is present to support local password changes, such as those affecting the root account.

Our entire discussion of `pam_smbpass` has assumed that it would be used on the same server where Samba was installed. To work around this assumption, we will need a means of distributing the `smbpasswd` from our main Samba server to `pam_smbpass` clients, such as what is shown in Figure 16.6. In this diagram, the server is using tools to copy files securely to another host. Because of the security sensitive nature of `smbpasswd`, I will wait to cover the details of this until next hour, when we will address general Samba security considerations.

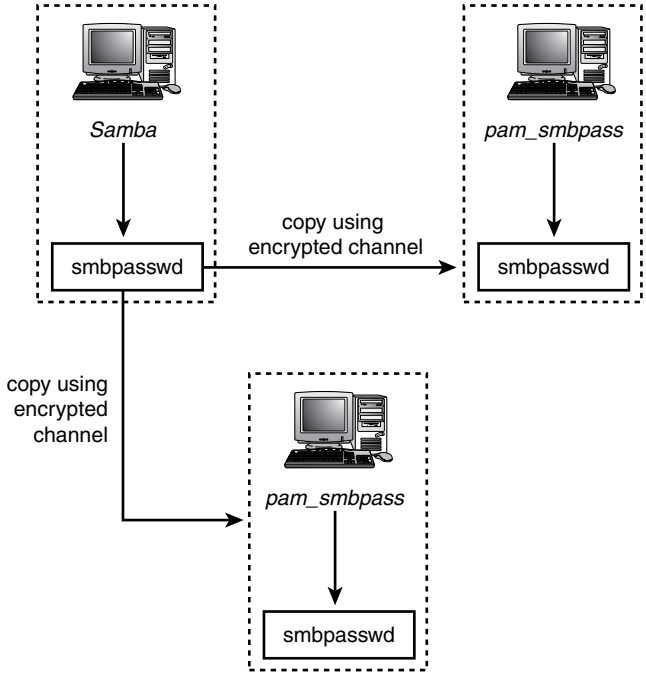


FIGURE 16.6
Distributing `smbpasswd` *to multiple servers.*

Recording User Connections to the Unix Login Records (`utmp`)

Many Unix administrators rely on using the `last(1)` command (or some customized version of it) to view the list of currently and previously logged-on users. Samba includes support for logging user sessions to the system `utmp(5)` file using the same interfaces as common Unix services, such as `login(1)` or `getty(1)`.

During this section, we will see how to enable Samba's feature of recording connections in the server's `utmp` file. This is a different twist on managing users than we have previously looked at because it does not relate to the process of authenticating the user. Rather, we are consolidating information regarding access to a system into a single location. Tools already exist for mining this file and processing the information found.

To enable this system logging, the `--with-utmp` option must be passed to Samba's configure script at compile time.

```
$ ./configure --with-utmp
<...output delete...>
checking whether to support utmp accounting... yes
<...output delete...>
```

Once the binaries resulting from running `make` are built and installed, there are two new global `smb.conf` settings to be introduced. The first is the `utmp` parameter, which determines whether or not the `utmp` logging feature should be enabled. Even though the `--with-utmp` argument was specified when compiling Samba, the `utmp` parameter is disabled by default. We will add this to the global section of `smb.conf` as

```
[global]
     <...>
     ## Log Samba sessions
     utmp = yes
```

The next parameters, `utmp directory` and `wtmp directory`, may or may not be needed, depending on whether the `utmp` (or `wtmp`) file resides in a non-standard directory. This option has no default value, indicating that Samba will use the standard location for these files as defined by the operating system.

To see the effects of this feature and how the connection is reported using the `last` command, we will view the last 3 log entries for the user named `jerry`.

```
$ last -3 jerry
jerry    smb/1        192.168.215.23   Thu Sep 27 21:28   still logged in
jerry    pts/6        192.168.1.137    Thu Sep 27 20:10 - 20:58  (00:47)
jerry    tty2                          Thu Sep 27 19:46   still logged in
```

Samba entries show a terminal name of `smb`. This particular entry (`smb/1`) is from a Windows NT client connected to two shares. Because the `utmp` implementation in Samba only logs sessions, there is only one entry for both tree connections. If you remember from Hour 2 and Hour 7, when operating in user level security, there is a single session setup step to obtaining a session `uid` for the client. This `uid` is then used in all subsequent connections to shares. Therefore, these two connections are contained in one session.

Samba's New `--with-ldapsam`, `--with-tdbsam`, and `--with-nisplussam` Options

For the final minutes of this hour, we will look at the new support in Samba 2.2.2 for experimental replacements to the traditional `smbpasswd` file. These replacements come in the form of NIS+ tables (`--with-nisplussam`), LDAP directories (`--with-ldapsam`), and local indexed databases (`--with-tdbsam`).

Technically, one problem with using a flat text file, such as `smbpasswd`, for storing account information is that searching for a specific user can be extremely slow when the file contains thousands of entries. The new SAM-like account storage backends are designed with fast lookups by using a search key, such as the user's login name or Windows NT SID.

A second problem, which is visible only when using Samba as a domain controller, is the lack of sufficient fields for storing all the information associated with a Windows NT user. In this respect, Windows NT/2000 is much more flexible than Samba, since each user has specific attributes stored with his or her account entry. Samba represents many of these attributes with `smb.conf` parameters (for example, `logon script`). Though the use of `smb.conf` variables can extend the flexibility of a setting for individual accounts, it does not provide the per user settings that Windows NT administrators are accustomed to.

The various `--with-XXXsam` configuration options all use a common structure for representing a user. This includes fields such as the UNC path to the user's home directory, the logon script for the user, the last time the user's password was changed, and when his password expires.

Support for Samba's traditional `smbpasswd` files will not be removed any time soon—possibly never. However, larger sites will most definitely want to explore these new storage formats for better performance and integration with existing directory services.

Summary

During this hour, we have examined ways of synchronizing password entries in Samba's `smbpasswd` file with the user's corresponding entry in the local system's password database (for example, `/etc/passwd`). Samba's `unix password sync` configuration option can be used to control whether or not smbd should update an account's Unix password when the user changes his (or her) password from a CIFS client.

Samba 2.2 also provides improved PAM support, which can be used to leverage existing security policies. Many administrators rely upon PAM to provide a consistent interface for authenticating and managing users.

We also explored alternatives to password synchronization by using the `pam_winbind` and `pam_smbpass` libraries to enable Unix services to authenticate users against a Windows domain and Samba's `smbpasswd` file, respectively. A complementing feature to this single sign-on is the capability to centrally log user sessions to the Unix `utmp` and `wtmp` files. In this way, administrators can use standard tools such as `last(1)` for viewing connections from Samba and `ssh` in the same location.

Q&A

Q What is the number one mistake to watch out for when first using PAM?

A This is a tie between two errors. The first mistake is to forget to create the PAM configuration file for the service. On many systems, this results in the application

using default settings, which may deny all authentication requests. The second error is to not enable PAM support when building the server application in the first place.

Q I cannot get the `smbd` to update users' Unix passwords when I have `encrypt passwords = no`. Why is this?

A Although the `unix password sync` parameter was not designed to be used when encrypted passwords have been disabled, the real problem is due to Windows clients that transmit passwords in all upper case letters. In these cases it is impossible to know what the user intended.



HOUR 17

Security Tips

Events of the past months and years, such as Code Red and the Nimda worm, have proven that networks, and the Internet in particular, can be very unfriendly places at times. Network security is an extremely broad topic that touches almost every aspect of a system administrator's daily life—from e-mail to file services to social engineering. I will not be so presumptuous as to claim that we can fully cover all aspects of Windows Networking security, or even Samba for that matter, in a single chapter.

What we will do during this hour is look at some of the best practices used by Samba administrators to secure their servers and better protect their clients. Samba has the responsibility of enforcing security policies set in place by administrators when CIFS clients access resources such as files and other user owned information. It is the administrator's duty to define these security policies so that Samba cannot be used as an instrument for distributing damaging programs, such as computer viruses or Internet worms.

This hour should act as a checklist for every Samba server you install. Our principle areas of concern will be

- Authentication
- File permissions on sensitive configuration files

- Computer viruses, Trojans, and Worms
- Encryption between clients and servers
- Network Security and Firewalls

Each of the security practices may not be necessary for every server, but you should be aware of the risks involved if you do not implement each policy. With the growing availability of DSL and cable modems in the US, not even home networks are immune to attacks from intruders. Consider the data that is stored on your Samba server and then consider the cost in time and money if it was lost or damaged due to viruses or intruders.

Figure 17.1 captures the division of responsibilities between Samba, the operating system, and the local administrator. When the client attempts to open a file, Samba performs an access check to see if the connected user has the appropriate privileges before passing the `open` request to the underlying OS. The OS then performs another access check based upon the file system permissions. The access checks are controlled by the security settings defined by the administrator in `smb.conf` and by using the `chmod(1)` or `chown(1)` utilities to assign file/directory permissions and ownership.

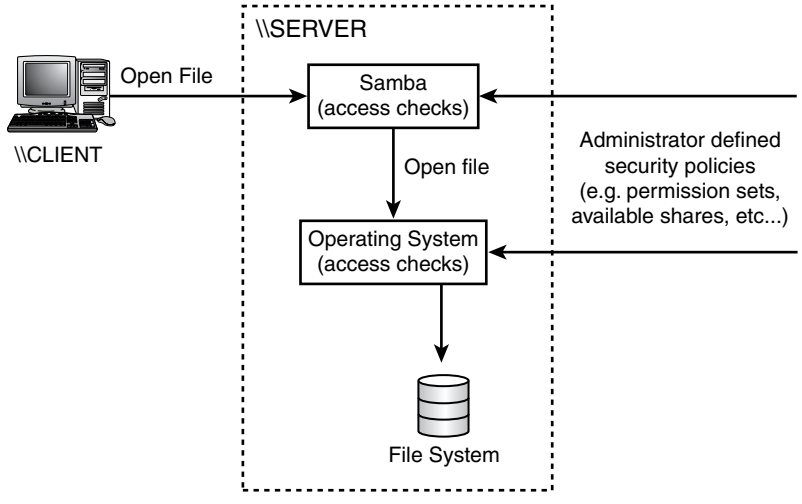


FIGURE 17.1
Samba relies on the underlying operating system to enforce many security policies beyond its control.

Authentication

The recommended configuration for Samba 2.0.x and later servers is to enable password encryption by using an `smbpasswd` file. Many administrators have completely removed services, such as Telnet and FTP, which use plain text authentication models from their networks. However, network security is only as strong as its weakest link. If Samba con-

tinues to ask Windows clients for the clear text of the password, then all other efforts to secure user accounts are in vain. It is well worth the effort to follow the instructions in Hour 7 for using `NTLM` authentication with Samba.



Samba developers are planning to make `encrypt passwords = yes` the default setting in Samba 3.0.

However, even within NTLM there are grades of security. The algorithm used to generate LanManager renders the resulting password case insensitive. This and a few other unmentioned details make it much easier to derive the original plain text string from LanMan passwords than from the newer NT hashing algorithm.

Many Windows NT administrators have disabled the use of LanMan passwords on their network altogether. This is possible only if the network is composed solely of Windows NT/2000/XP clients. Windows 9x/ME clients still use the older hashing algorithm.

By supporting both NT and LanMan passwords, Samba administrators can often be in for a rude awakening. When establishing a connection to a CIFS server, the Windows NT client transmits both the LanMan and the NT password hash in the session setup request. Samba will first attempt to authenticate the user with the NT password hash. If this fails, `smbd` will attempt to use the transmitted LanMan response. This means that a user can log in from a Windows NT client using a case insensitive password.

The use of LanManager passwords can be removed entirely in Samba 2.2 by disabling the `lanman auth` global parameter as shown here

```
lanman auth = no
```

Beware of using the setting carelessly. It will lock out all clients on your network except Windows NT/2000/XP and Samba's own `smbclient`. There are several other password- and security-related parameters presented in Table 17.1.



Samba 3.0 will disable this fallback behavior. When a client sends the `NTLM` challenge generated from the NT hash, `smbd` will use this only for authenticating the user. The `LanMan` challenge is ignored. The latter hash is used if it is the only one transmitted in the `SMBsesssetup&X` request.

TABLE 17.1 Password- and Protocol-Related Parameters

Name	*Default*	*Description*
`lanman auth`	`yes`	Should Samba use lanman passwords when authenticating users?
`max protocol`	`NT1`	What is the maximum CIFS protocol dialect that Samba should support? Possible values from lowest to highest include `CORE`, `COREPLUS`, `LANMAN1`, `LANMAN2`, and `NT1`.
`min protocol`	`CORE`	What is the minimum CIFS protocol dialect that Samba should support? Possible values from lowest to highest include `CORE`, `COREPLUS`, `LANMAN1`, `LANMAN2`, and `NT1`.
`null passwords`	no	Should `smbd` accept connections with a username but no password?
`restrict anonymous`	no	Should `smbd` accept connections with no username and password?

Configuration File Permissions

File permissions and ACLs were discussed in Hours 8 and 14. This time, we are not concerned about the security of data within shares as much as making sure that Samba can trust its configuration data. Figure 17.1 explains that Samba relies on the underlying OS to protect files using the permissions defined by the administrator. Samba's trust in its configuration data is based upon two things.

- The administrator specifies a logical security policy, such as forbidding normal users from modifying `smb.conf`.
- The operating system is able to enforce the security settings defined by the administrator.

If either of these two requirements is not upheld, Samba will continue to trust `smb.conf` and related files, but the system can easily be compromised.



A compromised system is one that cannot be trusted to tell the truth. If a cracker has gained root access on a server, then the logs could have been modified and therefore cannot be trusted to give an accurate report of events.

You should be able to identify which files are crucial to Samba's integrity from the topics covered in past hours. Table 17.2 displays the files of main concern in a default Samba 2.2 installation as well as the recommended security settings.

TABLE 17.2 Security Settings for Samba's Configuration Files

Filename	*Recommended Owner and Permissions*
`lmhosts`	root / 644
`smb.conf`	root / 644
The `username map` file	root / 600
`smbpasswd`	root / 600
`secrets.tdb`	root / 600

The type of risk associated with each file varies. Samba's main configuration, `smb.conf`, is the most sensitive to modification because the Samba daemons run as the root user. If a normal user has the capability to modify this file, that user can gain root access to the system in less than five minutes. Consider what would occur if a user named `bob` were able to create a share such as this one:

```
[myshare]
   path = /
   read only = no
   admin users = bob
```

`bob` would now have write access to every file on the server's file system. From here, changing the root's password to gain access to a shell prompt is only a few edits of `/etc/passwd` away. Once `bob` can login to a root shell on the server, there is no end to the mischief he can achieve.

This warning also applies to files that are writable by users and are later merged into `smb.conf` by the `include` parameter. This scenario is no prettier that the previous one where a user could modify `smb.conf` directly. In fact, the outcome is exactly the same.

A neglected `username map` file can also be a source of security problems. Imagine if our user `bob` were able to insert a line such as

```
root = bob
```

in the `username map` file. This is not as large a security hole as a writable `smb.conf` (although still an extremely bad situation to be in) because it is not immediately obvious how `bob` could gain a root shell. However, if `bob` is able create a symbolic link to `/etc/passwd` on a Samba share and `follow symlinks = yes` (the default), then `bob` has rooted the server again.

In Hour 7, we learned that the LanMan and NT password hashes stored in the smbpasswd file are clear text equivalents. In order to participate in an NTLM authentication session, the client needs to know only the hash of the original password. Because a password string will always hash to the same value, a hash is valid until the original password is changed. If an intruder is able to simply read a server's smbpasswd file, he can impersonate any user in the file.

There are certain features of Samba 2.2, such as using the "Take Ownership" button in the Windows NT security tab, that can be utilized only when connected to the Samba server as root. With the increased probability that an smbpasswd file will contain an entry for root in order to take advantage of some of the new features of Samba 2.2, the stakes have been raised in respect to protecting the list of password hashes. If a user is able to impersonate the root user when connecting to a Samba server, the possibilities are the same as the example where bob was able to modify the username map file.

Samba's secrets.tdb contains a variety of information. The most common item is the machine trust account information used when Samba is a member of a Windows Domain. This password can be used to impersonate the Samba server when communicating with the Windows domain controller. Certain information about domain users is available to member servers. Allowing non-root users access to this file can result in a security break of your Windows domain.

Finally, the lmhosts file is often overlooked by administrators but not by intruders. If the name resolve order has been set to query the local lmhosts file and a malicious user is able to change the IP address of the domain's PDC, Samba can be redirected to send authentication queries to the wrong server. The same is true for the wins.dat file in the case of wins support = yes.

Distributing smbpasswd to Multiple Servers

In some circumstances, it is necessary to distribute copies of the smbpasswd file from a master Samba server to multiple slave servers. During the early days of Unix, a similar process was done to synchronize a system's /etc/passwd. Modern networks use a directory service, such as NIS or LDAP, to distribute Unix user account information. Figure 17.2 shows how account information can be coordinated among several Samba/Unix servers. The "Secondary Samba Servers" are not necessarily replicas used to provide load balancing or failover capabilities for the main server. Here the term *secondary* is used to refer to the server's standing with regard to updating smbpasswd entries. Changes should be made only to the master server and then pushed out to the secondary servers.

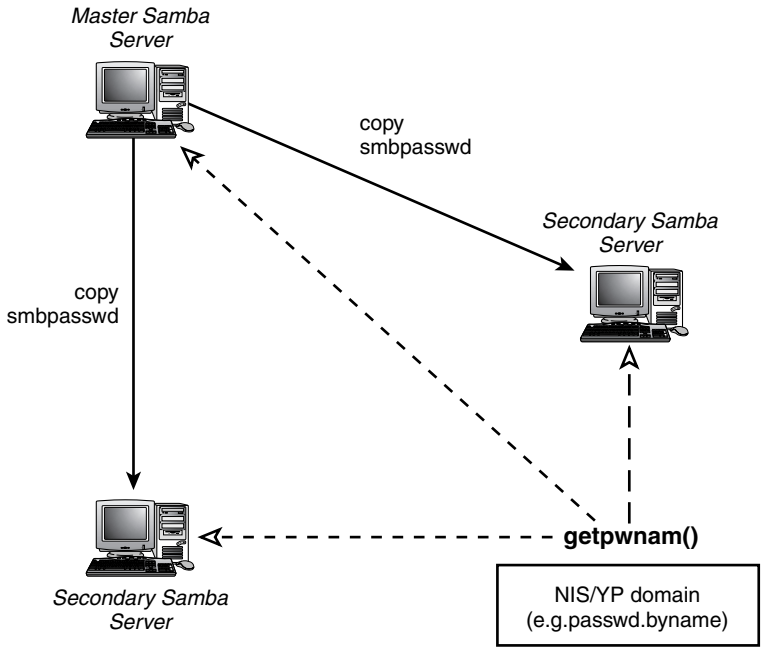


FIGURE 17.2 *Using* `smbpasswd` *copies on multiple servers.*

Given the nature of the hashes stored in `smbpasswd`, these entries should never be transmitted across a network without first securely encrypting the data. The most accepted means of doing this is to use an encrypted transport layer, such as IPSec or SSL. Another common solution is to use an encrypting remote copying tool, such as a secure copy (`scp`) or `rsync` when used with `ssh`.

The OpenSSH server was used last hour during our discussion of PAM. The package includes a secure replacement for the `rcp` command, aptly named `scp`. The full details of this tool are described in the `scp(1)` man page, but its basic form is almost identical to the syntax of `rcp`. To use `scp` to copy the `smbpasswd` file from the current machine to the host `queso`, we would execute

```
root# scp -c 3des /etc/smbpasswd queso:/etc/smbpasswd
The authenticity of host 'queso (192.168.1.74)' can't be established.
RSA1 key fingerprint is 16:cc:d2:d3:71:ca:a9:7b:cf:fd:1a:2e:22:cb:81:2e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'queso,192.168.1.74' (RSA1) to the
list of known hosts.
root@queso's password:
smbpasswd      100% |*************************   738 00:00
```

The `-c 3des` argument instructs `scp` to use the triple DES encryption algorithm for securing the communication between the current host and `queso`. If this option is omitted, `scp` will use whatever default encryption algorithm was configured for `ssh(1)`.

There are means of authenticating `ssh/scp` connections to servers using the clients RSA key. This option, fully described in the `sshd(8)` man page, provides an easy method for authenticating clients without having to prompt clients for passwords, which is better suited for running updates via the `cron(8)` facility.

Virus and Internet Worm Prevention

At the time of this writing, the most recent Internet worm to makes the rounds across the globe was the *Nimda* worm. This Win32 virus would search out all CIFS servers on the same network as the infected local machine and then it would probe the remote machine for file shares with an enabled guest account. Of course, if a user was logged on to the Nimda infected machine and had established authenticated connections to file servers, these would be used as well. The result was that Samba shares, even ones located on non Intel-based servers, appeared to be infected by a Win32 virus.

Samba, regardless of the hardware platform it is running on, will never execute Windows code, including scripting languages such as VBscript. So in this respect, a Samba server cannot be infected with a Windows virus. However, it is possible to use a Samba server, or any file server for that matter, to spread a virus or worm.

The simplest means of preventing the spread of a virus is to disable users from accessing infected files. This can be done using the `veto files` `smb.conf` service-level parameter. The veto files option accepts a list of filenames or wildcards that should be inaccessible to users. This is not the same thing as the DOS Hidden attribute. The files contained in the veto files list are not available to CIFS clients at all.



In Samba 2.2 and earlier releases, users can create files on a Samba share which matches the `veto files` list, but will never be able to access them again.

The following setting was used to prevent Samba from spreading the Nimda worm. Notice that the veto files parameter is defined in the `[global]` section, making it the default for all shares.

```
[global]
   <...>
   ## prevent the spread of the Nimda Internet worm
   veto files = /*.eml/*.nws/riched20.dll/
```

Specifying `veto files` can result in confusing behavior when trying to remove a directory on a Samba file share from a Windows client. Because the user is unable to see any of the vetoed files, a directory may appear to be empty when it is actually not. This can cause attempts to remove the folder to fail with an error message that the directory still contains files or directories.

If vetoed files need to be automatically deleted when a directory is removed, the `delete veto files` option must be enabled

```
[global]
   <...>
   ## remove any vetoed files when deleting a directory
   ## can result in a performance loss. See smb.conf(5)
   ## for details
   delete veto files = yes
```


Securing Access to SWAT Via SSL

The main security disadvantage with SWAT mentioned in Hour 6 was the transmission of unencrypted data, especially when sending the initial credentials to establish the identity of the user. (that is, the `root` password for the server). Just like the solution for securely distributing an `smbpasswd` from server to server, we will ask for help from some standard cryptography tools.

This time we turn to Stunnel, which is available at `http://www.stunnel.org/`. The `stunnel` server uses the OpenSSL libraries (`http://www.openssl.org`) to act as a proxy for non ssl-enabled applications. Figure 17.3 shows how `stunnel` is able to add HTTPS support to SWAT. The client talks directly to the `stunnel` daemon, which handles only the SSL communication. All HTTP traffic is passed off to SWAT.



SSL stands for "Secure Sockets Layer." The standard network-programming interface on Unix (and Windows) is the Berkeley Socket API. By combining cryptographic library with this programming model, we get the Secure Sockets Layer and hence the name.

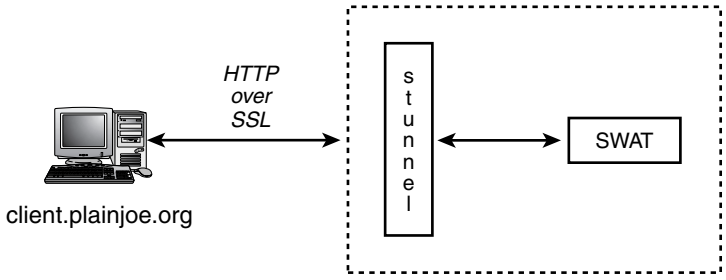


FIGURE 17.3
Using stunnel to implement HTTPS in SWAT.

Both OpenSSL and Stunnel must be properly installed prior to continuing. Cryptography and cryptographic software are not trivial to configure. Thankfully, there are binary distributions available for both packages. Check with your Unix vendor or favorite free software site for availability. If it becomes apparent that OpenSSL and Stunnel will need to be compiled from course, simply follow the documentation presented with the software and at each respective Web site. Generally this will boil down to the standard `./configure && make && make install`.

Once the OpenSSL and Stunnel packages have been installed, there are only three steps remaining.

1. Create the new certificate and private key

   ```
   root# /usr/bin/openssl req -new -x509 -days 365 -nodes \
     -config /usr/share/doc/packages/stunnel/stunnel.cnf \
     -out /etc/stunnel/stunnel.pem -keyout /etc/stunnel/stunnel.pem
   ```

   This step will present several questions needed to generate the certificate. The questions are self- explanatory, such as, "What country are you located in?" and "What is the fully-qualified domain name of your host?"

2. remove swat entry from `[x]inetd`
3. launch the stunnel server, as shown here

   ```
   root# stunnel -p /etc/stunnel/stunnel.pem \
   -d 901 -l /usr/local/samba/bin/swat swat
   ```

It should now be possible to connect our Web browser of choice to `https://<swathost>:901/` and enjoy the feeling of comfort that comes with knowing that your information in transit is secure.

Using Samba and SSL Together

Very few resource-sharing protocols, such as NFS and CIFS, were designed to be used across the Internet, despite what their names imply. There are times when even an inter-

nal network can be considered hostile. The campus networks at many universities are a prime example of this type of environment.

Because of the open access necessary for students and visiting faculty, many college IT shops take a defensive approach to network monitoring and administration. In this case, the resources that are foundational to the functioning of the network are aggressively protected. Most actions (and users) on the network are ignored unless they cause a disruption of service for other users. This is only one definition of a hostile environment. There are others that do not trust the network at all. In these environments, it must be possible to verify the integrity and author of each piece of data received from a remote client or server.

There are many aspects of the CIFS protocol that can be viewed as a weak security model. We have already talked about one—the use of clear text passwords for authentication. However, there are many more that allow packets to be modified in transit from the server to the client or vice versa. These so-called man-in-the-middle attacks can be thwarted by using encryption tools, such as OpenSSL.

Samba includes support for client connections using SSL. The problem is that very few CIFS clients (none of them Microsoft) actually support SSL connections. The only CIFS clients known to support these secure connections are `smbclient` (when Samba has been compiled to enable SSL) and Sharity from Objective Systems (`http://www.obdev.at`).

If you feel the need to enable SSL support in Samba, follow these steps:

1. Specify the `--with-ssl` option to Samba's `./configure` script. It may also be necessary to specify the location of the OpenSSL header files and libraries by defining `--with-sslinc` and `--with-ssllib` if the compiler cannot find the appropriate files.
2. Build and install Samba using the standard `make install` sequence.
3. Read the Samba-OpenSSL HOWTO included in the Samba documentation for the details of the various SSL-related `smb.conf` parameters.

For more information about SSL, public key cryptography and encryption protocols, Bruce Schneier's book *Applied Cryptography* (Wiley) is an excellent source of details regarding encryption algorithms and their practical use.

Firewalls

Network security is often implemented in layers. One of the more important components of this is the "keep the bad guys out" layer. Assume that we know that we will never connect to a CIFS server outside of our network and that only internal hosts should

connect to our Samba server. In this case, there is no reason to allow CIFS-related packets either destined to or arriving from the Internet to cross our network border (or gateway).

The computer that protects our network border is known as a firewall. There are too many products and operating systems available that offer firewall capabilities for us to examine them all. Our main concern in this section is determining which network ports should be disabled to protect our Samba and Windows Servers. Table 17.3 shows the most important ports and protocols in regard to protecting CIFS servers. All of these ports should be discarded as they enter or leave our network.

TABLE 17.3 Ports and Transports Used by the CIFS Protocols.

Port/Protocol	*Use*
`135/tcp`	End Point Mapper use by MS-RPC and DCOM (not needed by Samba)
`137/udp`	NetBIOS Name Service
`138/udp`	NetBIOS Datagram Service
`139/tcp`	NetBIOS Session Service
`445/tcp`	NetBIOS-less CIFS (used by Windows 2000/XP clients and experimental Samba releases)



The Linux 2.x kernel includes an implementation of a packet filtering firewall that has been rewritten in each release of the Linux 2.0, 2.2, and 2.4 kernels. The latest revision, named *Iptables* because of the tools used to manipulate the kernel filtering rules, has some very nice features and is well worth your investigation. Robert Ziegler's book, *Linux Firewalls* (published by New Riders) presents a nice overview of firewalls and how to implement them using a GNU/Linux 2.2 host.

The following log entries were collected from an office firewall running a Linux 2.4 kernel. These are just a few examples of some of the packets dropped. You can see the destination IP address (`24.216.85.59`) followed by a colon (`:`) and then the destination port number such as `137`, `139`, or `445`.

```
Sep 21 09:51:59 tumnus kernel: Packet log: input DENY eth0
   PROTO=6 24.82.165.179:3173 24.216.85.59:445 L=48 S=0x00
   I=27030 F=0x4000 T=118 SYN (#40)
Sep 30 01:36:57 tumnus kernel: Packet log: input DENY eth0
   PROTO=6 24.167.49.126:1796 24.216.85.59:139 L=48 S=0x00
   I=338 F=0x4000 T=113 SYN (#40)
Sep 30 23:03:55 tumnus kernel: Packet log: input DENY eth0
   PROTO=17 24.129.167.139:137 24.216.85.59:137 L=78 S=0x00
   I=40967 F=0x0000 T=118 (#41)
```

It is easy to see how the growth of cable modems and DSL is giving the *Network Neighborhood* a whole new meaning.

Samba's Very Own Bouncer

Have you ever wanted to be that rather large person who stood at the entrance to a club or a concert and had the final say on who was allowed in and who was not? Samba has its own personal bouncer that can be configured in `smb.conf`. During Hour 8, "Samba—The File Server," several parameters were presented to control access to a Samba server. These included the `valid/invalid users`, `max connections`, and `hosts allow/deny` options.

The last two parameters, `hosts allow` and `hosts deny`, can be used to protect Samba from unauthorized client machines by defining a list of hostnames, IP addresses, or networks that should (or should not) be allowed to connect to the server. Any connection will still require that the user present a valid password or can be mapped to a guest account. However, using these options can prevent users from sending account credentials from an untrusted host in the first place.

The values assigned to the `hosts allow` and `hosts deny` parameters follow the syntax of the `hosts_access(5)` man page installed as part of Wietse Venema's TCP Wrappers package (`ftp://ftp.porcupine.org/pub/security/`). The basic syntax can be described generally as a list of items separated by

- whitespace
- a comma
- or tabs

Each item in the list can take the form of

- a DNS hostname (for example, `pogo.plainjoe.org`)
- an IP address (for example, `192.168.1.65`)
- a network address (for example, `192.168.1.`)

- a network address and network mask (for example, `192.168.1.0/255.255.255.0`)
- a NIS netgroup (`@engineering`)

There are also two keywords available—`ALL` and `EXCEPT`. In the case that a given host could be denied or allowed, the `hosts allow` setting takes precedence.

To see how these settings work, imagine that we are trying to restrict access to a resource-limited server so as to allow only hosts that reside in the `192.168.10.0/255.255.255.0` subnet. This can be represented by

```
hosts allow = 192.168.10.
hosts deny = ALL
```

The `hosts deny` directive was implicitly stated by the `hosts allow` parameter. Any machine that does not match one of the entries in the `hosts allow` value will be denied.

Suppose that we wanted to restrict a single, problematic host from this list. How would that be represented? The answer is to use the `EXCEPT` keyword as shown here

```
hosts allow = 192.168.10. EXCEPT badhost.plainjoe.org
```

It may or may not be necessary to use a FQDN for the host, depending upon the Samba server's domain search setting in `/etc/resolv.conf`.

For the last example, we will give access to members of the NIS/YP netgroup `confusion` and a few other individual hosts.

```
hosts allow = @confusion caesar.plainjoe.org 192.168.1.200
```

An important point to make on closing, it is possible to define a default `hosts allow/deny` setting in the `[global]` section of `smb.conf`. However, it will not be possible then to specify a less restrictive set of hosts for a specific service. This is because the default `hosts allow/deny` value is applied prior to the client even attempting to establish a tree connection to the share (i.e. during the NetBIOS Session request).

Summary

During this hour, we have examined several aspects of securing a Samba server. Some of these suggestions revolve around additional `smb.conf` options such as the `hosts allow/deny` or `lanman auth` parameters. Others involved aspects external to Samba, such as securing configuration files with appropriate permission sets.

Our discussion can best be summed up with two thoughts.

- Network security is a multi-layered beast.
- Security is inversely proportional to convenience.

Choose and implement the security policies that best suit your needs and the needs of your users.

Remember that, due to constantly changing networks, security is a moving target that constantly requires monitoring. It is a good idea to maintain a close watch on the `samba@samba.org` mailing list for security announcements, as well as periodically checking for news posts on `http://samba.org/`.

Q&A

Q What should I do if I think I have identified a potential security hole in Samba?

A The first thing to do to make sure that the problem is not due to a misconfiguration in `smb.conf`. If everything appears to be configured correctly, the next step is to send an e-mail to `security@samba.org` with a detailed description of the problem and the steps to reproduce it. Helping developers recreate the problem will help them to determine if it is an actual security hole in Samba or something else.

Q How often should I upgrade Samba?

A As often as needed. Only you can gauge the exact risk a security exposure poses to your organization. Some organizations are comfortable with a level of risk other organizations find very threatening.

Q What's the worst Samba security sin?

A Number one is allowing non-administrative users to modify Samba's configuration. Number two is to not track configuration changes made by system administrators. Whether through comments in the file or through written records, it's important to know the rationale for a new share for setting and to revoke it when it's no longer necessary.



HOUR 18

WINS and NetBIOS Name Services

I've often thought it would be wonderful to have a photographic memory and be able to associate faces with names flawlessly. As a general rule, people address other people with a name. However, it is impossible to learn and remember the names of everyone in the world. We can hold a few of our close friends in our immediate memory, or cache. Whenever we see someone whose face and name is not in our memory cache, we must resort to asking a friend or companion who knows the person to help us.

Machines on a network deal with the same problem. They must contact unknown hosts on a daily basis. Imagine what happens when you visit a Web site for the first time. The address of the site's server is unknown to your computer until a DNS server gives you the correct information. Hour 2, "Windows Networking Concepts," briefly mentioned this problem in relation to Microsoft networks and also introduced NetBIOS name-registration and -resolution methods.

This hour focuses on NetBIOS name servers and other name-resolution mechanisms. We focus primarily on how Samba can use the Windows Internet Name Service (or WINS) and its own WINS implementation.

Why is WINS useful? The NetBIOS Name service is fundamental for many CIFS implementations, such as Samba and Windows 9x/ME/NT. When a CIFS client is unable to resolve a NetBIOS name, the client cannot establish a NetBIOS session with the destination machine. Therefore, all CIFS operations such as Network Browsing, Domain Logons, and connections to file shares and printers will break.

In closing, we examine how DNS affects NetBIOS name resolution in Samba and the `smb.conf` parameters available for configuring its behavior.

WINS

From Hour 2, you should remember that a NetBIOS client can register and resolve names using two methods: broadcast (sent to every machine on the subnet) or point-to-point (sent to a particular machine). One problem with this scheme is the amount of broadcast traffic that a large number of NetBIOS clients can generate. Another problem is that broadcasts are, by default, isolated to a logical subnet. This means that hosts on different subnets cannot communicate with each other. Figure 18.1 shows two hosts, CAESAR and BRUTUS, which cannot communicate or even be aware of each other because they are located on different networks.

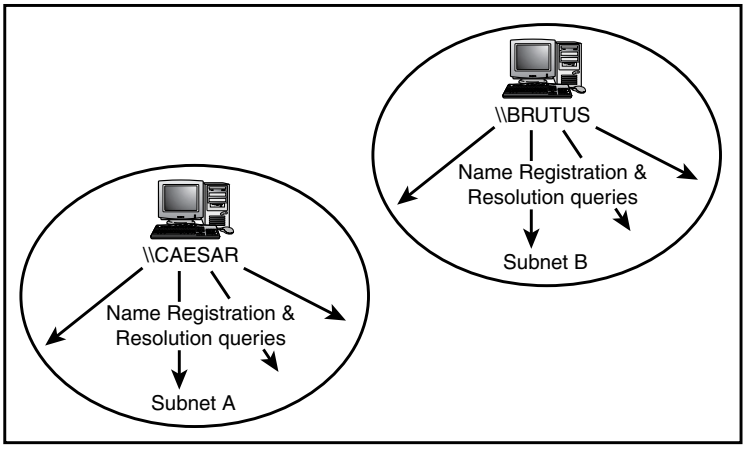


FIGURE 18.1
B-node NetBIOS hosts isolated from each other on separate IP subnets.

WINS was designed to address both these problems by configuring clients to register and resolve a name with a single machine directly. First, broadcast traffic is substantially reduced. Second, NetBIOS clients on different subnets can register and resolve names in the same name space. Although WINS provides the ability to extend the basic broadcast name space, it does not provide any additional functionality regarding name organization. The WINS name space is still flat, meaning that a unique NetBIOS name, such as a machine name, can be registered to only one host using the WINS server.

When a WINS-enabled client boots onto the network, it sends a name registration request directly to the WINS server. This name registration request is essentially the same as the broadcast name registration request discussed in Hour 2, except that it is unicast directly from the client to the WINS server.

The WINS server chooses one of three responses to the query:

- No response—The WINS server is disabled or otherwise configured to ignore packets from the client.
- Positive—If the WINS server finds no matching record in its database that would indicate a name collision with another NetBIOS host, the client is allowed to successfully register the name and the record is stored in the WINS database.
- Negative—If the WINS server locates a name in its database that matches the one in the registration request, it issues a name query request to the IP in the located record. It attempts this multiple times. If there is no response from the host that owns the name, the new client is allowed to register the name successfully. If the name's owner does respond, however, the requesting client is sent a negative response and is not allowed to register the name.

Name resolution works much in the same fashion. A client sends the name to be resolved in a packet to the WINS server. The server then searches its database for the name and, if the name is found, responds with the appropriate IP address. Otherwise, the server sends a negative response.

Figure 18.2 illustrates how this works. First, the client BRUTUS registers its name with the WINS server. Then, CAESAR asks the WINS server to resolve the name BRUTUS for it. The server replies with the IP address previously registered by the first machine. WINS is a requirement for any type of consolidation of multiple subnets, whether for browsing or for access to remote servers.

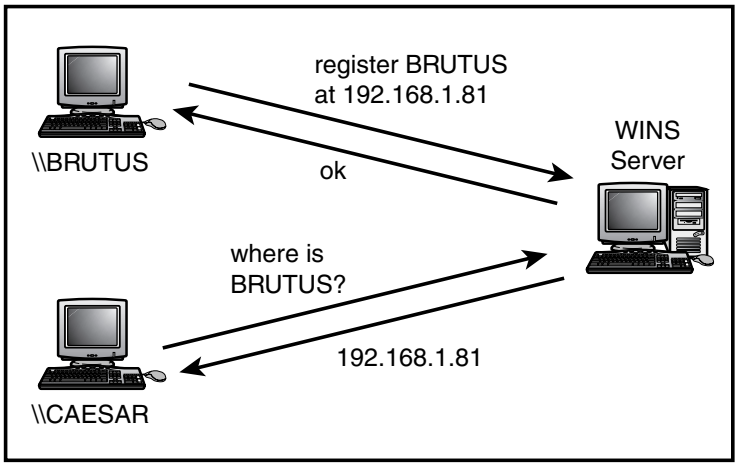


FIGURE 18.2
Name resolution using WINS.

The `wins server` Parameter

When implementing a NetBIOS network, only one WINS database should exist. Multiple servers can be used only if they are able to synchronize databases with each other, as the Windows NT 4.0 and 2000 WINS servers can. Otherwise, clients may be unable to access hosts that use different WINS servers. Remember, the purpose of WINS is to combine collections of NetBIOS names from individual subnets into one space.

Samba provides the `wins server` `smb.conf` global parameter to enable use of an external WINS server for registering and resolving names. This parameter accepts a single IP address of an existing WINS server as a value. It is possible to use the DNS name of the WINS server instead, but using the IP address is the preferred method. The default is to not use a WINS server at all.

Let's say that you have configured some machine (possibly another Samba server) at IP address 198.162.1.80 as a WINS server. You can specify that the current Samba server use this host for name registration and resolution queries simply by entering the following value in the `[global]` section of your `smb.conf` file:

```
wins server = 192.168.1.80
```

The `wins support` Parameter

If a machine is not currently providing WINS for your network, Samba can act as a WINS server if you enable the `wins support` parameter. This parameter, which is disabled by default, accepts a Boolean value and can be added in the `[global]` section of `smb.conf`, as shown here:

```
wins support = yes
```

Next, it is necessary to configure the necessary settings on the client. For Microsoft clients, this was covered in Hours 10 and 11.

Current versions of Samba use a flat text file, named `wins.dat`, for storing WINS information. Future releases will replace this file with a small database located in the `lock directory`. The best way to query the names and addresses register on a Samba WINS server is to use the `nmblookup` tool.

```
$ nmblookup –R –U <ip addr of WINS server> <name>
```

This same tool can be used to query Windows NT/2000 servers as well. If our Samba host's IP address is 192.168.1.1 and we wished to ask it for the IP address of the PDC of the NARNIA domain, we would execute the following command:

```
$ nmblookup –R –U 192.168.1.1 'narnia#1b'
querying narnia on 192.168.1.1
192.168.1.75 narnia<1b>
```

You should recognize the hexadecimal number following the # character in the domain's name. This is the resource byte for the master browser of a domain (which happens to be the name that Windows clients look up when asking for the domain's PDC).

If Samba is enabled as a WINS server, it internally configures itself to use the loopback address for performing WINS lookups and for registering its own NetBIOS name or names. In this way, Samba acts simultaneously as a WINS server and a WINS client.



Configuring both the `wins support` and `wins server` parameters concurrently is neither required nor recommended.

The `wins proxy` Parameter

If the TCP/IP settings on your PC clients are defined statically, it may prove difficult to implement WINS without paying a required visit to each machine in order to update the network configuration. One possible alternative is to set up a Samba server on each subnet to act as a proxy agent for broadcast name-resolution queries. This is illustrated in Figure 18.3.

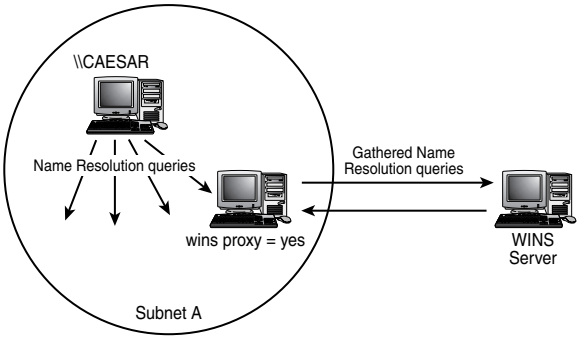


FIGURE 18.3
Using Samba as a proxy to handle broadcast name-resolution requests.

The `wins proxy` boolean parameter controls whether `nmbd` responds to broadcast queries on behalf of other hosts. If you enable this setting in the `[global]` section of `smb.conf`

```
wins proxy = yes
```

`nmbd` forwards name resolution queries to the WINS server specified in `smb.conf` (possibly itself in the case of `wins support = yes`) and also forwards the reply to the requesting client. The `wins proxy` parameter is disabled by default.

The `dns proxy` Parameter

When Samba is configured as a WINS server (`wins support = yes`), `nmbd` can be instructed to attempt to resolve name queries from clients using DNS if the name is not located in its WINS database. This behavior is controlled by the `dns proxy` Boolean parameter, which is disabled by default in Samba 2.2.

Adding

```
dns proxy = yes
```

to the `[global]` section of `smb.conf` causes `nmbd` to spawn a child process to handle any necessary DNS queries. This is done because the standard `gethostbyname()` function in Unix is a blocking call. The extra process allows DNS lookups without penalizing normal NetBIOS name service in the case of timeouts.



Samba queries DNS to resolve name queries from clients only if the resource byte of the name is <0x20>, meaning that the name refers to a file server. Samba never uses this feature to look up domain or workgroup names or nonserver names.

If you do not have a dedicated link to your DNS servers, for instance, if you connect to your DNS servers via a dial-up connection, disabling the `dns proxy` option prevents Samba from initiating a connection to your name servers every time someone queries a nonexistent name.

The `lmhosts` File

One method of name resolution that has not been addressed is looking up a name in a local `lmhosts` file. The LAN Manager `hosts` file is the functional equivalent of Unix's `/etc/hosts` file, except that it matches IP addresses to NetBIOS names rather than hostnames. The format of Samba's `lmhosts` file and the format used by Windows clients are slightly different, so the `lmhosts` file deserves some attention.

Information on the syntax of Samba `lmhosts` file can be found by reading its man page (`man lmhosts`). Each entry has this format:

```
IPaddress MachineName
```

Comments are denoted by the `#` character at the beginning of a line. The following listing shows an entry for a single server named `POGO`.

```
##
## lmhosts file with a single entry
##
192.168.1.75 POGO
```

Remember that NetBIOS names are not case sensitive, so we could have as easily used the name `pogo`. The `MachineName` entry can be any valid name accepted by the `nmblookup` tool. For example, you can specify a particular name type by appending a NetBIOS tag to the name.

```
##
## example lmhosts file illustrating how to use
## resource bytes in entries
##
192.168.1.75 POGO#20
192.168.1.75 NARNIA#1b
```

The first entry refers to the server resource `POGO`. Remember that `<20>` is the tag used to indicate share points. The second entry is used to locate the domain master browser for the group `NARNIA`.

Normally Samba's `lmhosts` file is located in the same directory as its configuration file, `/usr/local/samba/lib`. It is possible to specify another location when starting `nmbd` by using the `-H` filename flag. The following example starts `nmdb` as a daemon using `/etc/lmhosts` as the default:

```
/usr/local/samba/bin/nmbd -H /etc/lmhosts -D
```

As a general rule, `lmhosts` files should be avoided whenever possible due to the costs of keeping the contents current. It is much easier to rely upon a centralized NetBIOS name service such as WINS or even broadcast based name registration and resolution than it is to constantly roll out new versions of `lmhosts` to clients.

The `name resolve order` Parameter

The function of the global `name resolve order` parameter is analogous to the `/etc/nsswitch.conf` file on Name Service Switch–enabled systems. It controls the order and means by which Samba attempts to resolve NetBIOS names. However, it should be noted very strongly that this parameter affects only Samba's own name resolution queries, not those received from others in the case of Samba acting as a WINS server. Also, this lookup order is for name resolution only and does not affect name registration.

The parameter `name resolve order` accepts any combination of four possible values:

- `lmhosts`—The Samba `lmhosts` file is searched for a match to the requested name.
- `host`—Samba is instructed to perform a standard mapping of hostname to IP address using whatever means are available on the system, such as `/etc/hosts` lookups, DNS queries, or NIS/NIS+ matches. Be aware that this method is used only if the NetBIOS name being resolved has the server resource tag, `<20>`.
- `wins`—If a WINS server is specified by the `wins server` or `wins support` parameters, Samba attempts to resolve the NetBIOS name by querying the WINS server.
- `bcast`—Samba performs normal NetBIOS broadcast name resolution, which requires that the host in question be on the same broadcast subnet.

The default lookup scheme is

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

With the setting

```
name resolve order = lmhosts wins
```

Samba itself does not broadcast name queries and does not attempt to resolve the name via DNS. It only searches the local `lmhosts` file and then queries the configured WINS server (that is, the WINS server specified by the `wins server` parameter or itself in the case of `wins support = yes`).

Name Resolution Client or Server?

Here are a few final notes about Samba's use of an `lmhosts` file and name resolution in general. The contents of the `lmhosts` file affect name resolution for Samba internally. Samba does not use any entries to resolve requests that it receives from other hosts, nor does it currently use the contents of the `lmhosts` file to prime, or preload, its WINS database in the presence of `wins support = yes`.

The relation between the `lmhosts` file and WINS is analogous to the relation between `/etc/hosts` and DNS on Unix hosts. A DNS server does not lookup hostnames in `/etc/hosts` when it receives a name resolution request. It only uses its existing DNS zone files or another DNS server. Neither does a Samba WINS server look up a machine name or group name in the local `lmhosts` file when it receives a NetBIOS name resolution request.

Samba maintains a distinction between whether it is resolving a name for its own use and whether it is looking up the address for another client (that is, acting as a WINS server). The `wins server` and `name resolve order` parameters and the `lmhosts` file are only used when Samba is looking up names for its own use. The `wins support`, `wins proxy`, and `dns proxy` options control how Samba responds to name resolution requests received from other hosts.

WINS and Windows 2000

Microsoft has changed many of the services used by Windows 2000 clients from those used in past versions of Windows NT. These are two of the differences introduced by Windows 2000 that concern Samba:

- The use of DNS for all name services
- The ability to communicate via CIFS directly over TCP port 445 without establishing a NetBIOS session

In combination, these two changes mean that the NetBIOS layer is unneeded in a purely Windows 2000 environment. In practice, I know of very few homogeneous Windows 2000 networks. In Hour 14, "Replacing a Windows NT File and Print Server—Lock, Stock & Barrel," I mentioned that Samba 2.2 also requires NetBIOS support. This means that as long as non–Windows 2000 hosts exist on your network, WINS and NetBIOS are still present and required.

Summary

The NetBIOS name service is foundational to operations on many CIFS clients and servers. The Windows Internet Name Service (WINS) allows management of NetBIOS names across multiple IP subnets. By registering all client names with a single WINS server, it is possible to implement features such as cross-subnet browsing (see Hour 20, "Cross Subnet Browsing") and domain logons across a router (see Hours 21 and 22). If NetBIOS clients exist on multiple subnets or if you want to decrease the number of broadcast packets related to name registration and resolution, you should install a WINS server on your network.

Samba is able to operate as a WINS server (`wins support = yes`) or as a WINS client (`wins server = <ip address>`). The methods used by Samba to resolve NetBIOS names for its own use can be customized by defining the `name resolve order` parameter and tweaking the local `lmhosts` file. Samba also can forward name resolution requests for other NetBIOS client by enabling the `wins proxy` `smb.conf` option or even query DNS for the name (`dns proxy = yes`) if `nmbd` has been configured as a WINS server.

Q&A

Q Can NetBIOS clients still resolve names if the WINS server is unavailable because of a crash?

A If the client is configured as an H-node, as most Microsoft clients that register using WINS are, the client resorts to using broadcast queries to register and resolve names if it cannot contact the WINS server. Only if the name being resolved is owned by a machine on the logical subnet (that is, can be reached via broadcast mechanisms) does the client get a positive response. Refer to Hour 2 for complete explanations of the different NetBIOS node types.

Q Can I run multiple WINS servers at the same time?

A Yes, it is technically possible. The NetBIOS name space, however, remains segmented. Only clients registered with the same WINS server are able to see each other across routers.

Q How does WINS relate to dynamic DNS?

A WINS and DNS, dynamic or not, are two entirely separate beasts. Dynamic DNS (DDNS) is essentially an implementation of DNS that allows automatic updating of zones. Microsoft's DNS server can query its WINS server if a DNS lookup fails, but the two are still different services. DNS deals with IP names; WINS handles NetBIOS names.

New Terms

`lmhosts` An ASCII file that maps IP addresses to NetBIOS names. This is the NetBIOS functional equivalent of Unix's `/etc/hosts` file.

Windows Internet Name Service (WINS) Microsoft's implementation of an RFC 1001/1002–compliant NetBIOS name server (NBNS).



HOUR 19

Local Subnet Browsing

Network browsing is fundamental to Microsoft's peer networking model discussed in Hour 2, "Windows Networking Concepts." In a nutshell, network browsing is designed to help users locate resources, such as files and printers, in an ever-changing environment where machines can come on and go off the network on a whim. This is particularly true in small offices where individuals use desktop machines to provide remote access to their personal files or local printer.

There are three things to remember about browsing:

1. Browsing is complicated.
2. Browsing is hard to get right.
3. End users tend to like network browsing.

In this hour, we will cover how to configure Samba to participate in network browsing with other CIFS clients on its local network. In Hour 20, "Cross Subnet Browsing," we will expand this model to discuss how to configure Samba's browsing in broader environments that exist across more than one IP subnet.

Introduction to Browsing

Windows clients browse the network to find servers and services that they need to use. Under Windows 9x and Windows NT, this is achieved using Network Neighborhood. The hosts that help to coordinate the location of services are referred to as *browse servers*. These browse servers can be systems running Windows NT Server, Windows NT Workstation, Windows 9x, Windows for Workgroups, or Samba (and possibly ASU-based servers).

A *browse list* is a list of all servers in the master browser's workgroup or domain and all domains on that network. The browse list is built by the master browser over time as it listens to and collects the *server announcements* and *domain announcements* made by all servers and domain master browsers (explained in the next hour).

A *server* is defined as a machine that has the ability to share resources using the CIFS protocol. The host does not have to be currently exporting any shared directories or printers; simply possessing the capability of sharing them is enough. This means all Windows NT 4.0/2000/XP clients are considered servers by default because of the `Server` service. It also means any Windows 9x/ME host with the "File and Printer Sharing for Microsoft Networks" service installed is considered to be a server.



When discussing network browsing, a domain and a workgroup are logical equivalents. For that reason, I will use the term *workgroup* exclusively in our discussion.

Figure 19.1 shows the results of browsing a network containing the ROME workgroup along with another workgroup called GLASS. When users click one of the domains or workgroups, they are shown all the servers in that domain or workgroup. The Network Neighborhood shown in Figure 19.2 displays a list of servers in the ROME workgroup.

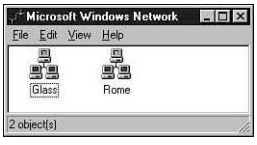


FIGURE 19.1
Displaying multiple workgroups in the network neighborhood from a Windows NT 4.0 client.

FIGURE 19.2
Using the network neighborhood to display the servers in the ROME workgroup.



In order to provide redundancy and distribute the browsing load across multiple systems, a Windows network will have several *backup browsers* as well as a master browser. Samba can operate as either one of these depending on the values of various browsing parameters in the `smb.conf` file.

When a Windows client browses the network, it follows these steps

1. Resolves the NetBIOS name of the workgroup's master browser (for example, `ROME<1d>`).
2. Contacts the local master browser located in Step 1 and sends a `QueryBrowserServers` request to obtain the list of backup browser servers for the workgroup.
3. Chooses a random server from the list received and sends a `NetServerEnum2` (or `NetServerEnum`) call to that server asking for the browse list.



Browsing in homogenous Windows 2000 networks is much different. When browsing a Windows 2000 network (and not using NetBIOS), the client locates services by locating an Active Directory server via DNS lookups. The list of resources is then retrieved by using LDAP to search the directory service. Samba is currently unable to publish its list of shares in Active Directory, and therefore will not be found using this method.

By using this browsing architecture, Windows obtains two important benefits:

- Clients do not have to devote CPU and memory resources to processing server announcements and maintaining browse lists.
- Clients are not required to build up browse lists slowly as they see each announcement. They may contact a browse server, which most likely has been up for considerably longer than the client, to obtain the list of machines.

In addition to maintaining the browse list for a workgroup, the master browse server (also known as the master browser) maintains a list of all the browse servers (or backup browsers) in the local subnet. This list is provided to clients when they issue a `QueryBrowserServers` request as part of Step 2.

A backup browser contacts the master browser every 15 minutes to obtain the latest copy of the browse list. It caches this list of servers and domains and uses it to respond to `NetServerEnum[2]` requests from browse clients.

The question that has not been answered yet is how a machine becomes a master browser. The process of choosing a host to be the master browser for a workgroup is known as the *browsing election* and can occur at any time. Here are some common scenarios:

- If a backup browser cannot contact the master browser, it will force an election.
- A Windows NT server will initiate an election when it boots on the network because it is a *preferred master browser* by default.

Contained in each election request is a 4-byte field called the *election criteria*. The structure of the `election criteria` field is shown in Figure 19.3 along with the fields you can control.

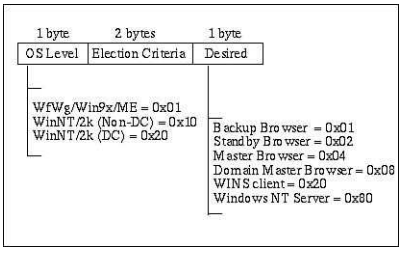


FIGURE 19.3
Contents of the `election criteria` *field in an election request.*

When the current master browser receives an election request, it examines values in the OS Level field as well as Election Revision and Election Summary to determine what to do. You can control Samba's behavior with regard to browsing elections by setting the `os level` parameter along with the other browsing parameters, which will be covered shortly.

A potential master browser enters the running election state once it determines that its own OS Level value is higher than that of any other candidate. While in this state, it sends up to four more election requests at intervals of 200ms or 400ms, depending on whether it is a master browser or a backup browser, respectively. If a browser wins each election four times in a row, it becomes the master browser for that workgroup or domain. If its OS Level value matches that of another browser in the network, the other fields in the criteria are checked to determine who should win the election. If a current master browser receives an election request that indicates another system will win the election, it will demote itself to a backup browser.

Once a system becomes the master browser, it immediately broadcasts a `RequestAnnouncement` request. Systems on the local subnet respond to the `RequestAnnouncement` request by sending a `ServerAnnouncement` after a random time within a 30-second interval. This helps to reduce the load on the master browser.

Samba Browsing Parameters

The behavior of Samba with respect to browsing is controlled by a number of `smb.conf` parameters. These parameters affect such things as

- Whether Samba provides browsing services at all
- Whether Samba seeks to become a master browser or a backup browser
- Which services show up when a Samba server is being browsed by a client Windows system

To begin, let's look at those parameters that directly affect the browsing election process and the Election Criteria field, which was mentioned previously in this hour. Samba has the ability to initiate a browsing election upon starting the `nmbd` daemon simply by enabling the `preferred master` parameter. A Windows NT/2000 server exhibits the same behavior. Normally, this option is set to `auto`, which means it will be enabled only if both the `local master` and the `domain master` parameters are set to `yes`. The `preferred master` parameter can be manually enabled by adding

```
preferred master = yes
```

to the `[global]` section of your `smb.conf`

Once a browsing election has begun, the `local master` parameter controls whether or not `nmbd` should participate in the elections. Do not confuse this with the `domain master` parameter, which will be discussed in Hour 20, "Cross Subnet Browsing."

By default, Samba will act as a local master browser for its workgroup. I have known circumstances where people deployed Samba servers for the sole purpose of stabilizing the Network Neighborhood. When the local master browser frequently changes from one machine to another because of hosts rapidly coming on and off the network, Windows clients can become confused and unable to consistently locate the workgroup's master browser. By placing a Samba server on the network, we can ensure that our particular local master browser remains present for extended periods of time. This helps to minimize browsing problems caused by clients querying servers that are no longer available in order to obtain a copy the workgroup's contents.

However, there are some circumstances in which adding `local master = no` to the `[global]` section of `smb.conf` is preferable so Samba will not disrupt an existing

browsing network. In the Hour 20, we will look at some common browsing environments and how this setting can affect large Windows networks. For most small networks, this option is almost always left enabled.

Enabling the `local master` parameter does not ensure Samba will win the browsing election. This is determined by the election criteria. The upper 8-bits of this field is defined as the `os level`. Therefore, by assigning a sufficiently high `os level` to Samba, we can ensure our server will become the local master browser.

Because the `os level` in the election criteria is 8-bits in length, the `os level smb.conf` parameter accepts an integer from 0 to 255. Table 19.1 displays the `os level` values of popular Microsoft operating systems.

TABLE 19.1 The `os level` Values for Versions of Microsoft Windows

OS Name	`os level` *Value*
Windows for Workgroups 3.11 Windows 95 Windows 98 Windows ME	1
Windows NT 4.0 (non-DC) Windows 2000 (non-DC)	16
Windows NT 4.0 (PDC/BDC) Windows 2000 Server (DC)	32

Samba 2.2 has a default `os level` of 20, which means that out of the box it will win an election against everything but a Windows NT 4.0/2000 domain controller. Beware of default values.

Browsing with OS/2 Clients

In addition to these basic browsing options, Samba's `smb.conf` also supports other parameters that affect how Samba presents itself to others in the network neighborhood. If you support OS/2 clients on your network, it may be necessary to tweak Samba's `lm announce` and `lm interval` parameters. Both options are used in the `[global]` section of `smb.conf`.

The `lm announce` parameter specifies whether Samba (`nmbd`) produces LanManager-style host announcements, such as those needed by OS/2 and other clients. These are necessary in order for Samba to appear in their browse lists.

The legal values for this parameter are

`yes`	Send `LanMan`-style host announcements at the frequency set by the `lm interval` parameter.
`no`	Never send `LanMan`-style host announcements.
`auto`	Do not send `LanMan`-style host announcements until one is seen on the network. After that point, send them at the frequency set by the `lm interval` parameter.

The `lm interval` parameter is used in conjunction with `lm announce` and specifies the frequency in seconds with which LanMan-style host announcements will be made. This option is applicable only if the `lm announce` parameter is set to `yes` or `auto`. By default, `nmbd` sets this period to be every 60 seconds. If this parameter is set to 0, then no LanMan host announcements are made, despite the setting of the `lm announce` parameter.

Viewing a List of Shares

Once the browsing election has been decided and clients start enumerating servers within a workgroup, we drill down one more level and contact the CIFS server directly to view the list of shares it holds. Samba provides a handful of options for customizing the shares that should be displayed.

The `auto services` (a.k.a. `preload`) and `load printers` Parameters

These global parameters provide a primer for the list of services automatically added to the browse list. The `auto services` parameter accepts a discrete list of share names, which will be added to the browse list. The share names are not required to be explicitly declared in `smb.conf`. This option is most useful for adding specific homes and printers services that would not normally be visible because they are created as needed from `/etc/passwd` or the `printcap name` file. For example

```
auto services = jerry lp postscript
```

would specify to Samba (`nmbd`) that the services `jerry`, `lp`, and `postscript` should be made available for browsing. Please note, however, that this affects only what people can see when they click the icon for your server in Network Neighborhood. These services may or may not be valid for a given user depending on the server's configuration.

The `load printers` Boolean parameter performs a function similar to `auto services` by parsing the defined `printcap name` and creating shares for all the printers located

there. Of course, this makes sense only if you have also defined a `[printers]` section in `smb.conf`. If `load printers = no`, users are still able to access printers by using the UNC path to the device (for example, `\\server\printer1`), but this requires that the name and network location of the printer is known by a user beforehand.



Using `load printers = yes` with a large `printcap` file (for example, 200 printers) can result in an increase in memory usage by `smbd`.

Browsing Examples

So far, we have looked at a technical overview of network browsing and the various `smb.conf` browsing parameters available. Now, it is time to look at a working Samba server and see how it interacts with other browsing clients. Our first scenario uses a server named BRUTUS to win the browsing elections against all Windows clients in the ROME workgroup.

```
[global]
   netbios name = BRUTUS
   workgroup = ROME
   guest account = nobody
   os level = 33
   local master = yes
   preferred master = yes
```

In fact, this is the configuration file used for the Samba server shown in Figure 19.2. In order to verify that our configuration is working correctly and that BRUTUS has in fact become the master browser for ROME, we will use Samba's `smbclient` and `nmblookup` to build our own Network Neighborhood from a command line.



There are several Network Neighborhood implements for X Windows that are just graphical wrappers around `nmblookup` and `smbclient`. The LinNeighborhood project, `http://www.bnro.de/~schmidjo/`, is a popular GTK+ based Network Neighborhood client for Unix. KDE's Konqueror, `http://www.kde.org/`, also has the capability to connect to CIFS shares using an `smb://` style URL.

First, we must determine the IP address of the master browser. This can be done by using the `nmblookup` parameter's `-M` option. The command should look like

```
$ nmblookup -M ROME
querying ROME on 192.168.215.255
192.168.215.1 ROME<1d>
```

The IP addresses returned on your system will be different, but the key point is that the command returned an address for the ROME<1d> NetBIOS name. The <1d> resource byte indicates that this is the address of the workgroup's master browser.

We can now take this address and run nmblookup again. This time, we will perform a node status request (using the -S option) on the address (specified with the -A option) to obtain a list of NetBIOS names registered by the client.

```
$ nmblookup -S -A 192.168.215.1
Looking up status of 192.168.215.1
        BRUTUS          <00> -         B <ACTIVE>
        BRUTUS          <03> -         B <ACTIVE>
        BRUTUS          <20> -         B <ACTIVE>
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>
        ROME            <00> - <GROUP> B <ACTIVE>
        ROME            <1b> -         B <ACTIVE>
        ROME            <1d> -         B <ACTIVE>
        ROME            <1e> - <GROUP> B <ACTIVE>
```

Once again, we see the ROME<1d> name as well as several other group names. The ROME<1b> name in particular will be covered in Hour 20, when we explore how browsing works in larger environments.

We now have the NetBIOS name of the master browser, BRUTUS, and can also query it for a list of workgroups and servers that it has knowledge of as well. The smbclient parameter's -L option instructs it to enumerate the shares and browsing information held by the server, and the -N switch uses an anonymous logon (no username or password) to do so.

```
$ bin/smbclient -L BRUTUS -N
added interface ip=192.168.215.1 bcast=192.168.215.255 nmask=255.255.255.0
Anonymous login successful
Domain=[ROME] OS=[Unix] Server=[Samba 2.2.2]

        Sharename      Type      Comment
        ---------      ----      -------
        IPC$           IPC       IPC Service (Samba 2.2.2)
        ADMIN$         Disk      IPC Service (Samba 2.2.2)

        Server               Comment
        ---------            -------
        BRUTUS               Samba 2.2.2
        CAESAR
        POGO-98              Windows 98 client

        Workgroup            Master
        ---------            -------
        GLASS                CELEBORN
        ROME                 BRUTUS
```

Currently, we can see that BRUTUS has no file or printer shares configured. Only the IPC$ and ADMIN$ services, which are automatically added by Samba 2.2, appear in the share list. We also can see that BRUTUS knows about an additional workgroup named GLASS. We could use `nmblookup` and `smbclient` to map this server as well and continue expanding our view of the Network Neighborhood. That, as they say, is left as an exercise for the reader.

Browsing Problems

Most browsing problems can be resolved by carefully setting up the relevant parameters in your `smb.conf` file. However, when master browsers go down and elections occur, you might see messages such as `The list of servers for this workgroup is not currently available`. It is also possible to see large parts of the network disappear if there are not enough backup browsers on the network. We have seen how `nmblookup` can be used to investigate which nodes are in the workgroup/domain and which node is the master browser, if any.

Some browsing-related problems to watch out for include the following:

- Not having a guest account configured on your system or not having a `guest account` entry in the `smb.conf` file. Either of these can result in clients being unable to browse servers to find out the list of services available.
- Windows NT clients requiring a password to browse a Samba server.
- Large parts of the network disappearing and reappearing at random. This is caused by excessive elections in the network. Two browsers are fighting to become the master browse server.

Finally, Samba keeps several important files that are useful for tracking down browsing problems. Both files are stored in Samba's `lock directory`.

`browse.dat`	This file contains the browse list. It consists of one line per server in the browse list.
`wins.dat`	This file contains all the entries that `nmbd` keeps in its WINS database.

Summary

This hour explores local subnet browsing in some depth and looks at all the `smb.conf` parameters that affect browsing in any way. CIFS clients are able to obtain the list of servers within a workgroup by querying the workgroup's master browser. A master

browser is chosen through an election process. The outcome of this browsing election is determined by the value of each candidate's election criteria. The host with the highest election criteria is declared as the winner.

If browsing elections occur too often due to hosts frequently coming on and off the network, Windows clients can become confused and be unable to locate a master browser. The result is that parts of the Network Neighborhood can appear empty or inaccessible.

You should now be equipped to resolve most issues surrounding browsing in a Windows network by using `smbclient` and `nmblookup` as diagnostic tools. In Hour 20, we will explore browsing in routed networks and how Samba can be configured to synchronize browsing information across subnets.

Q&A

Q **We can't browse our Samba server. That is, none of its shares show up when we click the server's icon. How can we fix this?**

A This might be because the shares have not been marked as not browsable. Check your `smb.conf` for `browsable = no` in the `[global]` section or within individual service definitions. Samba 2.2 will mark all shares as `browsable` by default.

Q **We are planning to move our Samba server to another machine, but we would like to enable our users to see the new machine in their browse lists as the current server, which is called `HOBBIT`.**

A You need to have the new server advertise a NetBIOS alias or give the NetBIOS name of the old server. It is sometimes worthwhile to give the Samba server a NetBIOS name that relates to the service it performs rather than using the default DNS name. You can give a Samba server a NetBIOS name with the following:

```
netbios name = hobbit
```

You can give a Samba server a NetBIOS alias in the following way:

```
netbios aliases = hobbit
```

Q **How can I make sure that my Samba server becomes the master browser in our workgroup? Sometimes we find that a Windows 95 PC becomes the master browser, which is not so good.**

A To ensure that Samba becomes the master browser, you need to make sure it wins browsing elections. To do this, give it an `os level` that is higher than any other potential browse server's `os level` on the network—for example, `33`. You can do this with the following in the global section of your `smb.conf` file:

```
os level = 33
```

Q We have a lot of printers defined on our Samba server in the `printcap` file. These all show up in the browse lists when users browse the server, but they are of little use to users because they mostly do not have drivers for the printers. We have defined the printers that are of interest to PC users in our `smb.conf` file. How do we stop Samba from displaying all these unnecessary printers?

A By default, Samba loads all printers in your `printcap` file. To switch this behavior off, simply add the following to your `smb.conf` file in the `[global]` section:

```
load printers = no
```

New Terms

backup browser A host that maintains a copy of the workgroup's browse list for the local master browser.

browse list A list of the current servers available within a particular workgroup.

browse server The host(s) that maintains a browse list.

browsing election The process by which a local master browser is selected on a network.

election criteria The 16-bit field in an election packet that is used to determine the local master browser during a browsing election.

local master browser The host that maintains the master copy of the browse list for a local network.

preferred master A field within the election criteria that can be used to give a potential master browser a slight favorable edge during the election process.

server A host that is able to share resources such as files and printers via CIFS.

server announcement The packet(s) transmitted by a server while it is active on the network. These are collected into the browse list by the local master browser.



HOUR 20

Cross Subnet Browsing

Samba supports browsing on a single subnet as well as across routed networks. In the previous hour, we explored browsing in one broadcast area (that is, one subnet) in some detail. This hour covers the details of configuring Samba to support browsing when the workgroup members are located on opposite sides of a router or cannot otherwise be reached by broadcast network requests.

In order to begin our discussion, we will need to come to terms with the differences between local browsing and browsing across routers. We will follow this with details on how Samba should be configured in these latter types of environments as well as how to troubleshoot problems when browsing does not work quite as expected.

Browsing Across Subnets

First, we will build upon the concepts of browse elections and local master browsers presented in the previous hour in order to understand how we can browse the contents of a workgroup across networks and TCP/IP routers.

You should recall that a CIFS server broadcasts host announcements to inform local master browsers of its presence. In theory, this allows for a designated host to generate a complete list of servers within its network.

The word "network" in this context refers to the TCP/IP concept of a network that is defined by the network mask. All hosts configured with the same network address (for example, `192.168.1.0`) and the same network mask (for example, `255.255.255.0`) will send broadcast packets to the same address. In the example here, this address would be `192.168.1.255`.



Sams' Teach Yourself TCP/IP in 24 Hours, 2nd edition by Joe Casad or *Sams' TCP/IP Primer Plus* provide good introductions to TCP/IP network concepts if you are feeling a little lost here.

Hosts existing on different networks cannot communicate with one another using broadcast mechanisms because these packets are normally (and for a good reason) stopped by the TCP/IP router connecting the networks. In this case, hosts must rely on packets that can be sent directly to a specific host.

Hour 13 showed that being able to correctly resolve network addresses for NetBIOS hosts and communicate with them is fundamental to network browsing. You should recall from Hour 18 that a WINS server allows hosts on different networks to register and resolve NetBIOS names with a centralized server. Once a name can be resolved, a request can be sent directly to that host.

Last hour, the process of browsing a workgroup was summed up by the following steps:

1. The client resolves the NetBIOS name of the workgroup's master browser (for example, ROME<1d>).
2. The client then contacts the local master browser located in Step #1 and sends a `QueryBrowserServers` request to obtain the list of backup browser servers for the workgroup or domain.
3. Finally, the client chooses a random server from the list received and sends a `NetServerEnum2` (or `NetServerEnum`) call to that server asking for the browse list.

This implies that if NetBIOS names cannot be resolved correctly, workgroup browsing will not function correctly, either. In order for browsing to function across routers, the NetBIOS name resolution mechanisms used must support the location of hosts on other networks. In other words, it is a requirement that all NetBIOS clients and servers use the same WINS server for registering and resolving names.

In order to coordinate a workgroup browse list when hosts are connected on different sides of a router, one of the servers is designated as the *Domain Master Browser* (DMB). This host is not elected, as in the case with local master browsers. It is designated using a configuration option. A Windows NT 4.0 PDC attempts to register this name automatically for its domain (for example, `ROME<1b>`) and must succeed or else it will complain that another PDC exists on the network and fail to start properly.



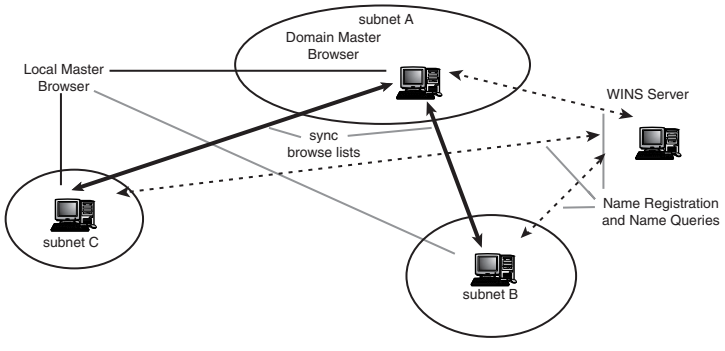
As we will see shortly, it is possible to configure a Samba server as the domain master browser for a workgroup. If the server succeeds in registering the `<1b>` unique NetBIOS name for its workgroup, the real Windows NT PDC will fail to start after the next reboot. This is not a security problem with Samba, but rather a design flaw with the NetBIOS name registration protocols.

The job of the DMB is to collect browse lists from the workgroup's various local master browsers (LMB) and merge these into one list. The workgroup's local master browsers will periodically (every 15 minutes or so) retrieve this merged copy of the browse list from the DMB. Over time, this list will contain a list of all servers within the workgroup (as reported by the LMBs).

Figure 20.1 gives a pictorial overview of how browsing works in a cross subnet environment. An LMB exists on each subnet (A, B, and C). The pictured host on subnet A is also acting as the DMB. However, there is no requirement that the DMB also act as the LMB for its own subnet. This is simply a common configuration.

The local master browsers on subnets B and C synchronize their browse lists with the domain master browser and therefore gain knowledge about servers that exist beyond their own network. The WINS server in the diagram is an underlying requirement for this to work properly.

FIGURE 20.1
Synchronizing browse lists between local master browsers and the domain master browser.



The `domain master` Parameter

The domain master global parameter is disabled by default. This means that Samba will not attempt to register the `WORKGROUP<1b>` NetBIOS name. In most circumstances, this is a good thing, as it prevents Samba from inadvertently breaking existing Windows domain controllers. If enabled by adding

```
domain master = yes
```

to `smb.conf`, `nmbd` will attempt to register the DMB name for its workgroup and `smbd` will begin collating domain-wide browse lists (or workgroup-wide browse lists if you have a workgroup-spanning subnet).

Once `nmbd` has registered the `<1b>` name and can be located as the DMB by local master browsers in the same workgroup, they can request a copy of the DMB's browser list and send their own browse list. In this way, Figure 20.1 can be brought to pass.

Designing Network Browsing

With any luck, you now have an understanding of how local network and cross-subnet browsing works. In this section we will design a browsing infrastructure and examine the Samba browsing configuration files necessary to implement it. Our scenario assumes the following facts:

- We are using a workgroup peer-networking model (see Hour 2, "Windows Networking Concepts," for a refresher on peer-networking and workgroup security models).
- Our workgroup name is `STY-SAMBA`.
- The NetBIOS clients are located in the following networks: `192.168.110.0/255.255.255.0`, `192.168.120.0/255.255.255.0`, and `192.168.121.0/255.255.255.0`. These subnet masks can be written as simply '`24`' to represent the number of leading ones in the netmask.
- The machine at IP address `192.168.2.10` is acting as a WINS server.
- The network clients are composed of a mixture of Windows 98, ME, NT, 4.0, and Windows 2000 hosts.

We will configure a Samba host as the domain master browser of the `STY-SAMBA` workgroup. Therefore, our initial `smb.conf` for this machine will look like

```
##
## smb.conf file for the workgroup domain master browser
## on 192.168.110.0/24
##
[global]
```

```
; basic server settings
workgroup = STY-SAMBA

; browsing election parameters for local master browser
os level = 64
local master = yes
preferred master = yes

; configure this host as the DMB
domain master = yes
```

This looks acceptable as a starting point. However, remember that all clients must register their NetBIOS names with the same WINS server. This means that we must add the following line to our newly created configuration file.

```
; set the IP address of the server to use for
; name registration and resolution
wins server = 192.168.2.10

; always look up hosts in wins first. Do not use
; DNS for looking up NAME<20> names
name resolve order = wins lmhosts bcast
```

Of course, the Samba host could act as the WINS server for our network. We choose not to do this because a WINS server already exists and Samba has no means to synchronize its local WINS database with another WINS server (Samba or Windows NT). Under these circumstances, configuring Samba as a client of the current WINS server makes the most sense.

The next step is to ensure also that all other NetBIOS clients (Windows and Samba) use the same WINS server as well. At this point we have a network like the one shown in Figure 20.2.

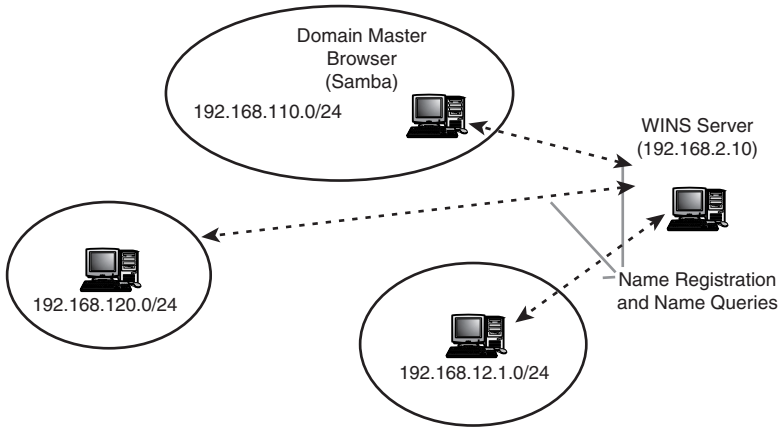


FIGURE 20.2
Stage one is designing our network-browsing infrastructure.

Now we must configure local master browsers on each of the various subnets. Perhaps you have noticed that our Samba DMB has also been configured to act as the LMB for that subnet. The only remaining networks without a local master browser for our workgroup are `192.168.120.0/24` and `192.168.121.0/24`. For the latter network, we will allow an existing Windows 2000 client to perform that role.

However, there are no clients on the `192.168.120.0` network that are deemed appropriate or stable enough to act as the LMB. We will configure another Samba host as the browse master for this subnet because it will be able to remain on the network for long periods without rebooting. By doing so, the new Samba server will help to stabilize the browse election process. This time, however, we will be very careful not to enable the `domain master` parameter. The resulting `smb.conf` is given here.

```
##
## smb.conf for a local master browser for 192.168.120.0/24
##
 [global]
   ; basic server settings
   workgroup = STY-SAMBA

   ; browsing election parameters for local master browser
   os level = 64
   local master = yes
   preferred master = yes

   ; **do not** enable domain master = yes
   domain master = no

   ; set the IP address of the server to use for
   ; name registration and resolution
   wins server = 192.168.2.10

   ; always look up hosts in wins first. Do not use
   ; DNS for looking up NAME<20> names
   name resolve order = wins lmhosts bcast
```



Remember that there should be only one domain master browser per workgroup. Our two Samba servers do not battle for the position of local master browser because each is isolated from the other by the router connecting the two networks.

Our final browsing design (Figure 20.3) appears very similar to Figure 20.1. It is, in fact, the same picture with a few lines of text replaced.

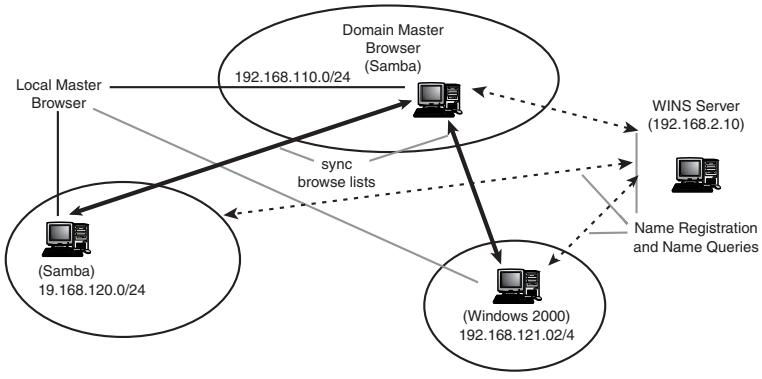


FIGURE 20.3
Completed network browsing infrastructure design.

For the closing comments of this section, we will follow the steps necessary for a machine on network `192.168.120.0/24` named `LOST` to locate a server on network `192.168.121.0/24` named `FOUND`.

1. The process begins when `FOUND` appears on the network and issues a server announcement.
2. The Windows 2000 host that is acting as the local master browser for the network receives the announcement.
3. The next time the Windows 2000 LMB attempts to synchronize its browse list with the DMB (which it locates by querying the WINS server for the name `STY-SAMBA<1b>`), it will propagate the updated browse list containing the server `FOUND`.
4. The Samba LMB on network `192.168.120.0/24` receives the updated browse list from the `STY-SAMBA` DMB (also located by querying WINS). This browse list is then propagated to the various Backup master browsers on the local network.
5. At this point, a user logged onto client `LOST` clicks on the Network Neighborhood. The server `FOUND` appears in the `STY-SAMBA` workgroup.
6. The user then selects `FOUND` to view what shares are available on that server.
7. `LOST` issues a WINS query to retrieve the IP address of the server `FOUND<20>`.
8. Once an IP address is known, `LOST` can issue a call to `FOUND` directly to gain a list of the available shares.

Simple right? The problem with browsing, especially cross subnet browsing, is that it tends to be extremely fragile. The next section focuses on some common problems and solutions for broken Network Neighborhoods.

Troubleshooting Remote Browsing

In debugging problems with browsing, remember that browse lists contain only names—the names of the servers that have sent host announcements in the workgroup or domain and the names of the domains whose announcements have been seen by the master browser. This means that it is very important that WINS and NetBIOS Name Services are functioning correctly in your network.

If you can see all the servers in Network Neighborhood but you cannot obtain a list of services on a particular server, you should ensure that the WINS servers are functioning and your clients can access the WINS server. If the client cannot resolve the CIFS server's name correctly, you will see errors like

```
\\Server is not accessible.
The network path was not found.
```

A few other common problems include being unable to see any nodes in the Network Neighborhood or being unable to see nodes that were there before. This problem is due to browsers changing roles and having frequent elections. It is a good idea to verify that Samba or another host is not frequently forcing elections to occur. One way to check this is to look at the `log.nmbd` file in the Samba log directory.

Another common problem is that a user may be prompted for a password in order to browse a server. This problem is usually due to the user not having an account on the machine that he is trying to use, or the user's password on the remote server machine being different than the password which was used to log on to the client's console or domain. Refer to Hour 7, "Security Levels and Passwords," for details on user accounts in Samba.

Just as we used `smbclient` and `nmblookup` in the previous hour to debug browsing problems on a single network, these same tools can be used to debug cross subnet browsing issues. The `nmblookup` tool can be used to query a WINS server directly, much like `nslookup` or `dig` can be used to query DNS. Querying a WINS server can be done by using a combination of the `-U` (unicast the request to a specific address) and `-R` (set the recursion bit in the request packet) flags. Here is how we could resolve the DMB for the `STY-SAMBA` workgroup using a WINS server at 192.168.2.10.

```
$ nmblookup -U 192.168.2.10 -R 'STY-SAMBA#1b'
querying STY-SAMBA on 192.168.2.10
192.168.110.45 STY-SAMBA<1b>
```

Once we have the address of the DMB, we can use `smbclient` to enumerate the browse list on the server just as we did in the previous hour (`smbclient -L <server>`).

We can also use `nmblookup` to specifically search for the local master on a network by using a directed broadcast. Remember that the LMB name (for example, `STY-SAMBA<1d>`) is not registered with WINS because it is by definition local to a single subnet. In order to locate the LMB for our `192.168.120.0/24` (assuming that is where our current client is located), the following command should return the address of the local browse master.

```
$ nmblookup -B 192.168.120.255 'STY-SAMBA#1d'
querying STY-SAMBA on 192.168.120.255
192.168.120.233 STY-SAMBA<1d>
```

Additional Browsing Parameters

There are some additional browsing-related parameters supported by Samba. These tend to be workarounds for specific problems or are not applicable in all environments. They have been placed at the end of our discussion to avoid confusion about how browsing should operate.

The `remote announce` Parameter

The `remote announce` global parameter specifies that `nmbd` should periodically send host announcements to the specified IP addresses under the specified workgroup name. This allows us to configure a Samba server to appear in the browse lists of remote (subnet) workgroups where it normally would not if it were following the traditional Windows network browsing rules.

The following line instructs Samba to send host announcements, in addition to the ones sent to the local subnet, to the address `192.16.200.255` under the guise of the workgroup `SAMS`.

```
remote announce = 192.168.200.255/SAMS
```

Many routers do not allow *directed broadcasts* (that is, an IP datagram to the broadcast address of a subnet) to be forwarded by default. This is done to prevent networks from being flooded by unnecessary broadcast packets. Normally, enabling directed broadcasts should never be done. However, if the IP address of the local master browser is known, the `remote announce` option can be used to send a server announcement to that particular host address.



I generally discourage use of the `remote announce` parameter except in circumstances where there is no LMB for a workgroup, but the browse list must be known to hosts on the remote network.

The `remote browse sync` Parameter

This global parameter specifies that `nmbd` should periodically request synchronization of browse lists with the specified master browsers on remote subnets. The synchronization mechanism used works only with other Samba servers. However, it does provide for the synchronization of browse lists between Samba servers across subnets.

The next configuration line says that Samba requests the master browser of our workgroup on the remote network (`192.168.190.255`) to synchronize browse lists with us.

```
remote browse sync = 192.168.190.255
```

Of course, this example also requires that the router between the Samba host and the remote network allow directed broadcasts. If this were not the case, the `remote browse sync` value would need to specify the address of the remote LMB.



The same advice which was mentioned for the `remote announce` parameter can be applied to the `remote browse sync` option. You should only use it if you have a good, solid understanding of it. It should only be used in circumstances that cannot be solved using the normal browse list propagation rules described in the last hour and earlier in the chapter.

The `enhanced browsing` Parameter

The purpose of the enhanced browsing global parameters is to make multi-workgroup browsing easier by obtaining a list of all DMBs for all workgroups from the WINS server and then synchronizing browse lists with these hosts. The second function of this parameter is to cause `nmbd` to periodically synchronize with all known DMBs. However, this can result in empty workgroups remaining in the Network Neighborhood for an infinite amount of time.



This parameter, new to Samba 2.2, is useful only when Samba is using another Samba server as the WINS server. However, it does no harm when the WINS server is not running Samba, which is why it is enabled by default.

Summary

In this hour, we explored browsing mechanisms in a multi-network environment. Browsing across subnets is simply an extension of browsing on a local network. In fact, Microsoft's definition of a *workgroup* is limited to a single subnet. A workgroup spread

across multiple subnets is defined as several workgroups that happen to use the same group name. The network administrator designates a Domain Master Browser that coordinates the interaction between workgroups. There is no election to determine the DMB.

Network Browsing is separate from Domain Control, but as we will see in the next hour, clients locate a Domain Controller by searching for the DMB in its domain.

Q&A

Q We have a number of Samba servers in a network that already has a Windows NT domain with its own Primary Domain Controller. A number of our client machines seem to want to log in to the Samba server, and thus cannot log in to the domain. What is the problem and how do we fix it?

A This is probably caused by Samba being configured to operate as a Domain Master Browser. If it is up before the PDC is started, the Samba server has registered the name `DOMAIN<1B>` (where `DOMAIN` is replaced with your domain name). Remove any domain master parameters from the `smb.conf` file on your Samba server and restart `smbd` and `nmbd`. You may also need to restart your Primary Domain Controller.

Q Our workgroup is spread across several subnets, and we have Samba servers in each subnet. We are having problems with browsing. I have read that only a Windows NT domain supports browsing across subnets. Do I need to replace our main Samba server with a Windows NT PDC to provide browsing throughout the whole network?

A Although Microsoft Windows Networking does not support browsing across subnets in a workgroup, Samba does. You can configure Samba as a Domain Master Browser by setting `domain master = yes` in the `[global]` section of `smb.conf`, and ensure that all clients and servers in the workgroup are configured to use the same WINS server (or group of servers).

Q We can browse servers in remote subnets in our network, but whenever someone tries to browse the services provided on a server (Samba or otherwise), they get an error message such as "Network name not found," and it does not work. How can we fix this?

A This is usually caused because you do not have WINS set up correctly in the clients in the remote network. Browse lists provide only the NetBIOS names of servers and domains. Translating a NetBIOS name to an IP address in a routed network requires the use of WINS (or a `lmhosts` file, but WINS is better). In cases where you have clients that do not understand WINS, you need to set up a Samba server on the local subnet as a WINS proxy (see Hour 18 for information on the `wins proxy` parameter).



HOUR 21

Domain Control for Windows 95/98/ME

This hour we will turn our focus again to authentication and build upon the lessons we learned in Hour 7, "Security Levels and Passwords." Up to this point, the authentication step occurred as part of the `SMBsessetup&X` request and reply such as when a client connects to a specific server by executing

```
C:\WINDOWS\> net use \\pogo\share1
```

During this chapter, we will see how to configure Samba as a logon server for Windows 9x/ME clients. A logon server is a host that provides central authentication for a group of Windows 9x/ME clients. Using Samba in this capacity can be a cost-effective solution for central account management on a network without incurring the cost of Windows NT/2000 connection or access licenses. Behaving as a Windows 9x/ME logon server is slightly different than acting as a Domain Controller for Windows NT or Windows 2000 clients, which will be described in the next hour.

Setting Up the Samba Domain Controller

In Hour 2, "Windows Networking Concepts," the difference between a workgroup and a domain was first introduced. This was summed up as being a difference in authentication models. A workgroup relies upon each server to validate each connection request it receives. A domain uses a central server to authentication connects on behalf of domain members.



It may be a good idea to review Hour 2 if you need a refresher on domains.

In order to make the discussion of configuration settings less abstract, let's assume that we are trying to implement a solution meeting the following criteria:

- A single Samba server that authenticates logons to the console of Windows 95/98/ME clients.
- All clients exist on a single network (for example, `192.168.134.0/24`).

This means that we will not bother using WINS; instead, we will rely upon broadcast mechanisms for NetBIOS name registration and resolution. Next hour, we will examine how to configure a Domain Controller/Logon Server that will support a domain spread across multiple networks.

The first requirement for configuring Samba as a domain controller is to enable user level security in `smb.conf` by adding

```
security = user
```

to the `[global]` section. This is the default security level in versions of Samba that are later than 2.0.

For the sake of simplicity, assume for now that the logon server must be the master browser for the domain. We will see in the next hour how this does not necessarily hold true in the case of a Backup Domain Controller. But for the moment, this assumption makes things much easier to describe and prevents us from contradicting ourselves later. The following settings will ensure that Windows clients can identify our server as the logon server.

```
[global]
   <...>
   os level = 64
   domain master = yes
   local master = yes
   preferred master = yes
```

Next, Samba must be informed that it should service logon requests by adding

```
domain logons = yes
```

to the server's smb.conf `[global]` section.

Finally, we will create a file share in `smb.conf` named `[netlogon]`. All Windows 9x/ME clients that attempt to log in to a domain connect to this service to retrieve certain information, such as logon scripts and system policies. However, the `[netlogon]` share does not necessarily contain any data at all.

We will use the following file service definition for our server.

```
[netlogon]
       path = /export/smb/netlogon
       read only = yes
       public = no
```



It is very important that the `[netlogon]` share and all the files it contains cannot be modified by normal users. Because all Windows clients will query this share for certain information when logging into the domain, an insecure `[netlogon]` service can be used as a quick means to distribute viruses or Trojan programs designed to steal data from users.

There is one more item that is optional but is often considered to be indispensable. When a Windows 9x client successfully authenticates to a logon server (or domain controller), it can download and execute a batch file from the logon server. This logon script is stored in the `[netlogon]` share and is specified by the `logon script` parameter. This filename is stored as a DOS path relative to the `[netlogon]` share. It must be a path that is understood by the Windows client. For example, the following setting is an intuitive way of assigning a unique batch file to each user based on his or her login name.

```
logon script = %U.bat
```

The login script for a connection is set to a `username.bat`, where `username` is obtained from the session setup information. If user `beckett` logs in successfully, the file that the Windows client attempts to run would be

```
\\server\netlogon\beckett.bat
```

For the server built in this hour, we will keep things simple and use a single batch file for all users by defining the following setting in the `[global]` section of our `smb.conf`.

```
logon script = logon.bat
```

This batch file will contain a few simple commands that map the user's home directory to drive H:. We must also make sure that the batch file uses DOS-formatted text with `CR` and `LF` characters at the end of each line. This means that the batch file should be created using a DOS editor, such as notepad, or a Unix editor that can convert from Unix test to DOS text. Here is our `logon.bat` file:

```
@echo off
echo Executing logon script...
net use h: \\pogo\homes
```

For completeness, Listing 21.1 shows the entire `smb.conf` file you have set up for your server. We are using encrypted passwords, but Samba's logon-serving capabilities for Windows 9x/ME will also function using plain-text passwords. With either method, make sure that you have set up the user accounts correctly. Verify that if you are using plain-text passwords, you have enabled this capability on Windows 98 clients and the necessary Windows 95 clients (see Hour 10, "Microsoft's DOS Network Client and Windows 95/98/ME").

LISTING 21.1 Samba Configuration File for a Simple Windows 9x Domain Controller

```
##
##          Mon Aug 27 01:09:17 CDT 2001
##          jerry carter <jerry@samba.org>
##
##          Sams Teach Yourself Samba in 24 Hour
##          smb.conf for Windows 9x logon server

[global]
     netbios name = POGO
     workgroup = STY-SAMBA
     security = user
     encrypt passwords = yes

     domain logons = yes
     logon script = logon.bat

     os level = 64
     domain master = yes
     local master = yes
     preferred master = yes
```

continues

LISTING 21.1 Continued

```
[netlogon]
     path = /export/smb/netlogon
     read only = yes
     public = no

[homes]
     comment = Home directories for STY-SAMBA domain users
     create mask = 0600
     directory mask = 0700
     browseable = no
     valid users = %S
```

After verifying the syntax of our `smb.conf` with the `testparm` tool, we are ready to launch the `smbd` and `nmbd` daemons using our normal means.

Setting Up a Windows 9x Client

Now that we have configured and started the Samba server, the next step is to configure the Windows client to log in to the domain. Assuming you have already configured the Windows box to access SMB servers (as described in Hour 10), relatively few additional steps are necessary.

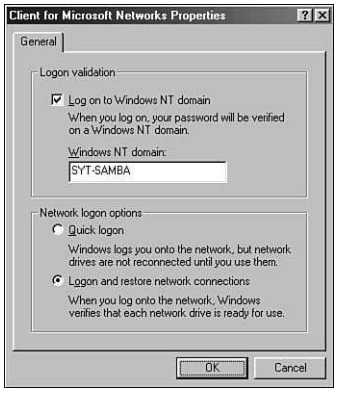
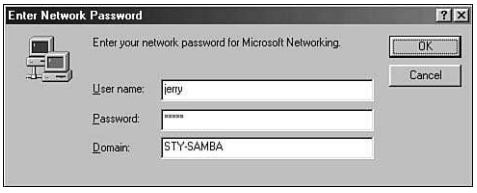


FIGURE 21.1
Enabling a Windows 98 client to logon to the STY-SAMBA domain.

Figure 21.1 displays the Client for the Microsoft Networks Properties window. To access this, open the Network control panel, highlight the Client for Microsoft Networks entry, and click the Properties button. Select the 'Log on to Windows NT Domain' checkbox in the Logon Validation section and enter the name of the domain you want to use. After you have finished, you can click OK buttons until you have backed out of the Network control panel. At this point, Windows might want to copy some files from the Windows 98 install

CD. After the gratuitous system reboot, you will see the logon dialog box shown in Figure 21.2.

FIGURE 21.2
Windows 98 domain logon screen.



Successfully Logging In to the Domain

If you can successfully log in to the domain, the login script will execute in a DOS window, as shown in Figure 21.3. The beauty of this is that, with a few small exceptions, the Windows 9x/ME client cannot tell the difference between a Samba-controlled domain and a Windows NT–controlled domain. Therefore, it is most likely that your users will not be able to tell the difference either.

FIGURE 21.3
Logon script execution window after successfully logging into a Samba (or Windows) domain.



One advantage of using a login script over using the Windows persistent connection setting to initiate drive connections is that you can make changes to the login script without sitting at the client machine. You can reassign network connections and even run OS and application patches in the login script. Be creative, but always test your changes first.

Troubleshooting Windows 9x/ME Domain Logons

There are three common errors that a Windows client may encounter when logging in to a domain (Samba or Windows NT/2000). The three error messages are

- No domain server was available to validate your password.

- The domain password you supplied was incorrect or access to your server has been denied.
- Incorrect parameter.

The first error, 'No domain server was available to validate your password,' is Windows's way of informing you that it '. . . asked around to see whether anyone would attempt to validate this username and password but nobody responded one way or the other.'

The reason for this is that the client was unable to locate a logon server by

- Using a broadcast mailslot query on the local network for the `WORKGROUP<1c>` name (that is, `STY-SAMBA<1c>` in our case).
- Sending a direct query to the WINS server (if appropriate) for the Domain Master Browser that holds the `WORKGROUP<1b>` name (that is, `STY-SAMBA<1b>` in our example).

If both of these methods fail, the client will give up and display this error message. The methods of resolving this are the same as those used for tracking down browsing problems we discussed in Hours 19 and 20. Namely, use the `nmblookup` and `smbclient` tools to view the browse lists of workgroups and determine why the server is not located by the two lookup methods given above.

The second error message, "The domain password you supplied was incorrect. . . " implies that the logon server was located by rejecting the username/password pair. Other than simply mistyping the information, there are two common sources of this problem.

The first is that the server refused the session setup. A few causes for this were previously discussed in Hour 4, "Starting Your Feet to Dance." To refresh your memory, this can occur if the client's IP address was not included in the `hosts allow` value in `smb.conf` or was listed in the `hosts deny` line. More reasons were outlined in Hour 13, "Troubleshooting Techniques."

Another possibility is that the server is set up to use plain-text passwords, but the client will not downgrade to clear text. You should follow the two solutions listed in previous hours:

- Set up the Samba server to accept encrypted passwords using the steps outlined in Hour 7, "Security Levels and Passwords."
- Enable the Windows 9x client to use plain-text passwords as described in Hour 10, "Microsoft's DOS Network Client and Windows 95/98/ME."

The final error message ('Incorrect parameter') is the most cryptic of the three. Although I cannot determine exactly what the client is trying to say, this is generally a result of using the same name for the `netbios name` and `workgroup` parameters in `smb.conf` (or perhaps on the Windows client) and seems to be tied to NetBIOS name registration issues.

User Profiles

What we have seen so far is fairly nice. We can ensure that users enter the correct information when they log in by validating the username/password immediately, we can be assured that each user has at least some required network drive connections, and we can execute other things on log in. What else is possible?

I have said repeatedly that the beauty of this feature of Samba is that the Windows client, for the most part, does not know that it is not a Windows NT Server acting as the domain controller. Therefore, the majority of the administration tools and techniques used for managing a Windows NT domain can also be applied to our Samba domain. The subjects we will explore next are not specific to using Samba as a domain controller. Rather, they are part of the Windows 9x network model. For more information on this model, refer to the Windows 95/98/ME Resource Kit help files on the Windows installation CD.

A user's *profile* is similar to the collection of dot (`.`) files that Unix uses to control login, logout, and application behavior. If your background is rooted more in Windows terminology, you can think of it as a collection of user-specific `*.ini` files and program groups. The bottom line is that profiles allow a user to customize his or her environment without permanently attaching it to an individual machine.

The Windows Registry 101

Consider this section as enough background to put us on the same page. If you are already familiar with the system Registry, bear with me for a few moments.

The Windows 9x/ME System Registry is a database composed of two binary files, `system.dat` and `user.dat`. The `system.dat` file is always located in the `\windows` directory (or whatever directory you used to install Windows). Normally, the `user.dat` file is located there as well. However, in the case of roaming profiles, it will be downloaded from a user-specific location.

The two files combine for a treelike structure with six main roots. Figure 21.4 shows the system Registry as displayed by the Windows 98 Registry Editor. I will concern myself with only two of them: `HKEY_LOCAL_MACHINE` (`HKLM`) and `HKEY_CURRENT_USER` (`HKCU`).

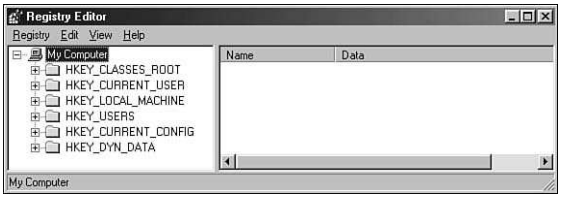


FIGURE 21.4
Windows 98 Registry Editor.

The HKLM hive (each of the six roots is called a hive) is supposed to contain information that is local to the machine only. This is not always the case because applications do not always use the Registry as they should.

The HKCU hive contains information related to the currently logged on user. Some examples of settings that are stored in the HKCU hive include wallpaper and screensaver settings, recently used file lists for applications, and other user-specific file locations. The local machine can be set to use the same user information for anyone who logs in, or it can keep track of information for each user. The former circumstance is the default behavior for Windows. In this case, the HKCU hive is the same as the HKEY_USERS (HKU) hive. You can enable individual profiles with the User Profiles tab of the Passwords Properties control panel (see Figure 21.5).

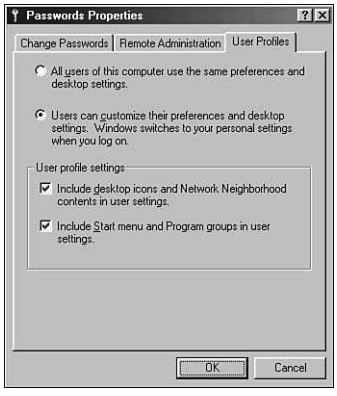


FIGURE 21.5
Enabling User Profiles in the Windows 98 Password applet.

What Else Exists in a User Profile?

Being able to maintain individual settings in the Registry alone is very useful. Windows will also allow you to associate its Start Menu and Desktop icons with a user profile.

You can configure individual user profiles in two ways. The first method allows only a single machine to use the user profile that is stored locally. The second method allows the profile to follow the user on the network so that any machine that the user logs in to has access to the profile. The latter is referred to as a roving (or roaming) user profile.

If the Windows client with user profiles enabled is configured to log in to a domain, the local system automatically attempts to store the profile on the user's home directory. The profile is then cached from the network to the local disk at login time and then back to the user's home directory upon logout.

When using Samba as the domain controller, the network location used by Windows 95/98/ME clients to store the roaming profile is determined by the `logon home` global parameter in `smb.conf`. The default value for this is, of course, the user's home directory. Other than hacking values of various registry keys, there is no way to get a Windows 9x/ME client to place the user's profile outside of the UNC path specified by this parameter. However, it is possible to inform the client to place the profile in a subdirectory below the home directory by defining a value such as

```
logon home = \\pogo\%u\profile
```

The Windows client uses the file UNC path for locating and storing the user's profile. However, the server and sharename are used to define the user's home directory. In this case, we get the best of both worlds.

Another feature of the `logon home` parameter is to define which share gets mapped when a Windows client executes

```
C:\WINDOWS\> net use * /HOME
```

The result is the UNC path of the value of the `logon home` setting (that is, `\\pogo\%u`).

Making User Profiles Work for You

Perhaps some justification is needed for the value of user profiles. How can they really make our jobs easier? Here is one example that I used quite often when I managed Windows 98 student labs at a university.

Suppose that we have created a Windows shortcut file that should be placed on everyone's desktop (or on the Start menu, for that matter). What are our possible options?

- You can send out instructions for creating the shortcut. (This is bad.)
- You can go to every machine and add the link manually. (Also bad.)
- You can use the login script to copy the shortcut. (This is okay.)
- You can set a `preexec` script on the user's home directory that copies the necessary files to the user's roaming profile. (This is even better.)

Although the third option would work, I favor the last one because I prefer to script in Perl or a shell language as opposed to using batch files. In addition, you can pass `smb.conf` variables to `preexec` or `postexec` scripts and have the variables interpreted correctly.

Let me set the stage. We have defined the `logon home` option as

```
logon home = \\pogo\%u\profile
```

We have configured the follow settings for the `[homes]` service:

```
[homes]
     comment = Windows user profiles
     preexec = /usr/local/bin/buildprofile %H
     path = %H
     create mode = 0600
     directory mode = 0700
```

Now here is the `buildprofile` script:

```
#!/bin/sh

home=$1
umask 077
if [ !-f $home/profile/desktop/somelink.lnk ]; then
    cp -p /usr/local/samba/lib/somelink.lnk $home/profile/desktop/
fi
```

In this script, `somelink.lnk` is the name of the shortcut file to distribute to each user's desktop. When Windows connects to the `[homes]` service to access the user's profile, `smbd` executes the `preexec` script that copies the necessary shortcut. It is possible to perform many kinds of elaborate tricks with this type of configuration. Be creative!

Windows System Policies

System policies are closely linked to the Windows Registry. The relationship can be explained like this: Policies define what should be allowed and what should be restricted. The Windows Registry contains the current policy settings in the form of Registry keys.

An example of a policy setting that we have already seen is the registry key for disabling password caching on Windows 9x clients (refer to Hour 17, "Security Tips," for the details). It is the operating system's responsibility to enforce the policy settings recorded in the Registry.

Just as Windows provides a Registry editor, `regedit.exe`, it also provides a policy editor named `poledit.exe`, which is displayed in Figure 21.6. Discussing every detail of policy files and templates is beyond the scope of this book. Rather, I will focus on how to configure Windows clients to download the policy files from a server as well as the potential of system policies.

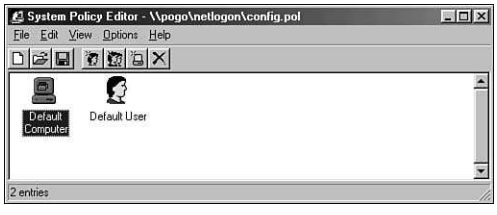


FIGURE 21.6
The Windows 98 Policy Editor.

The System Policy Editor can be loaded from the Windows installation CD. Instructions for installing the tool are located in `\admin\apptools\poledit` in a file named `poledit.txt`.



The location of the policy editor on the Windows 9x/ME install CD tends to change with each release. It if often easier to search the CD for all files that match `poledit*.*`.

The Policy Editor allows you to create a file that can be merged with the local system Registry during login. Windows 9x/ME clients will automatically download this file using the default UNC path of

`\\server\netlogon\config.pol`

where `SERVER` is the domain controller used to validate the user's logon credentials.

Now that you have been exposed to the Policy Editor, Figure 21.7 illustrates the process of logging in to the network when the policy file is downloaded.

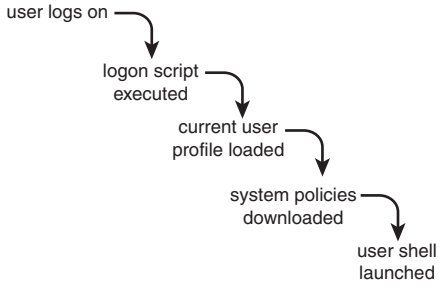


FIGURE 21.7
Chronological look at logging in from a Windows 9x/ME client.

To create a policy file (for example, `config.pol`), start the Policy Editor and choose New File from the File menu. You can define settings such as

- Restricting access to portions of the control panel
- Removing items from the Start menu and the Desktop, such as the Network Neighborhood
- Requiring the client to validate the user login before granting access to the Desktop
- Disabling file and printer sharing

Experiment with the settings available. You can find out more information about system policies in the Windows 95/98/ME Resource Kits. See the Windows installation CD for

the available documentation. When you have created your policy file, save the settings in a file named `config.pol` in the `[netlogon]` share on the DC.

I will offer one last word about policies. Although they do provide a great deal of control, there is no mechanism native to Windows 9x that prevents a local user from changing policy settings in the Registry. You have been warned.

Summary

Samba's domain control capabilities for Windows 95/98/ME clients provide a means to validate network logins and not just resource connections. The three requirements for a Samba DC are

- The server must be in user level security.
- Domain logins must be enabled (that is, `domain logons = yes`) in the `smb.conf` file and Samba is configured as the DMB (i.e., `domain master = yes`).
- A share named `[netlogon]` has been correctly configured in `smb.conf` and is accessible to users.

Beyond standard domain logins and login scripts, Samba also supports Windows 9x capabilities to download system policies and store user profiles on network drives.

Q&A

Q I have successfully set up Samba as a domain controller and use login scripts. However, when a user logs in, the batch file runs correctly, but the Windows NT Logon Script window hangs there and eventually the user clicks the Cancel button. What am I doing wrong?

A You are doing nothing wrong. This little annoyance is caused by the batch file signaling its end of execution by creating a file named `LMSCRIPT.$$$` in the current working directory. The Windows NT logon script window should then clean things up. The problem occurs when the batch file cannot create the `LMSCRIPT.$$$` file. Therefore the NT logon script window never knows that the batch file has completed. You may have changed to a drive and directory that the user does not have write permission in. To remedy this, add a line to the end of the script that changes back to the user's local hard disk (for example, c:).



HOUR 22

Domain Control for Windows NT 4.0/2000

Samba 2.2 is the first stable Samba release to officially support domain logons from Windows NT/2000 clients. The prior 2.0 release included some Primary Domain Controller (PDC) functionality, but this was an experimental option for that release and included only minimal PDC capabilities. With the 2.2 release, Samba is able to act as a Windows NT 4.0–style PDC for Windows NT 4.0 and Windows 2000 clients.

The capability to run a PDC on a Unix server opens up tremendous possibilities that would not exist otherwise. Managing user accounts is easier when the information (that is, usernames and passwords) is readily accessible on one machine (or at least the means to do so are known). Another advantage is the ability to leverage the same hardware for other Unix services in addition to Samba. Given a moderately sized server, there is no reason why a Samba PDC could not also function as the NIS master or DNS server for the network. Without this, it would be necessary to purchase, install, and maintain a completely different machine just to act as the PDC. Who really wants another server OS to support?

For the remainder of this hour, we will examine what PDC features have been implemented in Samba and how to configure them. As in Hour 21, "Domain Control for Windows 95/98/ME," we will also look briefly at system policies and user profiles. This hour does go into some complicated topics so don't be surprised if you find yourself thumbing through previous hours to remind yourself of how things like WINS and network browsing function.

Current Features

As of version 2.2.2, Samba is able to act as a Windows NT 4.0 PDC for Windows NT 4.0 (up to SP6) and Windows 2000 (up to SP2) clients by using NetBIOS and NTLMv1 (although NTLMv2 and Kerberos support is under development). At this point, it is far easier to highlight the major PDC unimplemented features, which are

- Samba cannot initiate a Trust Relationship with another Windows NT 4.0 style Primary Domain Controller.
- Samba cannot synchronize the Windows NT 4.0 System Account Manager (SAM) database of user and group accounts with either a Windows NT 4.0 PDC or a Backup Domain Controller (BDC). However, it is currently possible to configure a Samba PDC with multiple Samba BDCs. More of this last point will be presented later during this Hour.
- Samba does not provide any mechanisms for mapping Unix groups to Windows NT global or local groups.
- Samba allows only a few user account attributes such as passwords and the "Account Disabled" flag to be modified via the Windows NT User Manager for Domains.



It is always a good idea to test any essential applications with a Samba PDC before deploying them. Server applications, such as Exchange and MS's SQL server, may not work in all environments. The reason is that these applications can use new MS-RPC calls that Samba developers have not encountered during the normal PDC development. If you encounter a server application that will not function due to missing MS-RPCs, the best thing to do is to report the error to the samba-technical mailing list. Be ready to provide extensive log files to help developers gain the information needed to implement the missing functions.



Samba 2.2 can function as a Windows NT PDC only, not as a Windows 2000 Domain Controller using Kerberos 5 and Active Directory.

How to Configure a Samba PDC

Configuring Samba to act as a Domain Controller for Windows NT/2000 clients is very similar to the process of configuring it as a Logon Server for Windows 9x/ME clients. Therefore, we will use the `smb.conf` developed last hour as our starting point. The following `smb.conf` is reprinted from Listing 21.1.

LISTING 22.1 `smb.conf` File from Hour 21

```
##
##   Mon Aug 27 01:09:17 CDT 2001
##   jerry carter <jerry@samba.org>
##
##   Sams Teach Yourself Samba in 24 Hour
##   smb.conf for Windows 9x logon server

[global]
    netbios name = POGO
    workgroup = STY-SAMBA
    security = user
    encrypt passwords = yes

    domain logons = yes
    logon script = logon.bat

    os level = 64
    domain master = yes
    local master = yes
    preferred master = yes

[netlogon]
   path = /export/smb/netlogon
   read only = yes
   public = no

[homes]
    comment = Home directories for STY-SAMBA domain users
    create mask = 0600
    directory mask = 0700
    browseable = no
    valid users = %S
```

In addition to the required settings for acting as a Windows 9x/ME Logon Server (`security = user`, `domain logons = yes`, possessing a `[netlogon]` file share, and so on), there are three key points to realize when comparing a Logon Server to a Windows NT Primary Domain Controller.

- Windows NT/2000 clients locate the PDC by resolving the NetBIOS name of the Domain Master Browser for their Domain. (Refer to Hour 20, "Cross Subnet Browsing.")
- A Samba PDC must have encrypted passwords enabled in the `smb.conf` (`encrypt passwords = yes`) as described in Hour 7, "Security Levels and Passwords."
- All Windows NT/2000 (or Samba) hosts that are members of the Samba-controlled domain must possess a machine trust account on the PDC.

Our existing `smb.conf` from Hour 21 already meets the first two requirements. The third item, machine trust accounts, is a large enough topic to necessitate a section of its own.

Machine Trust Accounts

An easy means of describing a machine trust account is to call it a "user account where the user is actually a workstation or server that is a member of the domain." The purposes of having a user account for machines are to allow a domain controller to identify the domain member and to maintain a secret key (the account's password) that can be used to encrypt certain conversations between the DC and member. This encryption prevents other hosts on the network from being able to listen in on conversations and learn privileged information, such as usernames and passwords.



Windows 9x clients, though they might appear to log in to a domain, are not considered to be true members of the domain. This is because a Windows 9x machine has no trust account. Therefore, the DC cannot be for certain that the Windows 9x/ME client machine is who it says it is. If the client machine cannot be authenticated, the PDC has no means of determining whether this is really the machine named `BRUTUS` or if it is another host that has registered the NetBIOS name in its place.

Each domain member has its own machine trust account and therefore its own password. The username for a machine trust account is the NetBIOS name of the client machine with a '`$`' character appended.

Adding any type of client to a Windows NT domain follows the same steps

1. Creating a machine trust account for the client on the domain controller.
2. Joining the client to the domain.
3. Performing the obligatory reboot of the Windows client.

We will spend some time in the next few sections explaining each one of these steps in detail.

Creating a Machine Trust Account on a Samba PDC

There are two means of creating machine trust accounts in a Windows NT domain.

- Create the machine account manually before attempting to join the client to the domain.
- Create the machine accounts from the client at the time of joining the domain.

First, we will consider a prerequisite for machine accounts in general.

Each machine trust account is simply a user account that is used by a machine, and each one must also have an associated Relative ID (RID) (see Hour 14 if you need a refresher on RIDs and SIDs). In a Windows NT domain, all user accounts, trust accounts, and groups exist in the same number space ($1000 – (2^{32} – 1)$). This means that it is not possible to have a group with a RID of 1000 and also a user account with a RID of 1000. Many modern Unix systems have a common practice (and in fact standard practice on some Linux distributions) of creating a user and group use the same name and equivalent `uid` and `gid`. The reason this is possible is that the `uid` is allocated from on pool of integers and the `gid` is allocated from another pool, both distinct and unrelated from the other.

A Samba PDC must possess some mechanism to ensure that each machine trust account has a unique RID. The most obvious solution is to use the same method for machine accounts as for user accounts and generate the RID from the respective `uid` using a mathematical function.

The downside of this solution is that Unix does not support the concept of a login account for machines except in the case of a directory service such as NIS+. The most portable solution is to require that each machine trust account in a Samba domain have an entry in the local system password file (that is, `/etc/passwd`) and therefore a unique `uid`. Although not the most elegant solution, and one the Samba developers hope to

correct soon, this requirement has held over from the beginning Domain Control implementations from as early as the alpha releases of version 1.9.18.

Because many network administrators implement customized systems for creating user accounts, it is very difficult to describe a general method for adding an account to `/etc/passwd`. However, the following command is usually sufficient for modern Linux distributions. The home directory (`-d`) and shell (`-s`) are disabled and the primary group is set to a designated Unix group for machine trust accounts. The value of the "machine account$" is the NetBIOS name of the new domain member appended with a "$".

```
root# useradd -d /dev/null -g ntmach -s /bin/false "machine name$"
```

The next command prevents anyone from successfully validating against the `/etc/passwd` entry. Remember that `smbd` needs only the entry to obtain a `uid` for the machine account.

```
root# passwd -l "machine name$"
```

The result is an entry that appears similar to

```
nt-client$:x:790:200::/dev/null:/bin/false
```

This account need never be used by any Unix service except Samba. By disabling the shell, home directory, and password, we are securing the account from unauthorized use.



For more information on the `/etc/passwd` file format, see the `passwd(5)` man page. It may also be helpful to refer one of the following books: *Unix Unleashed* by Robin Burk, *Linux Unleashed* by Bill Ball, *Essential System Administration* by Aeleen Frisch, or *Unix System Administration Handbook* by Evi Nemeth.

Now we can return to our original discussion of how to create a machine account that can be used when a host attempts to join the domain. The first option was to manually create the account before joining the client to the domain. In addition to normal user accounts, the `smbpasswd` tool is also able to create machine accounts.

The following command will create a machine trust account entry in the `smbpasswd` file, which can be used by a host when joining the domain.

```
root# smbpasswd -a -m "machine name$"
```

This initial entry will have a well-known password, which is the NT hash of the machine's NetBIOS name in lower case letters. For example, if the machine account username is `nt-client$`, the initial password would be set to the NT hash of the string

`nt-client`. It is best to join the client host to the domain immediately after creating the account because, until the password is changed, any client claiming to be the new host could join the domain by successfully guessing the password.

The more secure method of joining a Windows (or Samba) domain is to use an administrative account and create the machine account at the time it is needed. In fact, Windows 2000 requires that this method be used. Figure 22.1 shows the dialog used to join a Windows NT client to a domain when specifying an administrative account for securing the conversation with the PDC (Start -> Settings -> Control Panel -> Network).

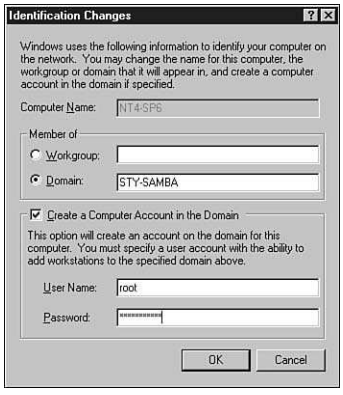


FIGURE 22.1
Joining a domain by specifying an administrative account to secure the client conversation with the PDC.

As of Samba 2.2.2, this administrative account must be an account that maps to the root user on the Samba server. This next command will add a user named `root` (which implicitly maps to the superuser on the Unix host) for Samba to use.

```
root# smbpasswd -a root -s secret
```

This requirement has been known to make some system administrators nervous. However, by defining a different password in `root`'s `smbpasswd` entry from the one contained in `/etc/passwd` (or `/etc/shadow`), the security risk is minimized. Another means used by some administrators is to create a `username map` entry that maps the `Administrator` login name to `root`. This method does not offer any more security but it does visually distinguish between the purposes of the `smbpasswd` entry.

It may also be prudent to specify the root account as an `invalid user` for all shares to further reduce the amount of damage that could be done if the account was compromised. However, you cannot place the root account in the default list of `invalid users` by specifying it in the `[global]` section of `smb.conf`. This will cause the client joining the domain to fail. In this case, the `copy` parameter can be used to specify a template

share that possesses default settings without requiring these parameters to be defined in the [global] section. An example of how this could work is shown here.

```
## Template share containing common settings
[template]
     available = no
     invalid users = root

## new share based on [template]
[new-share]
     copy = template
     available = yes
     read only = no
```

The [new-share] service is based on the [template] share definition. The available = no setting in [template] is used to prevent users from connecting directly to this share.

Once we begin creating accounts in the smbpasswd file on the fly like this, it opens the possibility of attempting to join a host to the domain that does not have an entry in /etc/passwd. The remedy for this is to define a value for the add user script parameter that can create these entries as needed. The following setting uses the same useradd command that was previously discussed.

```
add user script = /usr/sbin/useradd -d /dev/null -g ntmach \
  -s /bin/false %u && /bin/passwd -l %u
```

The flow of events for creating a machine account from an NT client joining the domain is

1. The smbd daemon attempts to locate an existing smbpasswd entry for which it can change the password.
2. If an existing entry is not found, smbd will execute the add user script (if it is defined).
3. The smbd daemon will then attempt to create a new smbpasswd entry. This step requires that the add user script executed in Step #2 succeeded.
4. The smbd daemon will then attempt to look up the entry in smbpasswd again. If it is found this time, it returns success to the client. If the entry cannot be located for the second time, smbd returns an error to the client.

Joining the Domain

Joining a client to a Samba-controlled domain is identical to joining a Windows NT 4.0–controlled domain. The example in this section uses a Windows 2000 client, but the process is very similar for Windows NT 4.0.

The first step is to open the Network Identification tab of the System Properties dialog. This window can be opened by right-clicking on the "My Computer" desktop icon and selecting the "Properties" option from the menu (see Figure 22.2).

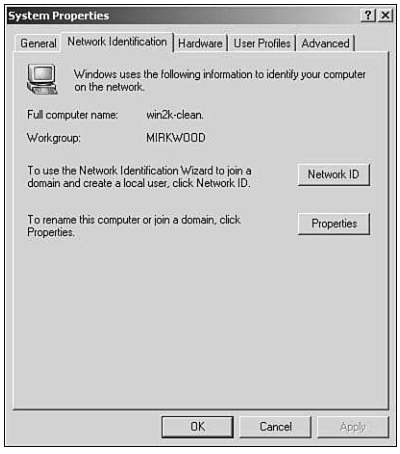


FIGURE 22.2
Windows 2000 Network Identification window.

After clicking on the Properties button, entering the desired domain name, and selecting the OK button, we are prompted to enter the Administrative username and password for joining the domain. After entering the root username and password defined in the `smbpasswd` file, we are finally greeted by the welcome message shown in Figure 22.3.



FIGURE 22.3
Welcome to the STY-SAMBA domain!

If you receive an error message indicating that the domain controller could not be found, make sure that Samba is running, that the client machine can resolve the name `STY-SAMBA<1b>`, and that the client can actually connect to the server. For example, verify that the machine is not listed in a `hosts deny` setting on the server.



It is possible to join a Windows XP client to a Samba–controlled domain. However, the following registry setting must be applied to the XP client before joining the domain.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\netlogon\parameter
"RequireSignOrSeal"=dword:00000000
```

Merge the registry file, distributed with Samba under the name `docs/Registry/WinXP_SignOrSeal.reg`, by double clicking on the file icon and reboot the XP client. Then follow the same steps for joining the domain as you would for a Windows 2000 client.

Time to Reboot the Client

After rebooting the NT or 2000 client, you should be able to log in using a valid domain account. If necessary, create a user account in `smbpasswd` to be used for testing purposes. Assuming that a user account named `testuser` already exists in `/etc/passwd`, simply add this user to Samba's `smbpasswd` file by running

```
root# smbpasswd -a testuser -s testpass
```

When logging into the console of the Windows NT/2000 client, ensure that the `STY-SAMBA` domain is selected, as shown in Figure 22.4, rather than the local machine name in the Domain drop down menu. After entering the `testuser` login name and the correct password, you are logged in as a domain user and are greeted with a normal Windows desktop.



FIGURE 22.4
Selecting the STY-SAMBA domain when logging into the console of a Windows 2000 client.

If you receive an error message that says "The `STY-SAMBA` domain is not available" when attempting to log in, again make sure that Samba is running and that the client machine can resolve the `STY-SAMBA<1b>` NetBIOS name using techniques presented in Hours 19 and 20 when we discussed network browsing. It might also be a good idea to refer to Hour 18 if you are in a multi-network environment and using a WINS server.

Of course there is more to a Windows NT domain than simply logging in. Many security mechanisms are built in to Windows NT/2000. For example, an NTFS file system supports access control lists for setting permissions on files, directories, and printers. A Samba PDC also supports the capability of defining a location for storing roaming user profiles and for distributing system policies as you can with Windows 9x.

Domain Admins and Domain Guests

The `Domain Admins` and `Domain Guests` groups are predefined accounts on Windows domain controllers, just as the `Domain Users` group is. By default, a Samba PDC defines all user accounts to be members of the `Domain Users` group. To add a user to either of these other two groups, Samba provides two `smb.conf` parameters that accept a list of users and/or Unix groups in the tradition of the `valid users` parameter described in Hour 8, "Samba—The File Server."

The `domain admin group` option allows us to specify a list of accounts that should be added to the `Domain Admins` group (in addition to any other groups of which the user is a member). All Windows clients automatically add this group to the local `Administrators` group when joining a domain. So though the `domain admins group` does not give a user special privileges on the Samba DC, it does grant administrative rights on domain members. The following setting makes all members of the `wheel` Unix group a Domain Administrator.

```
[global]
     <...>
     domain admin group = +wheel
```

The `domain guest group` performs a similar function by adding the `Domain Guest` group to the various group memberships for a user. In practice, this parameter is used very little in comparison to the `domain admin group` option. Neither of these parameters has a default value.

logon home, logon drive, and logon path

Part of a user's account information in a Windows NT domain is the location of and mount point of the user's home directory as well as a location for the roaming user profile (see Figure 22.5). The user's login script is also included with this information. We can associate some of the fields shown in the window with `smb.conf` parameters.

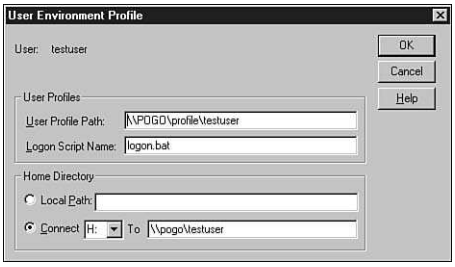


FIGURE 22.5
Using the Windows 4.0 User Manager for Domains for viewing Samba user account details.

First, examine the settings relating to the user's home directory. The Home Directory section of the window specifies a drive letter to which the network home path should be connected. These are the `logon drive` and `logon home` parameters, respectively

```
logon drive = Z:
logon home = \\%L\%u
```

These settings, along with those displayed in Figure 22.5, are the default settings for these parameters.

The `logon home` parameter accepts a path that is mounted as the user's home directory. With Samba, this is easy to do using the `[homes]` service. If you want, you can also specify things such as

```
logon home = \\server\users\%u
```

This assumes that you have defined a share named `[users]` and that each user has a directory that matches his username. `server` is the name of the (possibly remote) server that provides the `[users]` share.

Finally, the `logon path` parameter defines the location on the network that contains the user's roaming profile. The default is to store the profile information in the user's home directory. Our server will follow the convention of storing user profiles in a share separate from the `[homes]` service, as shown here:

```
logon path = \\server\profiles\%u
```



Storing a user's profile in her home directory can be very problematic for NT machines. The reason is that these clients have a tendency not to disconnect from the user's home directory upon logout. When the next user logs in, the profile is unavailable because the NT/2000 client attempts to reuse the existing connection to the home directory. In most circumstances, the new user will have no access to the profile of the old user and Windows will complain loudly (but not as loudly as the user). Always store NT user profiles in a share outside of the `[homes]` service.

If the `logon path` is defined local to our Samba PDC, the `[profiles]` file service could be configured as

```
[profiles]
   comment = Roaming user profiles
   read only = no
   path = /export/smb/ntprofiles
   create mask = 0600
   directory mask = 0700
```

The permissions on the directory `/export/smb/ntprofiles/` should be set to be world writable, similar to the `/tmp/` directory. This can easily be set by executing:

```
root# mkdir -p /export/smb/ntprofiles
root# chmod 1777 /export/smb/ntprofiles
```



Roaming user profiles can be disabled by defining the `logon path` to be an empty string (for example `logon path = ""`).

User Profiles and System Policies

User profiles and system policies are conceptually the same as they are in Windows 9x, but the importance of the two greatly increases because of the security mechanisms available in Windows NT.

Consider this example: Suppose that you use system policies to remove the Shut Down option from the Start menu. Under Windows 9x, the user can simply change the setting back to include the Shut Down option, kill `explorer.exe`, restart the shell, and voilà! The Shut Down menu returns.

However, if you set this same policy under Windows NT, you can restrict the user from changing it by using ACL on various Registry keys. Windows NT not only provides the policy options, but also the means to enforce the policy settings, something that is lacking in Windows 9x.



User profiles under Windows NT become more of a necessity than the luxury that they are under Windows 9x. I treat Windows NT clients more like Unix workstations than PCs, in the sense that the machine is generally locked down to prevent changes and the user has a specified location (usually the home directory) in which to save files. This is especially true in the case of public lab environments.

Therefore, it becomes necessary to provide users with an environment that can be set and that follows them from machine to machine. User profiles in a Samba controller domain are, by default, roaming profiles. This should be obvious from the default settings of the `logon path` parameter.

As a general rule, the management of user profiles in a Samba-controlled domain is the same as it is for a Windows NT–controlled domain. For example, one possibility is to create a `Default User` profile on the `[netlogon]` share of the PDC that will be used for all new users. This method works the same whether your PDC is a Samba server or a Windows NT box.

One issue does arise when using a Samba server for a PDC. If you do not have a copy of the Windows NT 4.0 Server CD-ROM, where can you get the server tools, such as the Server Manager, the User Manager for Domains, and the System Policy Editor?

Luckily, you can download these from the Microsoft Web site or its FTP site. The Policy Editor is distributed with the various Service Packs for Windows NT 4.0, although it is not installed on workstations by default. The Windows NT 4.0 versions of the Server Manager and the User Manager for Domains are available for download from `ftp://ftp.microsoft.com/Softlib/MSLFILES/`. The filename is `SRVTOOL.EXE`. A version of these tools that can be installed on Windows 9x is available from the same location and is named `NEXUS.EXE`.

Implementing Backup Domain Controller (BDC) for a Samba-Controlled Domain

Although Samba 2.2 does not support the SAM replication protocol used by Windows NT 4.0 domain controllers, it is possible to configure Samba-to-Samba replication and therefore a BDC. In order to understand why this is possible, however, we will need to have a firm grasp of how network browsing works. If you need a refresher on this, go back and reread Hours 19 and 20.



A Backup Domain Controller (BDC) is a server that is able to process logon requests by maintaining a read-only copy of the user and group accounts for a domain. This read-only SAM is periodically updated from the domain's PDC.

The key point to understanding how to configure a Samba BDC in a Samba domain is to realize that the list of logon servers is registered under the DOMAIN<1c> group name (for example, STY-SAMBA<1c>). Multiple Samba machines can register this name simply by enabling the domain logons parameter. However, only one machine can be allowed to register the domain master browser name (for example, STY-SAMBA<1b>). In Samba-speak, this means that only one machine should have domain master = yes, but many machines can have domain logons = yes. In fact, the smb.conf used by a Samba BDC should be identical to the one used by the Samba PDC with the following exception:

```
### smb.conf for Samba to Samba BDC
[global]
     <...>
     domain master = no
```

If multiple Samba domain controllers are installed on the same network (that is, the same network address with the same broadcast address), the rules of electing a local master presented in Hour 19 also apply. However, remember that, with the exception of the domain master browser NetBIOS name, network browsing is completely separate from domain control.

Using nmblookup, it is possible to determine the PDC and BDCs for our network. As we have already learned, the PDC for a domain is assumed to be the DMB for that workgroup. The following command queries the WINS server at IP address 192.168.24.3 for the STY-SAMBA<1b> name.

```
$ nmblookup -U 192.168.24.3 -R 'STY-SAMBA#1b'
querying STY-SAMBA on 192.168.24.3
192.168.1.75 STY-SAMBA<1b>
```

We can also determine the list of domain controllers for the STY-SAMBA domain by resolving the STY-SAMBA<1c> name.

```
$ nmblookup -U 192.168.24.3 -R 'STY-SAMBA#1c'
querying STY-SAMBA on 192.168.24.3
192.168.215.2 STY-SAMBA<1c>
192.168.1.75 STY-SAMBA<1c>
```

This shows that there are two domain controllers for our domain, one at 192.168.215.2 and another at 192.168.1.75. From our previous query, we know that the latter machine is also the PDC.

If we need to locate the local master browser (LMB) for a particular network segment, the WINS server will not be able to help us. Because the LMB is elected from all hosts

on a single subnet, we must query the broadcast address in order to resolve the STY-SAMBA<1d> name.

```
$ nmblookup -B 192.168.215.255 'STY-SAMBA#1d'
querying STY-SAMBA on 192.168.215.255
192.168.215.2 STY-SAMBA<1d>
```

If multiple Samba hosts in the Samba domain (that is, the same value for the `workgroup` parameter) are installed with domain logons enabled, it makes sense that these hosts should also have certain information synchronized such as:

- The `smbpasswd` file. All users including machine accounts in this file must also have a `uid` in `/etc/passwd`, so it may be necessary to synchronize that file among domain controllers as well.
- The domain SID stored in the `private/MACHINE.SID` file (or in `private/secrets.tdb` in more recent Samba releases). The `MACHINE.SID` (or `secrets.tdb`) file must be synchronized among Samba DCs, as this represents the Security Identifier for the domain. If this changes, domain members will not be able to logon to the domain.
- The contents of the `[netlogon]` file share (for example, system policies, logon scripts, and so on).



The standard mechanisms for registering and resolving NetBIOS names still apply here, so that a working WINS server is a requirement for configuring multiple Samba domain controllers in a multi-network environment.

Andrew Tridgell has written another wonderful piece of software named `rsync` that provides a very efficient means of synchronizing files or directory trees. More information about `rsync` can be found at `http://rsync.samba.org`.

The following script will ensure that all configuration files (except `smb.conf`) are up to date on all Samba domain controllers.

LISTING 22.2 Synchronization Script for Samba Domain Controllers

```
#!/bin/sh
##
## Simple script to synchronize configuration files
## among Samba DCs.  THIS IS ONLY AN EXAMPLE.
## Customize it to meet your needs.
##
```

continues

LISTING 22.2 Continued

```
## jerry@samba.org
## Sams Teach Yourself Samba in 24 Hours
##

## lists of Samba BDCs
DCLIST="bdc1.plainjoe.org bdc2.plainjoe.org"

## rsync arguements
RSYNCARGS="-tlpWrv --delete"

## make sure that root's identity.pub key has been added to
## ~root/.ssh/authorized_keys file on the destination host
## and that it is not password protected.
RSYNCSSH="-e ssh"

## files to synchronize
SMBPASSWD=/usr/local/samba/private/smbpasswd
SIDFILE=/usr/local/samba/private/MACHINE.SID
NETLOGONDIR=/export/smb/netlogon

for host in $DCLIST; do
  rsync $RSYNCARGS $RSYNCSSH $SMBPASSWD $host:$SMBPASSWD
  rsync $RSYNCARGS $RSYNCSSH $SIDFILE   $host:$SIDFILE
  rsync $RSYNCARGS $RSYNCSSH $NETLOGONDIR $host:$NETLOGONDIR
done

exit 0
```

Assuming that the Samba DCs Unix username information is shared with the BDCs via NIS or another means, this script is all that should be necessary.

Windows 2000 Domains

After all the features covered so far, there is not much to add about Windows 2000 domains at the moment. As of this writing (version 2.2.2), Samba does not support acting as a DC for a true Kerberized, Windows 2000 domain. There are many issues that much be resolved by developers, such as

- Kerberos 5
- LDAP
- Dynamic DNS updates
- and so on

The Samba developers are pursuing means to support the native Windows 2000 authentication protocols, but that will have to wait until the next edition of this book.

Summary

Samba 2.2 finally contains an "officially supported" Windows NT 4.0 Domain Control implementation, even if it is not entirely complete. This is different from Samba's capability to validate Windows 9x/ME domain logins because these clients do not possess a machine trust account. The main requirements of a Samba PDC are

- `security = user`
- `domain logons = yes`
- `encrypt passwords = yes`
- `domain master = yes`
- a `[netlogon]` file service
- storing machine trust accounts for domain members

The main features that are not implemented currently are

- Trust Relationships
- SAM replication with a Windows NT DC

It is possible, however, to install a set of Samba DCs (a PDC and multiple BDCs) to work together by synchronizing certain configuration details and properly enabling network browsing.

Q&A

Q Can Samba 2.0 act as a Windows NT domain controller?

A There is some limited functionality for PDC support implemented in Samba 2.0, but it is broken when compared with the latest Samba 2.2 releases. In particular, version 2.2 or later is required in order to support domain logons from Windows 2000 clients.

Q Can a Samba PDC have a trust relationship with another Windows NT 4.0 domain?

A No. Samba 2.2 does not support initiating trust relationships when configured as a Primary Domain Controller.

New Terms

machine trust account A type of user account used by a domain member to authenticate itself to the domain controller and protect certain information when in transit across the network.



HOUR 23

Capacity Planning and System Tuning

One of the questions that we should always ask when building a new Samba server is, "Will it be able to withstand the stress of our users?" There are no firm rules on how much hardware it will take to support a certain number of clients, but this hour will provide you with several options and guidelines for planning a Samba installation. We will examine things like memory usage and CPU requirements in addition to learning about new `smb.conf` parameters that can help Samba to make better use of available resources. Whether you are running Samba on a Unix desktop workstation for your personal use or planning support for thousands of clients, this hour will help you to understand the issues surrounding Samba's scalability.

How Big is Big Enough?

Assuming that we were building a Samba server to support 10 clients, how much RAM would the server need? What about 100 clients? 1000? How

does this change the server's requirements? These are the types of questions that should be asked up front when designing a new server of any kind, whether it is a file server or a Web server. It is crucial to know what will be expected of your server and in what range of conditions it must operate correctly.

Before trying to estimate how much memory and processor power our server requires, let's answer the following questions:

- How many unique CIFS clients will ever connect to the Samba server? This will gauge the maximum `smbd` process at a single moment.
- What is the average number of clients that will be connected at any one time? This is the daily load of our server and can be harder to estimate. Answering this question requires either a good understanding of the usage patterns of your users or statistics based on past server installations in your environment.
- Will the server be running any software other than the OS and Samba? Smaller networks generally install Samba on a multi-purpose machine that performs other tasks, such as running a mail server or Web server. For larger installations, it is better to dedicate a machine to a single purpose. Remember to take any other running software into account when estimating memory needs.
- Will Samba be a print server, a file server, or a mixture of both? The memory needs of a print server are different for Windows NT/2000/XP clients than they are for Windows 9x/ME clients. This will require more explanation and will be addressed later in this hour.

To illustrate how this estimation would work, consider that we are installing a new server to support a total of 700 clients (a combination of Windows 9x/NT/2000). Research on past servers has shown that we can expect about 500 of the clients to be connected at any one time. The server will be running only Samba and various components required by the OS.

We have estimated that the OS requires approximately 64 MB of RAM, based on the vendor's recommendation. How can we estimate the amount of memory that should be allocated for Samba? By knowing the average number of concurrent connections and the total number of clients, we can apply the following formula.

```
(avg_conn + (total_clients-avg_conn)/2) * smbd_mem_usage = est_req_mem
```

In layman's terms, this means that we calculate the memory needed on a daily basis and throw in enough to cover half of the remaining unconnected clients to handle usage spikes. The headroom factor may be more or less, depending on your environment. For example, at a university you may know that every student lab will be in use from

9:00 a.m. until 11:00 a.m. on Tuesdays. The server must be able to handle connections from every client during this time period. Your memory usage formula would simply be

```
total_clients * smbd_mem_usage = est_req_mem
```

Err on the side of decadence. You can never have too much RAM installed in a server.

The final missing value in our formula is how much RAM to allocate to a single `smbd` process. We will turn to the operating system for this information. Modern operating systems have various means of gathering system information. On Linux, the `/proc` file system is a gold mine for those who know how to dig through it (see the `proc(5)` man page for details). All variants of Unix include some implementation of the `ps` command.

The following statistics were gathered from a SuSE 7.2 GNU/Linux system running a 2.4 kernel. The actual command line options used on your OS will vary. Consult the `ps(1)` man page for the details.

```
$ ps -eo user,pid,ppid,vsz,rss,fname | egrep '(RSS|mbd)'
USER      PID  PPID   VSZ   RSS  COMMAND
root     8199     1  3228  1108  smbd
root     8201     1  2340  1064  nmbd
jerry    8204  8199  3660  1308  smbd
```

The columns of interest are labeled VSZ (virtual memory size) and RSS (resident set size). These tell us the total allocated virtual memory to a process and the portion of that memory which is currently loaded into real memory. This listing is from a lightly loaded server (only a single connection).

If a system does not have enough memory to support the current running process, it will swap a portion of a process out to the virtual memory stored on disk. When a system does not have enough memory to support the working set of a process (the minimal RSS under which the process will continue to operate), it will spend all of its time swapping memory pages in and out of RAM. This is commonly referred to as thrashing and will skew any measurements taken.

The following statistics were taken while the system was thrashing. Notice the RSS of the main `smbd` daemon. Only 152 KB out of an allocated 3 MB of the process are currently loaded into memory! This system is running low on memory. In order for the main `smbd` daemon to service a new client connection, the OS will need to swap more of the process into memory. However, on a system that is low on memory, this means that another process must be swapped out. The process repeats for each process that wants to run but is not currently in memory, and so forth and so on.

```
USER    PID PPID   VSZ  RSS   COMMAND
root   8199    1  3228  152   smbd
root   8201    1  2340  596   nmbd
root   8204 8199  3660 1324   smbd
```

We return to the first memory measurements gathered and estimate 1.5 MB of RAM for the working set of each active `smbd` process. Fill in the variables in our formula as follows:

```
(500 + (700-500)/2) * 1.5 = 900 Mb
```

After adding in the 64MB of RAM needed by the operating system, we round our final number off to 1GB of memory.

The few things that commonly affect `smbd` memory usage are

- Using the `[printers]` service with a large (>200) number of printers and enabling the `load printers` parameter will increase `smbd`'s memory needs slightly. To make all of the printers available, Samba must create a copy of the `[printers]` service for each entry in the `printcap name`. Each service is represented by a data structure in memory and thus the reason for the increased RAM requirements. This is true even if there are no connections to printer shares at all.
- A very active CIFS connection will require slightly more RAM than an idle one.

In this next listing, the process owned by the user `guest1` has been opening a large number of files and scanning his home directory for viruses. Samba is also sharing approximately 170 printers via `/etc/printcap`.

```
USER      PID  PPID    VSZ    RSS   COMMAND
root     7779     1   3456   1348   smbd
root     7781     1   2340   1128   nmbd
guest1   7784  7779   4012   2208   smbd
jerry    7878  7779   3876   1988   smbd
root     7935  7779   3732   1848   smbd
```

In this case, the RSS is 2208 KB. Although an additional 0.7 MB per process may not matter a great deal for a server supporting 5 connections, it increases our memory estimate to 1.42GB ((600*0.7) + original 1 GB)!

What if we know that many users mount their home directory in the morning but rarely access files on it? Can this information help us? Samba provides an option to drop idle connections after so many minutes. The `deadtime` global `smb.conf` parameter is set to 0 by default, which means that `smbd` processes should not exit until the client drops the connection. If we wanted to instruct Samba to drop idle connections after 30 minutes, the following line must be added to the `[global]` section of Samba's configuration file:

```
## drop connections after 30 minutes of no activity
deadtime = 30
```

What happens to the client when the connection to the server goes away? Windows clients will frequently reestablish the connection once it is needed again (for example, a user attempts to open a file on the share). This can be problematic if the server does not support encrypted passwords. Modern Windows releases cache only the hash of the user's password. Remember from Hour 7 that this is a one-way hash. If the client needs the clear text of the password to reestablish the connection, it is out of luck. Hence, another good reason to use `encrypt passwords = yes` in `smb.conf`.

On servers where I have implemented the `deadtime` option, the number of concurrent `smbd` processes has dropped by as much as 40%. Even using a conservative estimate of a 30% decrease, our memory requirements have dropped to

```
(350 + (700-350)/2) * 1.5 = 788 Mb
```

However, memory upgrades still deliver the biggest bang for the buck, so do not skimp on RAM if you think you will need it. These formulas are just guidelines, not laws.

How Much CPU Power Is Enough?

It is more difficult to answer the question of how much processing power is needed than it is to estimate the server's memory requirements. The best guidelines that anyone can provide are:

- Samba is generally more RAM intensive than CPU intensive.
- A multi-CPU system will provide better protection against a single `smbd` process taking more than its fair share of CPU.

There are certain operations that will cause Samba to consume more than an average amount of CPU. A few common examples include

- Scanning files shares for viruses using a PC anti-virus scanner, or file searches that cover a large directory tree.
- Storing Web browser cache files on a network file service. For best performance, these should be cached to the clients' local disks.

The best method of dealing with these issues is to monitor the server's CPU and mark those operations that can be problematic. Then put policies in place that prevent users from accidentally or intentionally performing these actions.

Tuning Your Server

Deciding upon the initial hardware configuration for your Samba server is only the first step. The actual test is in how well the server performs from day to day and in the

reactions from our users. Tuning a server is a gradual process. The number one mistake made by beginning administrators is to change too many settings at once. When performance does improve, it is impossible to determine exactly what modification is responsible.

During this section, we will examine some possible improvements that can be made. It is a good idea to experiment with this one improvement at a time. Some may prove to be fruitful, whereas others may offer no payback for your environment.

Windows NT/2000/XP Clients, MS-RPC, and the `deadtime` Parameter

Understanding how clients will use our Samba server can also help to estimate how many concurrent connections it must be able to support. Windows NT/2000/XP clients use entirely different methods for connecting to printers and domain controllers than Windows 9x/ME. The result is that supporting the former set of operating systems requires more attention to protocol details. In order to execute MS-RPC functions, Windows NT/2000/XP all open a connection to the `IPC$` share on remote servers, as shown in Figure 23.1.

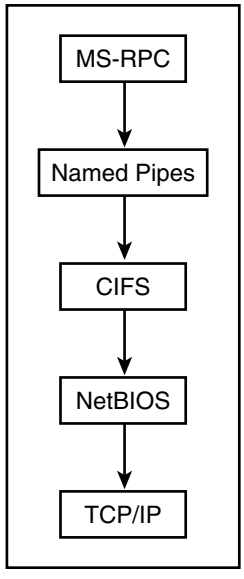


FIGURE 23.1
Implementing MS-RPC over CIFS.

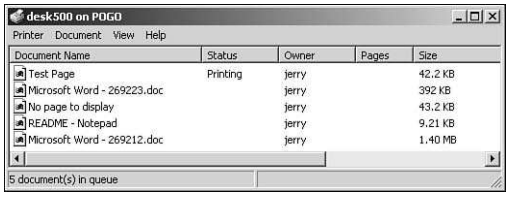
This causes a new `smbd` to be spawned for the connection. Although the RPC operations themselves are fairly short, the CIFS session over which they are executed tends to be long lived. For example, when a Windows 2000 client opens a print queue window on a

network printer, such as the one shown in Figure 23.2, an `smbd` process is continually associated with the printer in order to send periodic refreshes to the queue listing.



If the client has a pre-existing connection to the print server, it will be used. Remember that, when operating in user-mode security, each `smbd` can handle multiple tree connections from a single client.

FIGURE 23.2
Viewing the contents of a Samba Printer Queue from a Windows 2000 client.



Windows 9x/ME clients, on the other hand, use a polling mechanism to ask the server for a list of jobs in the print queue. The client will open a connection to the server, obtain the contents listing of the queue, and then disconnect. This means that Windows 9x/ME clients will use fewer resources on a Samba print server than Windows NT/2000/XP clients. Using a `deadtime = 1` can alleviate the extra stress placed on the print server by Windows NT/2000/XP. The client will reconnect as necessary.



Always perform tests when using a very low `deadtime` value. The one-minute `deadtime` configuration suggested in this section is only for print servers. If Samba provides a mixture of file and printer services, it is best to configure the server as if it were a file server only.

Members of a Windows domain often maintain an open connection to a domain controller in order to authenticate logins and MS-RPC operations. This is a very similar situation to NT printing mechanisms. Using a `deadtime` of five minutes often works well and will cause the domain member to re-establish the connection as necessary, such as when a user attempts to log on to the client's console.

Defining a Ceiling

Even when we have gathered all the statistics for correctly estimating how much memory or disk space our server should require, there are times when it is necessary to define a hard limit to prevent Samba from grinding to a halt. It can also be helpful to set a ceiling

at the point where performance starts to degrade due to resource exhaustion. Of course, this point in time can be determined only with experimentation.

Table 23.1 presents several `smb.conf` parameters that can be used to define hard limits on things such as total accepted sessions or the maximum print jobs supported by a single printer.

TABLE 23.1 Parameters for Specifying Hard Limits on Connections and Services

Parameter	*Default*	*Description*
`max smbd processes`	0	This global parameter defines a maximum number of `smbd` processes that can be running at a given time. A value of 0 indicates that no limit should be imposed.
`max print jobs`	1000	This service level parameter defines the maximum number of jobs that will be accepted by a print queue at any one time.
`min print space`	0	This service level parameter defines the amount of free space (in Kb) that must be available in the spool directory for a given printer share. A value of 0 indicates no minimum value.
`total print jobs`	0	This global parameter limits the total number of jobs across all print services. A value of 0 represents no limit.

If you know that the server has enough memory to support only 100 clients, then you can prevent any more connections after this number has been reached by adding the following line to `smb.conf`:

```
max smbd processes = 100
```

The parameters of Table 23.1 should be viewed as a safety net to prevent the server from being overwhelmed. As a general rule, these limits should not be met on a daily basis. It is much better to scale the server to meet the performance requirements and use these options as a last line of defense.

Oplocks and Performance

Opportunistic locks have already been covered in Hour 8, "Samba—The File Server." They are being mentioned here because of the benchmarks that support their performance advantages—as much as 30% increases, in some cases.

Samba supports both Exclusive and Level II oplocks by default. Both of these parameters, `oplocks` and `level2 oplocks`, should be left enabled unless there is a specific rea-

son not to do so. Any such reason will be gained from experience or an intimate knowledge of an application or the client OS.

The `socket options` Parameter

A socket is the most common programming interface used in TCP/IP applications. Tweaking Samba's global `socket options` parameter is as much an art as it is a science—even more so than estimating how much hardware our server needs. This option is the primary means of tuning sockets within Samba and accepts any combination of the following values:

- `SO_KEEPALIVE`
- `SO_REUSEADDR`
- `SO_BROADCAST`
- `TCP_NODELAY` (default)
- `IPTOS_LOWDELAY`
- `IPTOS_THROUGHPUT`
- `SO_SNDBUF` = integer
- `SO_RCVBUF` = integer
- `SO_SNDLOWAT` = integer
- `SO_RCVLOWAT` = integer

The benefits of each option depend upon our network topology and our servers. To define a reasonable non-default setting requires us to understand the details of the TCP/IP protocol, such as what is the Receive Window and what would be the effect of changing it. Such a topic is beyond the scope of this book, but I would encourage you to read the man page for the `setsockopt(2)` function for your system and to refer to the `smb.conf(5)` man page, as well. The default setting enables the `TCP_NODELAY` option and nothing else.

The last four options accept an integer value. For example, you could specify `SO_RCVBUF = 16384`. To enable the TCP No Delay feature and define the Default Receive Window, we would add

```
[global]
   <...>
   ## enable TCP No Delay and set the TCP Receive
   ## Window to 16Kb
   socket options = TCP_NODELAY SO_RCVBUF=16384
```



Sams' TCP/IP Primer Plus provides a good overview of TCP/IP networking. For specifics regarding Microsoft's TCP/IP implementation in Windows NT, see Dr. Karanjit S. Siyan's book entitled *Windows NT TCP/IP* (New Riders). For the more ambitious reader, Richard Stevens's *TCP/IP Illustrated* three-volume series has long been considered the textbook for studying TCP/IP.

High Availability and Fail Over

In some environments, a single Samba server is not able to meet all the needs of users. The terms high availability (HA), load balancing, and fail over are often mentioned in relation to large Samba installations. HA and load balanced solutions often involve clusters of machines that share the load of providing a particular service. If one node in the cluster fails, it is not noticeable to users because the system load is distributed among the remaining nodes. Fail over capabilities are implemented by frequently moving the services provided by one server onto another machine in the event that the first server crashes.

The CIFS protocol requires the server to maintain certain information regarding the state of the client connection. This makes it difficult to fully support any of these solutions automatically. Many vendors have made a name for themselves by providing the expertise to create these highly scalable solutions. Our goal during this section is to examine what features Samba provides for designing our own fault-tolerant solutions.

We have already covered one feature, new to Samba 2.2, which can be used to distribute clients among multiple servers. In Hour 14, when we learned how Samba could replace a Windows NT 4.0 member server in a domain, I explained how to configure our server to support the Microsoft Distributed File System. Part of this support included providing a list of servers that provided access to the copies of the same data. When creating the MS-DFS link on the file system, multiple servers were separated by a comma, as shown here:

```
$ ln -s msdfs:server1\\docs,server2\\docs documents

$ ls -l documents
lrwxrwxrwx  1 jerry users  31 Oct 3 12:59 documents ->
     msdfs:server1\docs,server2\docs
```

How do we ensure that `\\server1\docs` and `\\server2\docs` contain the same files? There are several possibilities for replicating a directory tree from one server to another. A favorite among Unix administrators is to use `rsync` over `ssh`. We were briefly introduced to `rsync` last hour, during our discussion of replicating the `smbpasswd` file and `[netlogon]` file share. Figure 23.3 illustrates how `rsync` and MS-DFS can be used to implement Samba replicas.

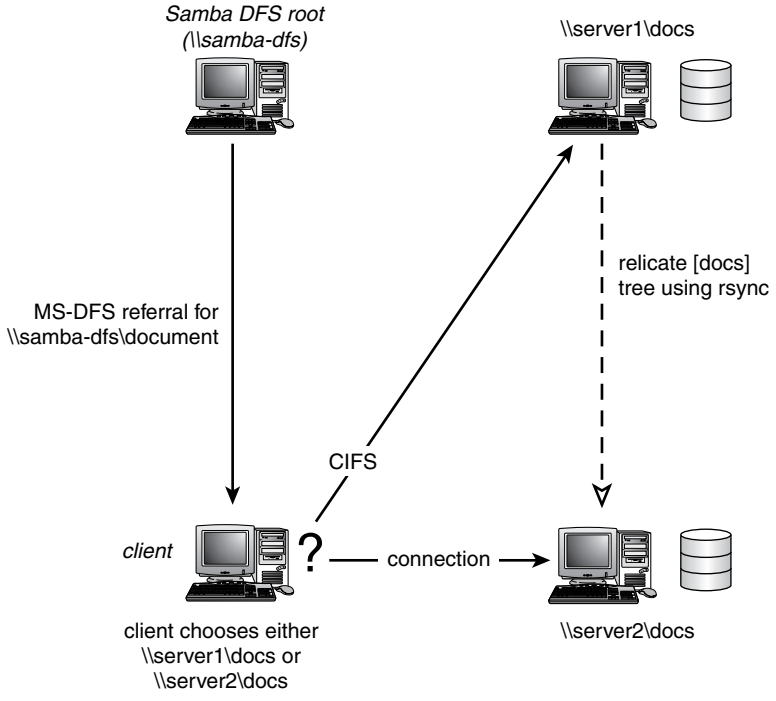


FIGURE 23.3
Using `rsync` *and MS-DFS to support load balancing for clients.*

Sometimes a solution calls for placing multiple CIFS servers on a single host. A solution for implementing one variation of this is to use the `netbios aliases` and `include` parameters as covered in Hour 15, "Server-Side Automation."

There are times when it is necessary to run multiple `smbd`/`nmbd` installations on the same host. By default, the Samba daemons will use all functional network interfaces on the server. If we wish to restrict this to a subset of those interfaces so that each Samba installation can bind to a particular NIC, the `interfaces` and `bind interfaces only` global parameters must be defined.

The `interfaces` option allows us to specify a list of network interfaces that should be used by Samba. This list can take the form of

- Interface names, such as `eth0`
- IP address and network mask pairs, such as `192.168.1.75/255.255.255.0` or `192.168.1.75/24`

Each interface in the list is separated by whitespace, tabs, or a comma.

The `bind interfaces only` Boolean option affects `smbd` and `nmbd` in different ways. When enabled, `bind interfaces only` causes `smbd` to reject all packets not arriving from one of the entries in the `interfaces` list. The `nmbd` daemon, however, must service broadcast packets as well as those destined for the Samba server's IP address. `Bind interfaces only = yes` causes `nmbd` to ignore all broadcast packets that do not match the broadcast address of one of the valid network `interfaces` defined for use in `smb.conf`.

Both of these parameters can be used in conjunction to isolate Samba servers on the same host as one another. For example, we can define an `smb.conf` for a server named `HELLO` to exist with another one named `GOODBYE`. The `lock directory` is used to give each Samba server its own set of database files for maintaining state information.

```
## smb.conf for HELLO
## must play fair with other Samba servers on the same host
[global]
   netbios name = HELLO
   workgroup = GREETINGS

   ## needed to keep things separate
   lock directory = /var/spool/locks/%L
   interfaces = eth0
   bind interfaces only
```

We can then define a second server on the same host, but bound to the network interface `eth1`.

```
## smb.conf for GOODBYE
## must play fair with other Samba servers on the same host
[global]
   netbios name = GOODBYE
   workgroup = SALUTATIONS

   ## needed to keep things separate
   lock directory = /var/spool/locks/%L
   interfaces = eth1
   bind interfaces only
```

There is no coordination from one configuration file to the other except that the `lock directory` and `interfaces` do not conflict.

What does this have to do with fail over? Figure 23.4 shows how we can again use `rsync` and these new parameters to quickly move services from one Samba host to another. When `\\SERVER-A` fails, its services can be manually restarted by a network administrator on a spare network interface on `bighost.plainjoe.org`. However, there is no means of maintaining the state information, such as file locks, from the crashed server when restarting the services on another Samba host.

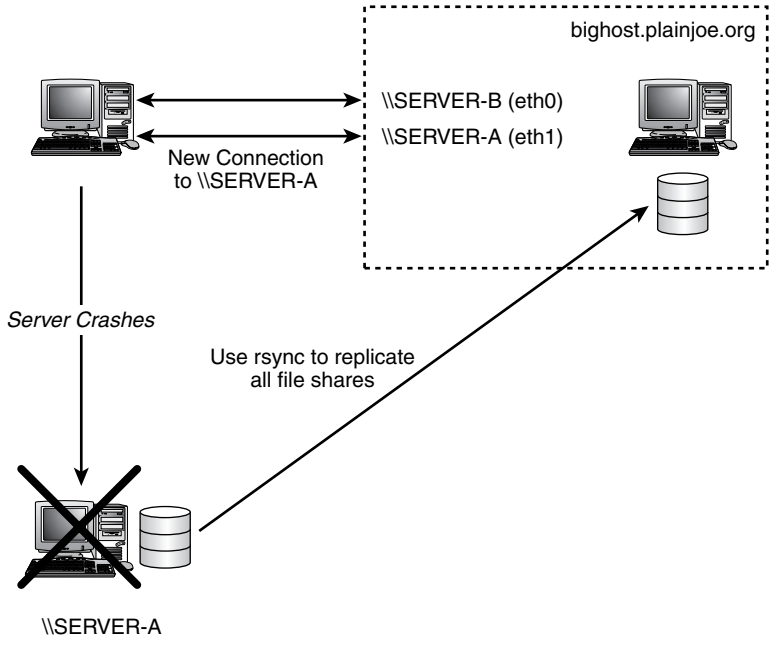


FIGURE 23.4
Using `rsync` *and the interfaces parameter to implement fail over capabilities for a Samba installation.*

Summary

Capacity planning and performance tuning are ongoing efforts during the life of a Samba server. The former helps us to estimate the hardware requirements of our server, whereas the latter helps us to improve the end user experience by making our server more responsive.

The basic formula for calculating how much memory our server needs is

```
(avg_conn + (total_clients-avg_conn)/2) * smbd_mem_usage = est_req_mem
```

Sometimes good plans need a backup. Samba provides several parameters, such as `deadtime` or `max smbd processes`, that can help to act as a safety net and prevent the server from becoming overloaded or slowly dying as performance degrades.

Sometime a single Samba server is not enough to serve all the needs of our clients. We learned about some of the solutions that can utilize Samba configuration options and standard Unix tools to provide Samba solutions that scale well beyond a single machine. Implementing HA and fail over Samba solutions requires a lot of work. However, using features such as MS-DFS support and the `interfaces` parameter can allow for some creative solutions.

Q&A

Q **Does average file size impact speed and performance? What if I store lots of engineering drawings; does that mean I need more RAM?**

A No. This tends to be bound by network bandwidth, the client's processing speed, and the client application. However, Windows 2000/XP clients do support enhanced operations for reading and writing data in larger chunks than Windows 9x/ME/NT, which can result in substantial performance boosts. Samba 2.2.2 introduced the `large readwrite` parameter to control whether Samba should support these new commands. Leaving the parameter enabled presents no detrimental effects for non-Windows 2000/XP clients.

Q **If my network is generally 100 MB ethernet, does a gigabit card help it in any way?**

A Determining the bottleneck for performance requires a good deal of research. For example, increasing the available network bandwidth will do no good if the main problem is disk I/O on the server. The `vmstat(8)`, `iostat(8)`, and `netstat(8)` commands are provided with most versions of Unix and can provide a good starting place for gathering information on a server's behavior.



HOUR 24

Exploring the Samba Community and Looking Down the Road

Samba 2.2 contains huge advancements compared with previous versions. An obvious question to ask next is, "What does Samba's future have in store for us?"

For the final hour of this book, I will talk about development plans in Samba, how the Samba Team operates, and how you can get involved. This hour is akin more to a behind the scenes look at Samba development than to the tutorials presented in the previous hours.

Who Is the Samba Team?

The best place to begin is with the people who develop and support Samba. The Samba Team is a loosely connected group of people who have

write access to the Samba CVS repository. Some of these people are supported by companies to develop Samba, whereas others do so out of personal interest.



CVS stands for *Concurrent Versions System* and is a software package for supporting access to files by members of a distributed group, such as the Samba Team. More information about CVS can be found at `http://www.cvshome.org`.

In Hour 1, we learned how the Samba project was begun by Andrew Tridgell. Through the years, Andrew has been joined by a number of people who work on various features, documentation, and testing. A complete list of team members (and pictures) can be found at `http://samba.org/samba/team.html`. Some of these members have been more active in the past than in the present. Nevertheless, Team membership has historically been for life.

Who Guides Samba Development?

In the arena of Open Source, source code has a tendency to speak louder than words, and good code speaks volumes. This of course does not lessen the value of anyone's contributions. It simply means that, ultimately, decisions are made by the people willing to put the time and effort into supporting their ideas. However, bad ideas are still bad ideas no matter how much code is written to support them. So in this respect, Andrew Tridgell and Jeremy Allison act as the benevolent dictators when reviewing submissions.

Several things make Samba development difficult for newcomers. The first is that the CIFS protocol is horribly messy and also that Samba contains a great deal of history in its code base. More than once I have observed someone "fix" something that broke a particular client or feature. CIFS is not something you dive into over the weekend and come out an expert in three days.

The second problem is the size of the Samba code source. Although it is only approximately 220,000 lines of code, understanding the architecture of Samba can be quite difficult. It is important to understand how one change will affect the rest of the code base. This means that only someone with a good feel for how Samba works is able to review new code patches.

How does all of this relate to the steering of Samba's development efforts? The samba-technical mailing list should be considered the pulse of development efforts. It is also the best place for asking questions when trying to make modifications to the Samba source code.

In past experiences, proposals for changes or new features can come from two places. Frequently, enhancements or fixes are suggested by community members. Though technically not a part of the Samba Team, these people have normally spent a good deal of time understanding how the code works. These proposals generally come in one of two forms. The first variation is when a new feature is suggested, such as a Network Trashcan for Samba file shares. At this point, the debate begins whether it is a good idea at all and, if it is, whether it should be implemented inside of Samba or outside. Proposals can also take the form of a source code patch or bug fix for a currently broken feature. This time the discussion revolves around whether or not the patch was needed and, if so, whether or not it solves the problem correctly.

The acceptance or rejection of a proposal should always be based upon the idea's technical merit and relation to the general Samba development roadmap. So far, this has proven to be successful. However, it does require that technically competent people are involved in the discussion.

If the idea or patch is a good one, hopefully it has generated enough noise to catch the attention of a Samba Team member. This is a requirement for the proposal to reach the next layer on its way to being integrated into the main Samba distribution. This is also a potential weakness in the development model. If team members are overloaded with other tasks, such as preparing for the next release or their day jobs, it can be difficult for them to respond to the discussion in a timely manner. This can give the appearance of unresponsiveness and frustrate those who initiated the debate.

Specifically, Samba has suffered from this ailment when attempting to address messages sent to the `samba-patches@samba.org` mail alias. Efforts are underway to fix this, but this is one of the hurdles that all Open Source projects face when supporting a global community.

There are two possible outcomes once a proposal has gained the attention of a Samba team member. He (or she) can reject the idea outright in a polite fashion, or the team member can champion the proposal and therefore shoulder the responsibility of presenting it to the remaining Samba developers. This is the second level of the approval process.

If the majority of the active Samba Team members deem the feature or patch as acceptable, it is placed on the list for inclusion in future Samba releases. Of course, new proposals can be generated directly by Samba Team members. In this case, the idea has already reached the second stage of the approval process.

This development model can succeed only if Team members are willing to work together and compromise when necessary. Although all of this makes the Samba development

sound like a democracy (and it is, for the most part), there are times when executive decisions must be made regarding the project's direction or scope. During these times, the fate of Samba rests in the hands Andrew Tridgell, much in the same way that the Linux kernel currently rests in the hands of Linus Torvalds.

Figure 24.1 attempts to capture all of this by illustrating the Samba development process.

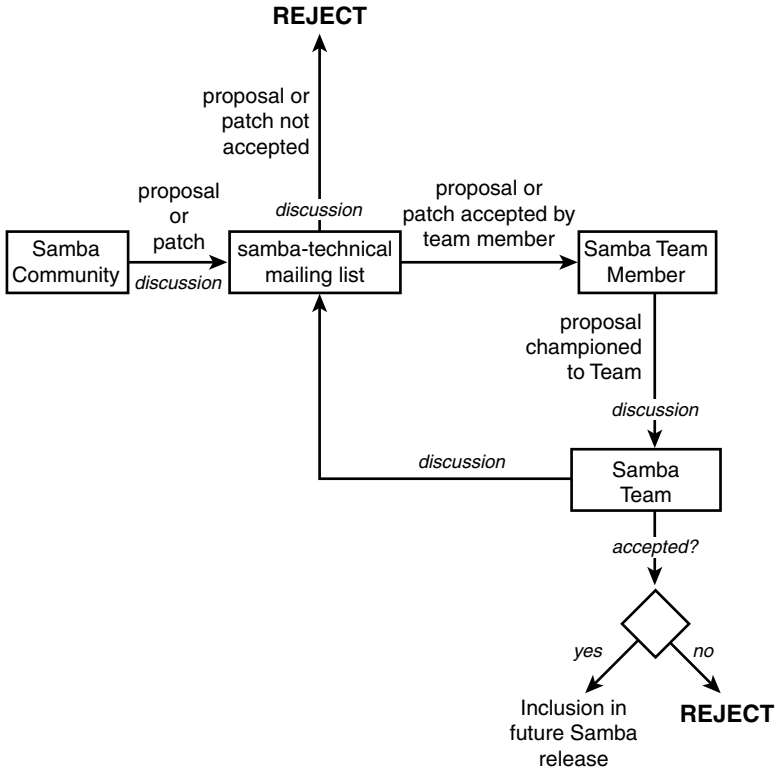


FIGURE 24.1
The dynamics of Samba development.

Future Plans for Samba

Now that we have a general understanding of how Samba development moves forward, we may be wondering what the plans are for the next major release. Writing software can be compared with driving through the states of Nebraska or Kansas. Because the landscape of these states is generally flat, determining exactly how far an object on the horizon is from your current location is extremely difficult. Samba developers and those with other Open Source projects know this as well and have adopted a release-when-it's-ready policy.

This means that we can discuss the proposed features of future releases but not the timeframes. With Samba, a timeline is difficult to keep due to the nature of network reverse engineering itself. It is much like the exploration work done in the old west by people like Meriwether Lewis and William Clark. Sometimes one piece of work unearths more requirements than initially anticipated. Nevertheless, there are some concrete plans to support certain features in the next major Samba release, tentatively named 3.0.

Samba 3.0

A major milestone of the Samba 3.0 release will be the inclusion of improved support for non-English CIFS clients. Hour 5 covered the DOS code page and Unix character set support in the 2.2 series. In an effort to move away from the code page solution to internationalization, Microsoft has implemented support for UCS2 UNICODE in its current clients and servers.

It would take a great deal of time to fully discuss all of the issues surrounding UNICODE and international clients. In simple terms, UCS2 has an advantage over a DOS code page in that the former can represent every character within the same space. A code page is specific to the particular set of characters for a language such as Arabic or Spanish. UCS2 uses two bytes (16-bits) to represent each character. This gives it the ability to represent 2^16 (or 65536) unique characters. ASCII uses only 8 bits per character, which yields 256 unique characters. The first 127 of these are made up of English symbols and characters. Therefore, a code page (which uses the ASCII character set) sometimes has to represent a character using more than one byte. UNICODE solves the problems of these so-called multi-byte code pages by allowing all characters to exist within the same set—UCS2. These examples have been simplified a great deal, but they do at least allow you to see how the addition of internal UNICODE support will advance Samba a great deal. It will no longer be necessary to know the code page used by a client in advance and manually configure the server to support them.

Within the same topic of internalization, Samba 3.0 will include support for documentation translated into languages other than English. Efforts have already begun to convert Samba's various help files into Polish, Japanese, and Turkish. In particular, SWAT will integrate the changes to make the help files readily available to administrators. Documentation will be split up by language packs with only English being distributed in the Samba source archive. This allows administrators the flexibility to install only those languages needed.

Finally, Samba 3.0 will close the lid on our Windows NT 4.0 Primary Domain Controller implementation by providing the missing

- Ability to synchronize SAM databases with a Windows NT PDC or BDC.
- Capability to participate in Windows NT 4.0–style trust relationships as a Domain Controller.

There are many other less visible improvements that will also be in Samba 3.0, such as better error checking and handling for Windows NT Status Codes and a completely rewritten internal authentication subsystem. The latter will make it immensely easier to support authentication modes such as PAM and NTLMv2.

Beyond 3.0

Beyond Samba 3.0, the issue of Windows 2000 domains looms ominously on the horizon. Supporting a true Windows 2000 domain, either as a member or as a domain controller, involves three components in addition to Samba:

- The ability to communicate with, or possibly replace, the Windows 2000 Active Directory Service
- The ability to validate users using Microsoft's Kerberos implementation
- A Windows 2000–compatible Dynamic DNS implementation

Many of these pieces are available individually. Efforts are being focused on how to integrate them for the purpose of creating a replacement to a Windows 2000 Domain Controller.



To show how fast things can change in the world of Open Source, during the five weeks between the first draft of this chapter and the final review process, Tridgell has implemented some initial support for Kerberos in `smbclient` and `smbd`. Developers are preparing Samba 3.0 for alpha releases soon. This will most likely be a long pre-prelease process with the goal of getting as much testing as possible from the Samba community. Samba 2.2 will be the current release branch for several months to come.

How Can I Help?

Becoming involved in an Open Source project at any level is a thrilling ride. Here are a couple of words of advice if you are interested in helping the Samba project.

First, join the samba-technical mailing list and check the pulse of current discussions. Make known your willingness to volunteer. Also, it is a good idea to have some ideas about what you would like to work on. Remember, code and patches speak louder than words, even patches to documentation. It is a shame, but I have seen many people volunteer only to never be heard from again.

Secondly, we understand that not everyone has 40 hours a week that they can devote to developing Samba. It is important to have fun with what you do. I have often found that the best motivator is when you have your own itch to scratch, as Eric Raymond puts it. If there is something that is not quite working for you, don't be afraid to dive in and find out why. If it is a bug, fix it. If the documentation was clear enough, make it more specific. Samba is a large project, and there is room for many people.

The key to making community development work is to coordinate efforts among volunteers. The Samba Team is happy to help coordinate among self-motivated individuals. No one will hunt you down (at least not in a bad way) if you volunteer for something and then cannot finish it. Of course, if you do this too often, people will begin to ignore your offers to help. But by all means, "Forge ahead!"

Summary

Throughout past hours, we have examined many aspects of Samba, all the way from "What is it?" and "How do I get it?" to "How can I make Samba act as a Windows NT PDC?" During this hour, we have talked about Samba's future and what features are in the works for Samba 3.0. We all look forward to better internationalization support and a full-fledged Windows NT PDC replacement. Beyond that, plans have been made for Windows 2000 domain integration. But as John Steinbeck once wrote, "The best laid plans of mice and men. . . " This project relies upon a community for feedback and support. Who knows? Perhaps you could be the next Andrew Tridgell or Linus Torvalds?

Q&A

Q Where can I find out about the latest plans for Samba development?

A The best place to start is the Samba Web site, `http://samba.org`. In particular, see the `http://samba.org/samba/development.html` page for the latest roadmap. However, if you really want to find out what people are thinking, you should check the various mailing lists given in Hour 13, "Troubleshooting Techniques."

Q I have an idea for a really cool feature to add to Samba. What should I do with it?

A The preferred thing to do would be to implement it first and then send the patches to samba-`patches@samba.org`. If for some reason you are unable to do this, the next best thing is to suggest it on the `samba-technical@samba.org` mailing list. Perhaps someone has thought of the same thing and is already working on it. Samba-technical is the place to coordinate new development efforts in Samba.

Q What is Samba TNG, and how is it related to the Samba we have been talking about?

A Samba TNG (The Next Generation) was initially a Samba development branch. Over time, the people working on this branch became interested in continuing their work as a separate project and so formed the "Samba TNG project." This is what is known as a code fork in Open Source development. Other famous code forks include the emacs/xemacs projects and the gcc/egcs fork.

Samba developers and Samba TNG developers swap bits of code and ideas on development mailing lists, and both projects benefit. More information on Samba TNG can be found at `http://www.samba-tng.org`.



INDEX

A

ACL (Access Control Lists), 248-249, 274
Add Printer Wizard (APW) program, 144-146
adding/deleting printers, 151-152
add share command, 268
add user script, 384
adding print drivers, 144
authentification, 28, 312–314
autoconf program, 39

B

backup browser, 350
BDC (Backup Domain Controller), 390-393
binary distribution methods, 43-44
BIOS, definition, 16
broadcast name resolution, 32
broadcasts, directed, 359
browse list, 340, 350
browse servers, 340, 350
browsing
cross subnet, 351-354
election, 350
examples, 347
local subnet, 340-343
network, designing, 354-357
parameters, 343-344, 359-360
problems, 348
troubleshooting remote, 358-359
with OS/2 clients, 344-345

C

chgrp command, 49
chmod command, 49, 312
chown command, 312

CIFS (Common Internet File System), 25
clients for Unix, 193
code pages, 77-78
command line arguments, 55-56
comment field, 177
compiling Samba,39-42
compromised system, 314
configure command, 39-41
autoconf system, 39
help option, 39-41
fhs option, 41-42
configuring
file permissions, 314-318
Linux server, 249-255
creating new shares, 262
mapping domains, 256
testing the server, 265
printers, 140-153
Samba PDC, 379-380
smb.conf, 47-48
configuration files, 46-47
security settings, 315
configuration options, 40-41
connection-oriented protocol, 26-27
connection requests, validating, 95-96
cross subnet browsing, 351-354
Cygnus Source Navigator, 239

D

daemons
inetd, 52-53
nmbd, 52-56, 226
rpc.pcnfsd, 10
smbd, 52-56, 224-226
syslogd, 282
winbindd, 256, 257, 259
xinetd, 53-54
default initialization files, copying, 277-278
delete share command, 268
designing network browsing, 354-357
DHCP (Dynamic Host Configuration Protocol), 175
directories
backing up, 206-207
directory names, printer driver, 145
disable DHCP, 169
DMB (Domain Master Browser), 353
DNS (Domain Name Service), 17
Domain Admins group, 387
Domain Guests group, 387
domain controllers
defined, 32
backup (BDC), 390-393
primary (PDC), 377-380
setting up, 364-367
Domain Name Service (DNS), 220
domains, 30-31
versus workgroups, 364
DOS attributes, 125-126
relate parameters, 125
dumb terminal, 29

E

etc files
exports, 49
group, 49
hosts, 138, 212
inetd.conf, 53, 82-83
nologin, 296

nsswitch.conf, 261
pam, 294
pam.d, 294, 296
passwd, 104-105, 109, 260, 290-307
password, 49, 84
printcap, 154-156
security, 296
services, 53, 82
shadow, 84, 105
smbpasswd, 107-108, 289-307
swat.conf, 83
xinetd.conf, 53-54
ethereal, 234-236
ethereal program, 104
ethernet, 234

F

fail over support, 404-407
file permissions
configuration data, 314-316
Unix, 118
file sharing/serving
access control lists, 124-125
basic group share, 116-119
file system differences, 119-121
home directories, 132-135
restricted access share, 121-125
public share, 126-132
file systems
case sensitivity, 119-121
DOS 8.3 filenames, 120
quotas, 135
file locks
share mode, 130
deny mode, 130
oplocks, 131
files
backing up, 37-39
getting, 204-206
lmhosts, 333-334
log, 231-238
managing on remote servers, 194
printing, 207-208
putting, 204-206
Samba configuration, 38, 46-47
Samba-specific user account, 39
firewalls, 321-323

G

gcc compiler, 2
getent command, 261-262
getfacl tool, 253
GNU GPL (General Public License), 12-13
GPL Preamble, 13
Graphical User Interface (GUI), 181
grep command, 2, 232-234
group shares, 116-135
access control lists (ACL), 124-125
DOS attributes, 125-126
file system quotas, 135
home directories, 132-133
oplocks, 131-132
public, 126-132
restricted access, 121-124
symbolic links, 134-135
gzip command, 36
guest access, 111-112
guest account, 112
guest ok, 112
only guest, 112

H-K

hard limits, 402
high availability (HA) support, 404-407
HUP signal, 83, 88

ifconfig command, 221
inetd daemon, 52-53, 81-82
installation, testing, 56-58
Integrated Development Environment (IDE), 239
Internet worm prevention, 318-319
internationalization, 413
IP Address, 169
ipconfig.exe tool, 222

kill command, 2

L

LanManager, 313
passwords, 313
Linux
compilation, 250-252
make xconfig tool, 251
download site, 250
kernel-HOWTO, 251-252
smbfs system, 194
Virtual File System (VFS), 194
Linux/Samba server
configuration
ACL, 252-255
ACL patches, 249-251
docs section, 263
PAM module, 257
user section, 262
Winbind package, 256-257
MS-DFS options, 270-272
MS-RPC functions, 267
NT server replacement, 247, 255-267
testing, 265-267
Windows management tools, 267-270
MMC, 268-270
LMB (Local Master Browser), 353
lmhosts file, 316, 333-335, 337
lpr command, 138
log levels, 73, 232
logging, 72-76

M

machine trust accounts, 380, 394
creating on a Samba PDC, 381-384
make command, 2
xconfig option, 251
make install command, 42
man pages
inetd, 82
inetd.conf, 82
nsswitch.conf, 261
smb.conf, 96, 141
smbmount, 197
managing
shares, 267-270
user accounts, 291
Microsoft Network Client Version 3.0, 165-173
Microsoft Network Monitor (netmon), 236-237
mkdir command, 49
mksmbpasswd tool, 108

modify_samba_config script, 268
MS-DFS (Microsoft Distributed File System), 270-272
Windows 98/ME, 270
Windows 95, 270
Windows NT/2000/XP, 270
MS-DOS
client installation, 166-170
setup options, 168
network boot disk, 171-173
autoexec.bat, 171-173
config.sys, 171-172
DOS directory, 171
NET directory, 171
obtaining software for, 165-166

N

name resolution, 22-23
name service, 17
names,
called, 32
calling, 32
NBT, 16
nbstat.exe command, output, 20
nbstat.exe tool, 20-21, 227
NDIS2 (Network Desktop Interface Specification), 180
Net BEUI (Net BIOS extended user interface), 16, 32
NetBIOS (Network basic input/output system)
aliases, 283-284
broadcast name resolution, 22
datagram services, 25
defined, 12, 32
hostnames, 211-212
machine name, 177, 185
name registration/resolution, 18-19, 22, 237-335
broadcast resolution, 23
node types, 23
point-to-point resolution, 23
name characters, 18
names and workgroups, 188-189
name servers (NBNS), 23
name services, 13
nbstat.exe tool, 20-21, 33
resource types 19-20
scope, 21, 33
session services, 24
workgroup names, 177, 185
net.exe command, 230
netlogon, 365
net view command, 230
networks
name services, 13
redirectors, 164
basic, 168
full, 168
subnets, 352
TCP/IP, 17, 352
network adapter. *See* Network Interface Card
network browsing, 340-348, 354-357
backup browser, 341
browse list, 340
browsing election, 343-344
election criteria, 342
master browser, 341-343
network neighborhood, 340
OS/2 clients, 344-345
preferred master browser, 342
server, 340
troubleshooting, 348, 358-359
Network Interface Card (NIC), 174, 182

network redirector, 164
basic, 168
full, 168
Windows NT/2000, 189-190
network traffice, monitoring, 234-238
NFS (Network File System), 10-11
Nimda worm, 318. *See also* virus preventions
NIS/NIS+, 108
nmbd daemon, 52-56
starting from inetd, 52-53
starting from shell, 54-56
command-line arguments, 55-56
starting from xinetd, 53-54
nmblookup tool, 56-57, 226
NTLM, 114
authentication, 313

O

Open Source Software (OSS), 3, 414
Open SSH server, 317
Open SSL libraries, 319-320
oplocks, 132, 402-403
OS/2 clients, browsing with, 344-345

P

PAM
control-flags, 295-296
modules, 295
smb.comf parameters, 297
support in smbd, 296-299
pam_smbpass, 304-307
packet sniffer, 241
packaged code pages, 77-78
panic action command, 238-239
parameters
autoservices (preload), 345
browseable, 117
browsing, 343-344, 359-360
character set, 78
client code page, 77
case, 121
case-mangling, 121
case sensitive, 120
default, case, 120
preserve case, 120
short preserve case, 120
code page directory, 78
comment, 117
common global, 70-72
netbios aliases, 71
netbios scope, 72
create mask, 118
deadtime, 400
debug timestamp, 74
debug pid, 74
debug uid, 74
directory mask, 118
dns proxy, 332
enhanced browsing, 360
force create mode, 118
force directory mode, 118
guest related, 127
hosts allow, 323
hosts deny, 323
include, 282-286
inherit permissions, 118
limitation, 402
load printers, 345
lock directory, 79
log file, 72
logon drive, 387
logon home, 387

logon path, 387
log level, 73
map to guest, 112
max log size, 75
message command, 281-282
name resolve order, 334
password-related, 314
password level, 105
path, 117
postexec, 117
preexec, 276
print hook, 141
printing, 143
protocol-related, 314
read only, 117
remote browse sync, 360
root postexec, 279
root preexec, 279
security, 95
domain, 96
share, 97-99
server, 99-101
user, 96-97
smb.conf, 70
socket options, 403
syslog, 75
syslog only, 76
username level, 102-103
username map, 103-104
wins proxy, 331
wins server, 330
wins support, 331
password changing, 291
password encyrption, 105-107
password chat script, 292-293
password-related parameters, 314
passwords
encrypted, 104-105, 178
hashes, 109-110
irreversible, 105
LAN Manager encryption, 105
null, 108-109
PAM (Pluggable Authentification Module), 293-307
control-flags, 295-296
modules, 257, 295
pam_smbpass, 304-307
pam_windbind, 299-304
password levels, 105
plain-text, 104, 178
plain-text equivalent, 105, 114
smbpasswd file, 107-108
synchronization, 290-299
update encryption option, 109-110
Windows NT encryption, 105
PC-NFS, 10-11
file locking problems, 10-11
lack of native (PC) support, 11
licensing fees, 11
pcnfsd daemon, 10
PDC (Primary Domain Controller), 377-380
configuring for Samba, 379-380
peer networking. *See* workgroups
permissions (Unix)
modifying from Windows, 128-130
parameters that affect permissions, 118
take ownership, 129
ping command, 17, 219-221
plain-text equivalent password, 114
point-to-point name resolution, 33
poledit.exe command, 373
postexec script, 276-278
preexec script, 276-278
printer drivers, 144-151
adding, 144
creating, 151-152
defining, 148
deleting, 152
directory names, 145
download sites, 144, 148

installing, 150
local drivers, 152-153
printer sharing/serving
access controls, 156-158
Add Printer Wizard (APW) program, 144-146
adding/deleting printers, 151-152
overview, 138-139
printer drivers, 144-151
directory names, 145
download sites, 144, 148
local drivers, 152-153
printing from Unix to Windows, 209-211
printing systems supported, 143
troubleshooting, 157-158
printer shares
securing, 156-157
troubleshooting, 157-158
printer variables, 142
printing
support, 138-139
from Unix to Windows, 209-211
protocol-related parameters, 314
ps command, 2, 224

R

registry editor, 370
regedit.exe command, 373
Relative ID (RID), 381
Remote Procedure Calls (RPC), 137
replacing Windows NT with Linux/Samba, 255-267
resource types, NetBIOS, 19-20
RFC (Request for Comments)
RFC 1001, 16
RFC 1002, 16
root preexec command, 279-280
rpcclient tool, 144
rpc.pcnfsd daemon, 10

S

Samba
ACL (Access Control Lists) support, 248-249
administrative tools, 55
bugs, 124, 259
capabilities, 9-10
client tools, 60
commercial, 194
free BSD, 194
Macintosh, 194
smbclient tool, 193
smbfs system, 194
compiling, 39-42
configuration files. *See also* smb.conf file
security settings, 315
configuration options, 40-41
described, 8-10
diagnostic tools, 59-60
documentation, 217
domain controller support, 364-375
logon script, 365-366
sample smb.conf file, 366-367
troubleshooting, 368-369
Windows 9x setup, 367-368
download sites, 36
fail over support, 404-407
future plans for, 412-414
high availability (HA) support, 404-407
history of, 7-8
HOWTOs, 217
installation, 36-44

internals, 238-239
internationalization support, 76, 413
ldapsam option, 308-309
licensing agreement, 12-13
nisplussam option, 308-309
nmblookup tool, 56-57
nss module, 211-212
Open Source Software, 3, 414
packaged code pages, 77-78
PDC, configuring, 379
security features, 95-101
server tuning, 399-404
deadtime, 401
msrpc functions, 400-401
oplocks, 402-403
smbclient tool, 56-58
smbstatus tool, 60
supported platforms, 12
system requirements
CPU, 399
RAM, 395-399
tdbsam option, 308-309
unicode support, 76, 413
version 2.0, 259
version 2.2, 37, 259, 296-297, 404, 409
version 3.0, 413-414
Virtual File System for shares, 286
Samba development
Open Source Software, 410
development team, 409-412
Samba installation, 36-44
binary distributions
Red Hat, 43
rpm package, 43
Solaris, 43
Debian, 43
compiling from source code
configure command, 39-41
default options, 40-41
help option, 39-41
fhs option, 41-42
directory tree location, 42
extracting source files, 36
make install command, 42
SWAT option, 82
version 2.0
version 2.2
Samba team, 217
scopy.exe tool, 263
Secondary Samba Servers, 316
secrets.tdb file, 316
security
authentication, 312-314
domain mode, 101
guest access, 111-112
levels, 95-101
mode vs. level, 96
passwords, 104-111
encrypted, 104
hashes, 109
irreversible, 105
LAN Manager encryption, 105
null, 108-109
PAM module, 257
password levels, 105
plain-text, 104
plain-text equivalent, 106
smbpasswd file, 107-108
update encryption option, 109-110
Windows NT encryption, 105
session setup request, 98, 101-102
server mode, 99-101
Unix uid, 100-101
share mode, 97-99
Windows 9x, 97-98
tips, 311-312
user mode, 96-97

user names, 101-104
username level, 102-103
username maps, 103-104
server-side automation, 275-276, 287
server tuning, 399-404
servers, 340
metadaemon
inetd, 82-83
xinetd, 83
print, 137
prerequisites, 138
tuning, 399-404
upgrading, 37-39
session service, 24
setfacl tool, 253
setuid script, 280
shares
defined, 47
group directories, 48-51
managing on a Samba member server, 267-270
parameters for controlling access to, 123
printers, 153-156
securing printer shares, 156-157
public access, 126-128
viewing a list of, 345-346
shared group directories
setting up, 48-51
sites,
for downloading Samba, 36
smbclient tool, 56-58, 144, 201
command line options, 202-203
file printing option, 207-208
get command, 204-206
internal commands, 203-204
options, 202-204
put command, 204-206
tar option, 206-207
smb.conf file
access control parameters, 123
add user script, 384
basic file service parameters, 117
built-in section names, 66
case sensitivity, 66
configuration parameters
add share, 267
auto services, 345
bind interfaces only, 406
change share, 267
character set, 78
client code page, 77-78
code page directory, 78
config file, 285
create mask, 51, 117-118
debug pid, 74
debug timestamp, 74
debug uid, 74
delete share, 267
directory mask, 117-118
domain master, 354
dns proxy, 332-333
directory mask, 51
encrypt passwords, 236
enhanced browsing, 360
force create mode, 50-51, 118
force directory mode, 50-51, 118
force group, 50, 122
force user, 122
host msdfs, 271
hosts allow, 323-324
hosts deny, 323-324
include, 282-285
inherit permissions, 118
interfaces, 226, 405
lanman auth, 313
load printers, 345-346
log file, 72-73

log level, 73-74
logon drive, 388
logon home, 388
logon path, 388-389
log size, 75
map to guests, 111-112
message command, 281-282
msdfs root, 271
name resolve order, 334
netbios aliases, 190
netbios scope, 72
netlogon, 365
network browsing, 343-344
passwd syn, 292
postexec, 276-278
preexec close, 278
preexec, 276-278
print, 142
printer admin, 146
printer drivers, 145
printing, 143
read only, 50
remote announce, 359
remote browse sync, 360
root postexec, 279-280
root preexec, 279-280
root preexec close, 279
socket options, 403
syslog, 75-76
syslog only, 76
timestamps, 126
security parameter, 236
server string, 177
variables, 67-69, 142-143
veto files, 318
vfs object, 286
wins proxy, 331-332
wins server, 330
wins support, 331
domain security parameters, 257-258
DOS attribute parameters, 125
environment variables, 69
global section, 47-48, 67, 116
NetBIOS name, 47-48
security level, 48
workgroup, 47-48
group shares, 48-51
guest parameters, 127-128
hard limit parameters, 402
layout, 65-66
logon script, 365-366
netlogon share, 365
oplock parameters, 132
overview of, 46-47
PAM parameters, 297-298
password parameters, 314
password, changing parameters 90, 291
password chat script, 292-293
passwd tool, 292
password server, 258
permissions, 116-119
permission mask parameters, 129-130
printer configuration parameters, 140-144
printer section, 153-156
protocol parameters, 314
sample file, 66
symbolic link parameters, 134
starting, 52-55
testparm tool, 51-52
variables, 67-69
verifying, 51-52, 223
viewing, 90
smbcontrol tool, 232
smbprint script, 208-209

smbd.conf file
configuration parameters
homedit, 260
template, 260
template shell, 260
panic action parameter, 238-239
smbd daemon, 52-56, 400-401
PAM support, 296
print hooks, 141
starting from inetd, 52-53
starting from shell, 54-56
command-line arguments, 55-56
starting from xinetd, 53-54
smbfs filesystems
mounting, 200
smbmnt tool, 196
smbmount tool, 196
options, 197-198
permissions, 199-200
user mounting, 200
smbpasswd command, 90
smbpasswd file, 316
authentication, 312
multiple server distribution, 316-318
null passwords, 108-109
smbprint tool, 208-209
SMB (Server Message Block) protocol. ***See also*** **Samba**
description, 4
over NetBIOS, 25-26
overview, 27-29
virtual circuits, 26-27
smbumount tool, 196
smbpasswd, 109-111
distributing to multiple servers, 316-318
smbstatus tool, 60
SNIA (Storage Network Industry Association, 25
SSL (Secure Sockets Layer), 319, 320-321
status information, obtaining, 89
Stunnel server, 319-320
STY-Samba workgroup, 185
subnet browsing, 351-354
subnet mask, 169
SWAT (Samba Web Admininstration Tool)
defined, 81
features, 84
global section, 85-86
logging on, 83-84
main page, 84-85
managing file shares, 86-88
managing global sections, 85-86
managing printer shares, 88-89
PAM support, 84
password page, 90-91
securing access to, 319-320
security, 84, 90
status page, 89-90
full view option, 90
using with SSL, 320-321
symbolic links, 134
system.dat, 370
system policies, 389-390

T

tar command, 36
tcpdump tool, 104
TCP/IP protocol, 183, 404
addresses, 17
testing installation, 56-58
testparm tool, 223-224
testprns tool, 155
tips, security, 311-312

troubleshooting, 215-216
enumerating shares from, 229-230
log files, 231-234
name resolution, 228-229
NetBIOS interface, verifying, 227-228
network browsing, 231
network traffic monitoring, 237
nmbd, 226-227
ping command, 219-221
DNS configuration, 220
printer shares, 157-158
pyramid, 216
remote access to shares, 228-231
remote browsing, 358-359
share connecting remotely, 230-231
smb.conf settings, verifying, 223-224
smbd, 224-226
source level debugging, 238-239
test environment, 218-219
testparm tool, 223-224
tools, 217-218
Windows 9x/ME domain logons, 368-369
Windows client, 229

U

unicode, 76, 413
upgrading servers, 37-39
Usenet newsgroup, 218
user.dat file, 370
username map file, 315
usernames, 101-104
user accounts, managing, 291
user connections, recording, 307-308
user home directories, 132-134
user profiles, 370-373, 389-390
password properties tab, 371
postexec script, 372
preexec script, 372
roaming, 371
USR1 signal, 83

V

variables, 67
environment, 69
smb.conf, 68
verifying smb.conf settings, 223
virtual file system (VFS) module, 194, 286, 287
viewing, list of shares, 345-346
Virus prevention, 318-319

W

Webmin tool, 91-93
defined, 91
download site, 91
front page, 92
installation, 91
Samba page, 92-93
security, 92
winbind architecture, 256
winbindd daemon, 256-259
components, 299
Windows 9x/ME
client configuration, 174-177
network interface adapter, 174-175
TCP/IP network protocol stack, 175-176
setting up, 367-368

Windows NT 4.0/2000, domain control, 377-378
Windows NT, 181-182
NetBIOS settings, 185
network adapter installation, 182
security tab, 264
server service, 184
system policies, 389-390
TCP/IP protocol installation, 183-184
user profiles, 389-390
workstation service, 184-185
Windows NT RID, 260
Windows registry, 370
Windows system policies, 373-374
Windows 2000
configuring, 186-188
domains, 393
NetBIOS names and workgroups, 188-189
TCP/IP configuration, 187-188
WINS and, 335
WINS (Windows Internet Name Service), 328-329, 337
broadcast name resolution, 328-329
workgroups, 29-30, 340
defined, 33
versus domains, 364
Worm. *See also* virus prevention
prevention of, 318-319

X-Z

xcopy command, 173
xinetd daemon, 53-54, 83