Linux and Windows Integration with Samba

International Technology Solutions, Inc.
Wake Forest, North Carolina

This page intentionally left blank

Welcome

Welcome to Linux and Windows Integration with Samba

Linux and Windows Integration with Samba introduces you to the concepts and strategies necessary to use Linux and Windows together. Presented as lecture and hands-on labs, this class concentrates on the practical application of Samba server administration, including configuring the shares to provide, restricting access, and tuning performance.

The class's text provides material for in-class discussions and may also be used as an invaluable Samba administration reference.

Course Objectives

Linux and Windows Integration with Samba will teach you:

- to recognize SMB shares and their value in heterogeneous networks.
- how to quickly, correctly, and efficiently create SMB file and printer shares with Samba.
- secure Samba shares with host and user-based access controls.
- troubleshoot common Samba problems.

Course Structure

This course is a two-day, lecture and lab intensive, fast track curriculum. Lectures follow the structure of the class's text, with labs and question and answer sessions woven in after each chapter.

About International Technology Solutions

Since 1994, International Technology Solutions Inc. (ITS) has been providing training and consulting services to Fortune 500 companies such as Alcatel, Blue Cross Blue Shield NC, Cisco Systems, Duke Power, Ericsson Inc, Fujitsu, Lucent Technologies, Nortel Networks, Sprint, and many more.

Our corporate mission is to provide high-quality cost effective technology solutions that increase efficiency and productivity, resulting in a return on investment for our clients.

ITS is committed to providing superior corporate education programs and related services. Our main goal is to increase the productivity of those we educate and show our clients a return on investment.

ITS offers an entire curriculum of Linux courses for the user, programmer or administrator. These include:

- Linux Fundamentals
- Linux bash Shell Programming
- Linux System Administration
- Linux Network Administration
- Linux and Windows Integration with Samba
- Apache Web Server Administration
- Introduction to Linux Development
- Linux Systems Programming
- Linux Kernel Programming
- Linux Device Driver Programming

For these courses, plus many more, please visit us on the Internet at http://www.itsinc-us.com/.

Table of Contents

Chapter 1: Introducing Samba

Chapter Overview

Many computing environments today house a variety of different operating systems, including UNIX, Windows, and System 7. In these heterogeneous environments, configuring each operating system to share files and printers can be a daunting task.

Samba, which this chapter introduces, eases the administrator's burden by allowing Windows users to access Linux files and vice-versa. Emulating Windows' native file-sharing capability makes a Linux Samba server a powerful network addition.

Chapter Objectives

After completing this chapter, you will be able to:

- describe Samba’s functionality.
- explain NetBIOS and the SMB protocol.
- list and describe Samba's components.
- describe share and user-level security.
- control Samba's operation.
- locate Samba logs.

What Samba Does

Samba is an Open Source package that provides Common Internet File System (CIFS) functionality. CIFS is the latest incarnation of the Server Message Block (SMB) protocol, which is the primary means of sharing printers and files between Windows computers.

Linux can act as a SMB/CIFS server, providing secure access to its files and printers from Windows clients. In fact, Samba has been shown to outperform Windows NT servers in file and printer sharing, so using Samba has not only ease-of-use but also performance benefits; as an administrator, these are qualities you strive to achieve.

Samba provides:

- file and printer sharing.
- windows networking log serving.
- Primary Domain Controller (PDC) support.
- domain member support.
- support for Windows browsing.
- WINS support.
- automatic installation of printer drivers under Windows 9x.
- synchronization of passwords between Windows and UNIX systems.

Samba and NetBIOS

NetBIOS, designed originally for extending the PC BIOS, provides the following services:

- Name services allowing computers to locate others by name.
- Session services allowing computers to connect to each other and send data.
- Datagram services providing methods of sending small amounts of data.

NetBIOS name services

Under NetBIOS, each system has one or more NetBIOS names. A NetBIOS name service (NBNS) manages NetBIOS mappings between host names and IP addresses. You can use Samba as a NBNS server.

The NBNS allows hosts to register NetBIOS names and allows look-ups to translate the NetBIOS names into an IP address. For example, after performing a Map Network Drive function, the client would use the NBNS to retrieve the selected server's IP address before a connecting to the server.

NetBIOS classes

When a client registers or looks up a NetBIOS name, the action performed depends on the name's class:

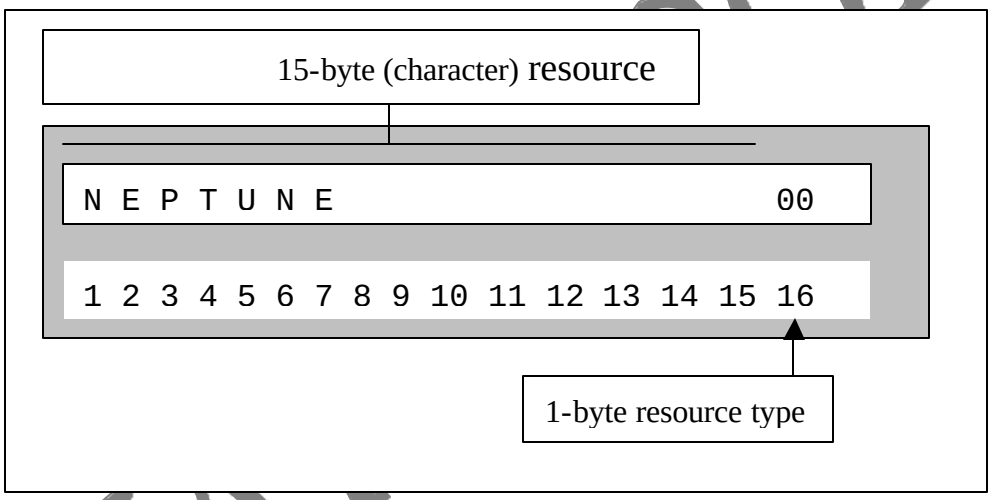
Class	Description
Unique names	Only one IP address. If a system attempts to register a name that is in use an error will occur
Group names	Many IP addresses. When registering a group name, its IP address is added to the listing maintained for that name. When translating a group name, all the IP addresses associated with that name are returned
Multihomed	Unique names where the system owning the name has multiple interfaces. All the multihomed host's IP addresses are associated with one name
Domain	Associated with NT domains

NetBIOS formatting

NetBIOS names are always stored and held in uppercase. Additionally, the name can contain any characters except:

- Characters less than a space (control characters)
- `" . / \ [ ] : | < > + = ; ,`

NetBIOS names are always at most 15 characters long. Names less than 15 characters are padded up to 15 characters. When names are transmitted, an additional character, in the 16th position, holds the NetBIOS suffix:

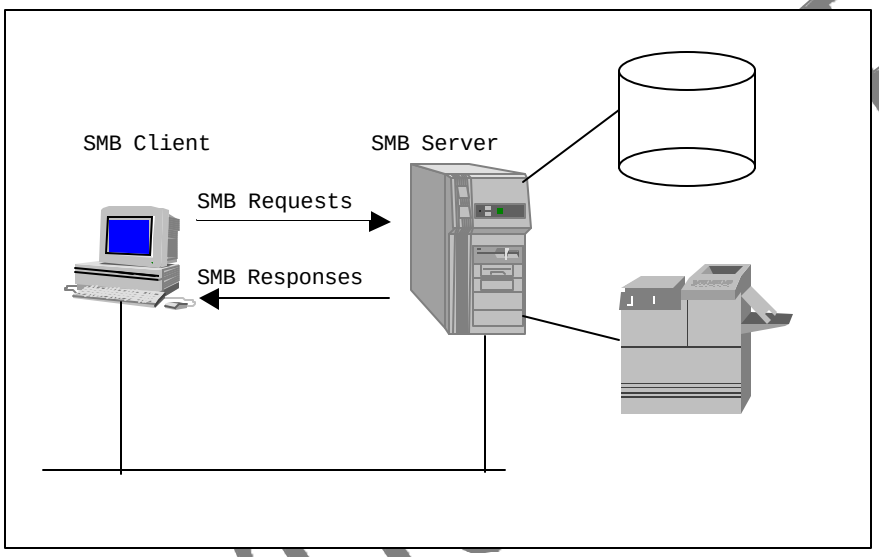


Some NetBIOS suffix values are listed in the table below:

Suffix	Class	Description
00	Unique	Workstation names
00	Group	Domain/workgroup name
03	Unique	Messenger name
20	Unique	Server name

SMB Protocol Overview

The SMB protocol is a client/server protocol that allows clients to request access to a server's resources. A client is any system that requests resources from another system. The client sends request and the server sends back responses.



Clients access the server's *shares*, which are the shared file and printer resources the server makes available. The SMB protocol handles the details of sending the share's information between the client and server.

You specify the path to a share in the form `\\server\share`, where:

- `server` is the server's NetBIOS name
- `share` is the share's name

Windows users refer to a share's full path as the "service name."

Security

Samba is very security conscious. Before Samba performs a client's request, the client must be verified as a valid user. Administrators have complete control over who may access which shares by configuring *security levels*.

Share-level security

The server does not require user authentication, but before connecting to each share, a per-share password may be required. Though this security is share based, a valid user and password are still required; sometimes, this is achieved by using a guest account.

User-level security

The client supplies user information before Samba allows access to any shares. The Windows logon username and password are normally used, which causes a problem when different logons are used for Linux and Windows systems.

Samba Components

Samba consists of the `smbd` and `nmbd` daemons. The `/etc/smb.conf` file configures both of these daemons. While only these components are necessary to actually use Samba, several other pieces make Samba a more robust server.

The table below describes Samba's components.

Component	Description
`nmbd`	The NetBIOS Name Service Daemon (`nmbd`) handles NetBIOS name lookups and WINS requests. A properly configured Samba server requires at least one `nmbd` process to be running. Additional copies are required if Samba is configured as a WINS server, or if Samba is translating NetBIOS names using DNS
`smbd`	The Server Message Block Daemon (`smbd`) handles file and print access. Proper configuration requires a minimum of one `smbd` process and an additional one for every client connected to Samba
`smb.conf`	The Samba configuration file contains all the configuration information for `smbd` and `nmbd`
`smbclient`	The SMB client program enables Linux users to access other SMB servers, such as Windows NT and Windows 9x
`nmblookup`	The NMB lookup program enables users to query servers for registered NetBIOS names
`smbstatus`	Administrators use the SMB status command to discover information about the system's Samba server, including who is currently accessing which shares
`smbprint`	A shell script for printing to Windows systems from Linux
`smbtar`	A shell script for backing up Windows systems under Linux

Both TCP and UDP are used when clients access Samba. TCP is used for network logins and accessing file and print shares by establishing a TCP connection to `smbd` on port 139. UDP is used for registering or translating NetBIOS names and browsing the network. Depending on the function being used the UDP datagrams are sent to ports 137 or 138.

Controlling Samba

Administrators control Samba with a special script. Red Hat Linux and TurboLinux administrators use the `/etc/rc.d/init.d/smb` script, while administrators of other distributions use the `/usr/sbin/samba` script.

These scripts are identical, and consequently take the same parameters, as shown in the following table:

Parameter	Description
`start`	Start the Samba daemons, `smbd` and `nmbd`, if they're not already running
`stop`	Stop the Samba daemons if they are running
`restart`	Restart the Samba daemons, by executing `stop` and then `start`
`status`	Print status information about the Samba daemons

The following section of an administrator's login shows how to control Samba:

```
# Start the Samba server
$ /etc/rc.d/init.d/smb start
Starting SMB services: smbd nmbd

# Stop the Samba server
$ /etc/rc.d/init.d/smb stop
Shutting down SMB services: smbd nmbd

# Restart the Samba server
$ /etc/rc.d/init.d/smb restart
Restarting SMB services:
 Shutting down SMB services: smbd nmbd
 Starting SMB services: smbd nmbd
done

# Query Samba's status
$ /etc/rc.d/init.d/smb status
smbd (pid 10744 2935) is running . . .
nmbd (pid 2935) is running . . .
```

Samba Logs

The Samba log files provide information on Samba's startup procedure and operational status.

All Linux distributions install Samba log files into the `/var/log/samba` directory. Within this directory are two general log files:

- `lob.nmb`, the NetBIOS Name Server Daemon (`nmbd`) log.
- `log.smb`, the Server Message Block Daemon (`smbd`) log.

After normal startup, the following entries should be in these logs:

```
# Display the NetBIOS Name Server log
$ head -1 /var/log/samba/log.nmb
[2000/09/25 01:00:45, 1] nmbd/nmbd.c:main(684)
Netbios nameserver version 2.0.5a started.
copyright Andrew Tridgell 1994-1998

# Display the Server Message Block log
$ head -1 /var/log/samba/log.smb
[2000/09/25 01:00:45, 1] smbd/smbd.c:main(628)
smbd version 2.0.5a started.
copyright Andrew Tridgell 1992-1998
```

Chapter Summary

Samba allows a Linux network administrator to integrate Windows into the Linux environment and vise versa. This creates a heterogeneous network where Windows and Linux can cooperate.

Samba allows an administrator to flexibly integrate Linux into a mostly Windows environment or vice-versa. This flexibility leverages a company's growth with both operating systems, minimizing the impact of reliance on a single operating system vendor.

Chapter 2:
Samba Configuration

Chapter Overview

Samba's configuration is very flexible and allows you to tailor it to meet your system's needs. Samba's entire configuration is maintained in the `smb.conf` file making system modification easy. Samba also incorporates the use of macros or variable substitutions that allows further customization of the system based on individual users.

Chapter Objectives

After completing this chapter, you will be able to:

- describe the `smb.conf` file.
- automate Samba using variable substitution, include files, and configuration files.

Configuring Samba

You configure both the `smbd` and `nmbd` daemons configuration in `/etc/smb.conf`. This file houses shares and configuration parameters.

Shares

The `smb.conf` file is split into sections known as *shares*. Shares begin with a descriptive string, such as `myShare`, enclosed inside of brackets: `[myShare]`. Once you've defined the share name, you'll declare configuration parameters in `name=value` format.

Standard shares include:

- `[global]`, which defines the server's general operating parameters.
- `[homes]`, which defines users' home shares properties.
- `[printers]`, which defines the shared printers' attributes.

Shares that you define for your specific needs are tied to directories within your Linux filesystem. The Samba server then makes these shares available for "sharing" to Windows clients.

Configuration parameters

The table below summarizes some of the common general configuration options.

Configuration Name	Description
`workgroup =` *string*	This sets the SMB workgroup name. The string, as others in this file, does not require any quoting
`server string` = *string*	This is a descriptive string for this server
`hosts allow =` *IP address* [*IP address…*]	This is useful for limiting the machines from which a Samba or Windows client may connect to this server. You may specify as single IP address, network numbers, or domain names
`guest account =` *username*	A guest account, which usually only has read privileges on shares. You must add the user *username* to your Linux system to provide guest access
`security =` *securitytype*	Sets the type of security used to authenticate connecting clients. See below for an explanation of the different types
`interfaces =` *interface*	If you have more than one network interface, you must list the ones Samba should listen to here
`wins support =` *yes*	Enables `nmbd`'s WINS server support
`wins server =` *IP address*	Enables `nmbd`'s WINS client support

Samba server security types

There are four security types for a Samba server:

Security	Description
`user`	Has Samba act like an NT server, where access can be restricted to certain users and groups
`share`	Has Samba act like a 95 server, where each share has a password that allows either read-only or read-write access
`server`	Has Samba pass authentication on to SMB server. Specify the server location with the configuration option: `password server =` *IP address*
`domain`	Has Samba delegate authentication to an NT domain controller

Defining Shares

Configuration blocks in `/etc/smb.conf` define share's attributes:

```
[myShare]
  comment = My personal shared directory
  path = /home/user06/sambashare
  browsable = no
  writable = yes
  printable = no
```

Configuration blocks, which include the share name and all subsequent configuration parameters, continue until the next block's definition or the end of file.

The table below summarizes some common options:

Configuration	**Description**
`comment =` *string*	Provides a comment for this share
`path =` *pathtoshare*	Tells `smbd` where to find this share's files
`browseable =` *yes*	Sets whether this share can be found by browsing available NetBIOS names
`writable =` *no*	Sets whether or not users who access this share can write files to it
`printable =` *no*	Sets whether or not users who access this share can print with it. Usually all shares but `[printers]` will not be printable
`guest ok =` *no*	Allow or deny guest access
`valid users =` *username* [*username…*]	List users who should have access to the share
`write list =` *groupname*	If `writable = yes` and this option is given, only members of this group will be able to write to this share
`create mask = ####`	Sets the file creation mask for files written to this share. This is not an *umask*, but an actual file mask

Samba Automation

Samba has a very powerful set of features that help to make the `smb.conf` file more compact and provides great flexibility in configuring it. These include `smb.conf` macros, variable substitution and include files.

Samba variable substitutions

Samba maintains many variables internally, which relate to connection, services, users, and so on. Many of these are available for substitution into parameters in the `smb.conf` file, as long as the parameters are strings. When one or more variables appear in parameter's string value, Samba substitutes the variable's value at that point in the string.

A percent sign (%) introduces variables, and all variables consist of a single character name. For example, the following parameter defines the share's path:

```
path = /home/%U/public
```

The `%U` macro substitutes the user's name into the string, so if the current user is `jennie`, then the path would be, after substitution, `/home/jennie/public`.

There are three general classes of variable substitution in Samba: basic, extended, and ad-hoc.

Basic substitutions

These substitutions are performed wherever they appear in the `smb.conf` file. They are performed regardless of whether the parameter's value is set in the `smb.conf` file or by default in the source code.

The basic substitutions performed are listed in the following table.

Variable	Description
%a	Returns the remote architecture, when known
%d	Returns the process ID of the current process
%h	Returns the first component of the Samba server's hostname
%m	Returns the NetBIOS name of the client
%v	Returns the version number of Samba
%G	Returns the group name of the user's primary group
%I	Returns the client's IP address
%L	Returns the server's NetBIOS name
%M	Returns the client's DNS name
%N	Returns the NIS home directory server for the current user
%R	Returns the remote protocol
%T	Returns the server's current time
%U	Returns the username for the current session

Extended substitutions

Samba applies extended substitutions after applying the basic substitutions. The extended substitution variables are:

Variable	Description
%G	Returns the group name of the connection or service
%U	Returns the username of the connection or service
%H	Returns the current user's home directory
%P	Returns the path for the current service
%S	Returns the current service name

You cannot apply extended substitutions everywhere within `smb.conf`. Only the parameters below allow extended substitution:

- `comment`
- `print command`
- `logon script`
- `lpq command`
- `lprm command`
- `lppause command`
- `lpresume command`
- `path`
- `preexec`
- `postexec`
- `root preexec`
- `root postexec`

Ad-hoc substitutions

Samba also does several ad-hoc substitutions that are performed as needed to support specific features in some parameters. For example, in handling the print command parameter, Samba applies the following ad-hoc substitutions:

Variable	Description
%s %f	Replaces the print filename
%p	Replaces the printer name or the print service name if no printer name is defined

The following parameters have ad-hoc substitutions applied to them:

- add user script
- comment
- delete user script
- ldap filter
- force group
- force user
- lpq command
- lprm command
- lppause command
- lpresume command
- message command
- passwd program
- passwd chat
- print command
- read list
- valid user
- invalid users
- user, users, username
- write list

Include files

Samba allows you to include other files into `smb.conf`, and because the names of these included files can contain macros, Samba's behavior can be modified at runtime.

The include file parameter is `include = filename`, where `filename` is the any system file. Since the filename can contain macros you can configure Samba's behavior depending on the client. If a file does not exist Samba ignores it, allowing you to include parameters that might not work in all situations without crashing Samba.

The `include` parameter can aid in troubleshooting. By adding the following to the `[global]` share of your `smb.conf` file, you can affect the behavior of Samba for individual clients because the `%m` in the include file parameter is replaced with the client's NetBIOS name:

```
include = /var/log/samba/conf.%m
```

Another use is to have different include files for different client architectures or users. The following allows you to include additional parameter for some client architecture:

```
include = /etc/smb.conf.%a
```

Then, by simply creating the file `/etc/smb.conf.winNT`, you can include parameters that are included only for Windows NT systems.

Config files

The third feature that assists with automation is the `config file` parameter. This parameter allows you to replace the entire `config file` with another `config file`.

Syntax

```
config file = filename
```

Since the `config file` parameter takes any basic substitutions, each client can have different `config files`. The `config file` option takes advantage of variable substitution allowing you to load a special configuration file based on the client's machine name or client's user name that is connecting.

For example, the following line instructs Samba to use a configuration file specified by the connecting client's NetBIOS name. If the file exists the options specified in the original configuration file are ignored. The following example attempts to lead a new configuration file based on the client's NetBIOS name:

```
[global]
      config file = /etc/smb.conf.%m
```

However, the `config file` parameter is not as flexible as the `include` parameter, because the `config file` parameter replaces the entire configuration file.

Chapter Summary

Configuring Samba is made easy by only having one file, the `smb.conf` file, that needs to be modified in order to incorporate changes. Through the use of macros or variable substitutions you can create a system that is customized by many variables, including users, machines.

Samba provides an enormous number of configuration options, which you can use to tailor it to meet your needs. Samba's configuration man page (`man smb.conf`), describes most of Samba's configuration options.

Chapter 3:
Samba File Shares

Chapter Overview

Sharing files is one of the main functions of Samba, and file shares can be configured at a detailed level. A Samba administrator can create guest shares that can be accessible to all users or she can limit access to specific users or workstations.

Chapter Objectives

After completing this chapter, you will be able to:

- find and access shares.
- configure file shares.
- restrict access to shares.
- use Samba to map UNIX permissions.
- use Samba to map NT permissions.
- describe the advanced `smb.conf` file parameters.

Locating and Accessing Shares

There are a couple ways a user can request share access:

- Use the `net use` command from a DOS window:

```
net use h: \\server\share
```

- Use the "Map Network Drive" dialog box from Windows. Right-click either the Network Neighborhood or My Computer to locate the box.
- Browse the network and find a server, then select the shares.

Determining a share's existence

Before connecting a client, Samba determines whether the requested share exists by completing the following process:

1. Look for a section corresponding to the requested share name in the `smb.conf` file. If found return it; otherwise go to step 2.
2. Look for a `[homes]` share in the `smb.conf` file. If found, verify the share name matches the username by using the `/etc/passwd` file. If it matches clone the `[homes]` share and return the new share, otherwise go to step 3.
3. Look for a `[printers]` share in the `smb.conf` file. If found, check the `/etc/printcap` file to determine whether the requested share is listed. If it is clone the `[printers]` share and return it, otherwise go to step 4.
4. Look for a default service. If one is found, change it to match the requested service and return it, otherwise return an invalid network name error to the client.

If Samba is unable to access the directory specified, an invalid network name error is returned to the client. There are a couple reasons why Samba might not be able to find the share because of misspellings, forgetting to create the share, or the necessary permissions are not set.

Caveats with Samba's share search order

Because of Samba's share search order, some share names might not be available. A `smb.conf` file's share name has priority over any entries in the `/etc/passwd` file or the `/etc/printcap` file, even if `[homes]` and `[printers]` sections are defined. This means that john's home share will not be found if the `smb.conf` file contains a share named `john`.

Home shares have priority over printer shares. If a printer called `john` is defined in the `/etc/printcap` file, it will not be found if a user named `john` is in the `/etc/passwd` file and the `[homes]` share has been defined, or if the `smb.conf` file contains a share named `john`.

User verification

After Samba knows the share exists, it determines if the user has share access. The following process determines whom the user accessing the share is recorded as:

1. If the client has submitted a username/password pair previously that validates, the validated user is recorded as the user seeking access.
2. If the client has already submitted a valid username and now supplies a correct password, the validated user is recorded as the user seeking access.
3. The client's NetBIOS name and any previously used usernames are validated using the operating system's standard mechanisms with the supplied password. If any validate successfully, the validated user is recorded as the user seeking access.
4. If the client has previously validated a username/password pair with the server and the client passed the validation token in the share access request, the previously the validated user is recorded as the user seeking access.
5. If a user = list has been specified on the share, the client has supplied a password, and the combination of username specified in the share and the password validates, the validated user is recorded as the user seeking access.

After Samba determines the local user accessing the share, it checks the various parameters to determine whether that user can access the requested share. If that user is not allowed share access, Samba returns an access denied error.

However, even if Samba gives a user share access, two more sets of checks are completed before he or she can access the share's files:

1. To write to files in the share, the share must be `writable`.
2. Normal Linux file permissions must be satisfied.

Configuring File Shares

To create a file share, place a share name in the `smb.conf` file. Enclose the share name in square brackets and then add all applicable parameters.

All `smb.conf` file parameters are share parameters, unless specified as global in the `[global]` share. Placing share parameters in the global section causes them to be the defaults for all shares.

Basic share properties

Within the `smb.conf` file, several parameters defining the share's basic properties may be specified. These include a share's write status, a share's visibility status to other workstations, and the share's text description that appears in browser lists.

The following table describes the basic share parameters.

Name	Default	Description
`read only`	`true`	Specifies the share cannot be written to. Users specified in the write list parameter are given write access
`writable`	`false`	If set to true and the user has the correct permissions then the files on the share can be changed
`comment`		Describes the share's purpose and appears next to the share name
`volume`		Allows the volume label of the share to be changed. The default is the share's name
`browsable`	`true`	Appears in the visible share's list when browsing the Samba server
`available`	`true`	Allows connection to the share and access to its resources
`path`	`/tmp`	Specifies the Samba server's path to the file share
`time offset`	`0`	Number of seconds added to each file's access timestamp on the share

Example

```
[tmp]
      comment = temporary space
      path = /tmp
      browsable = true
      writable = true
```

File selection

When creating a file share, you can allow the visibility of only certain files or directories. Using the DOS hidden attribute, you can mark files as inaccessible.

Controlling the use of symbolic links in file shares can enhance security.

The following table lists the parameters that are available:

Name	Default	Description
`hide files`		Specify a hidden file listing
`hide dot files`	`true`	Specifies whether dot files are hidden
`veto files`		Contains a listing of files and directories that are invisible and inaccessible
`delete veto files`	`false`	Disallows the deletion of a directory that contains a veto file
`dont descend`		Contains a comma-separated list of directories Samba can't enter
`follow symlinks`	`true`	Follows the file share's symbolic links only when the links are within the share's directory
`wide links`	`true`	Follows the links pointing outside the file share being accessed

Example

Continue the temporary share defined above, adding some security:

```
[tmp]
      comment = temporary space
      path = /tmp
      browsable = true
      writable = true
      follow symlinks = false
```

Guest shares

If you have a share you want to allow access without managing usernames and accounts, you can set up a *guest share*, also called an *anonymous share*. A guest share provides access without requiring a username and password. Access to the share will be allowed by anyone who can make a network connection to your Samba server.

The following table shows the parameters available in setting up guest access:

Name	Default	Description
`guest ok`	`false`	Specifies whether guest access to the share is allowed
`guest account`	`nobody`	Linux username if the guest ok parameter is set to true
`guest only`	`false`	If set to true only guest connection are allowed

Example

Again, a continuation from the previous example, making the share completely anonymous:

```
[tmp]
      comment = temporary space
      path = /temporary
      browsable = true
      writable = true
      follow symlinks = false
      guest ok = true
      guest only = true
```

TIP:

Some older Windows clients do not send more than 8 characters for a share name. If you intend to use Samba in an environment with older clients (such as Windows 3.1), you should restrict share names to no more than 8 characters.

Restricting Access to Shares

Creating a completely anonymous share is fine as long as limitations are enabled on who can access the share. One way you can control access to a share is by restricting connecting hosts.

Restricting access by host

Samba uses a syntax similar to TCP wrapper's to set share access restrictions.

The following table lists the parameters available to restrict hosts:

Name	Default	Description
`hosts allow`		Listing of hosts that are permitted share access
`hosts deny`		Listing of hosts that are denied share access
`use rhosts`	`false`	Specifies whether hosts can access without passwords using the rhosts mechanism
`hosts equiv`		Set the hosts that should be considered equivalent (in the security sense) to the Samba server

There are several rules for specifying hosts:

- Separate multiple hosts with commas or whitespace.
- Specify a host by either name or IP address.
- Specify IP address ranges in either `network/netmask` format or by partial IP address.
- If your system supports NIS net-groups, then specify a group of hosts using `@`.
- The `ALL` keyword matches all IP addresses.
- The `EXCEPT` keyword excludes addresses from a range.

Configuring security policies

Using the `hosts allow` and `host deny` parameters allows two, mutually exclusive, types of access policies to file shares:

1. A "mostly open" policy consists of setting `hosts allow` to `ALL` and explicitly denying workstations by adding them to the `hosts deny` list.
2. A "mostly closed" policy consists of denying access to all workstations by setting `hosts deny` to `ALL` and explicitly enabling access by adding workstations to the `hosts allow` list.

Your organization's security requirements will determine the type of policy used, if any. A mostly closed policy is appropriate for a business network where confidential information is stored, whereas a mostly open policy works better within an educational environment.

The following code, which further restricts the temporary area, limits the access to only a couple workstations:

```
[tmp]
      comment = temporary space
      path = /tmp
      browsable = true
      writable = true
      follow symlinks = false
      guest ok = true
      guest only = true
      hosts deny = ALL
      hosts allow = work01, work02
```

Restricting Access by User

If users do not use the same machine all the time, it might be easier to restrict share access by specifying usernames instead of workstation names.

Samba allows you to secure a share by user and group. These parameters can be combined with the host restriction parameters to provide flexible share access control.

The table below lists the user-level access restriction parameters:

Name	Default	Description
`read list`		User listing of those who have read-only access
`write list`		User listing of those allowed to write to the share's files
`valid users`		User listing of those allowed share access
`invalid users`		User listing of those denied share access

The `valid users` and `invalid users` parameters supercede the `read list` and `write list`. If a user is not allowed share access, then he or she will not be able to read from or write to files, even if he or she is in the read or write lists.

Linux Filesystem Permissions

Since Samba's shared files are stored on a Linux filesystem, you must manage file and directory access permission because they influence how the files are ultimately accessed.

The table below lists the parameters related to file creation and access:

Name	Default	Description
`create mask`	`0744`	Set the mode to logically OR with a file's creation permissions. This is similar to a file's umask
`directory mask`	`0755`	Set the mode to logically OR with a directory's creation permissions. This is similar to a file's umask
`force create mode`	`000`	Explicitly set created file's permissions to the given value. This value is logically OR'd with the create mask
`force directory mode`	`000`	Explicitly set created directory's permissions to the given value. This value is logically OR'd with the directory mask
`force user`		Forces all file operations performed on the share to be done as the specified Linux user
`force group`		Forces all file operations on the share to be performed as the specified Linux group

Using Windows to Modify Permissions

Windows 2000 and NT 4.0 users can change file and directory permissions on a Samba server through the Properties dialog box. Because the archive and hidden attribute do not exist, *per se*, in Linux, selecting either of these attributes has no effect.

Username Mapping

Sometimes, you may need to map a Windows username, such as "Administrator" to a Linux user, such as "root". You could do this by creating a Linux user named as your Windows user, but sometimes that is not feasible.

Samba allows you to map Windows' usernames to Linux usernames with a *username map*. Create the map, usually `/etc/username.map`, and set the `username map` parameter to the file, as in:

```
username map = /etc/username.map
```

The username information format is a Linux username followed by an equal sign and then a list of Windows or Linux usernames. Only one entry is permitted per line. There are some additional points about username mapping including:

- Text after a # or a `;` are ignored
- You can match *any* username with a `*` after the equal sign
- Specify Linux or NIS groups using the `@` after the equal sign
- Surround Windows usernames containing spaces with double quotes

With username mapping, you can group multiple Windows users into a single name. This allows you to preserve your Linux system's native grouping (in `/etc/group`), but still have the ability to conglomerate users.

Example

The users `john`, `mike`, and `rick` map to the username `proj`:

```
proj = john mike rick
```

TIP:

If you specify a mapping that contains all users, such as with `guest = *`, then you need to precede all other group declarations that follow with an exclamation point (`!`).

User Home Directories

When a user connecting to a file share has a Linux account on the Samba server, the user's home directory can be made accessible automatically with the `[homes]` share.

The `[homes]` share can use all the `smb.conf` file parameters that are applicable to normal file shares.

The following code listing is a sample configuration for user home directories:

```
[homes]
     browsable = false
     guest ok = false
     read only = false
     create mask = 0744
     directory mask = 0750
```

No path is needed, because Samba retrieves this information from the `/etc/passwd` file. This is a feature available only for the homes share; for all other shares you must explicitly declare the path.

Advanced smb.conf Parameters

There are many more parameters available to configure Samba. A few of them will be discussed here. A complete listing is available in Samba's man page: `man smb.conf`.

Security

There are several parameters that can enhance the security of a Samba share. These include:

- determining who may have root access when connected to a file share.
- denying any anonymous logons.
- rejecting any accounts that have null passwords.

Opportunistic locks

The SMB protocol used by Samba and clients connecting to it contains a method of file locking called *opportunistic locks* or *oplocks.* This is a client-side protocol that allows SMB clients to cache file data instead of re-reading it from the server, resulting in significant performance improvement.

Unfortunately, oplocks only work correctly when all users access the share through the oplock mechanism provided by Samba. Users accessing files on a Samba share must do so through the SMB protocol or caching problems and data corruption can occur.

A Samba user may change the file's content, but when the file is accessed locally, the changes do not appear. This problem can occur if a file share refers to a directory that is accessibly through NFS as well as from a Samba share. This can be especially true of Linux user home directories.

TIP:
By default oplocks are not enabled. If you do not have any outside access to the share's files (other than through Samba) or the share is completely read-only, enable oplocks with `oplocks = true` to reap a significant performance gain.

Magic scripts

Samba allows shell scripts and other commands to execute when certain events occur. This functionality allows for handling unusual situations such as mounting CDROM devices or performing post-processing of files copied to Samba file shares.

Internationalization

Samba supports the use of non-English characters in filenames through its internationalization features. Several of the international character sets defined by the ISO Standard 8859 are supported. This covers most of Western and Eastern European languages as well as Russian Cyrillic and Japanese.

Chapter Summary

Maintaining and configuring Samba file shares is accomplished through the `smb.conf` file. This file lets you set which files can be accessed, who can access the files, how permissions are determined and if home directories should be mounted.

You learned about the importance of search order and how some shares might be unexpectedly available. You may restrict access to shares by host or by user. Also, remember that Windows and Linux do not have identical access permissions (such as the archive attribute).

This page intentionally left blank.

Chapter 4:
Samba Printer Shares

Chapter Overview

Configuring and managing printers for a Samba server is necessary for a successful system. All of your users need to be able to print jobs to local printers and also may require the access to special purpose printers. Samba provides quick and easy methods of configuration, management, and troubleshooting to aid you in this task.

Chapter Objectives

After completing this chapter, you will be able to:

- configure printers under Samba.
- manage print jobs.
- troubleshoot printing problems.

Configuring Printers under Samba

By default, Samba uses the Linux system's `/etc/printcap` file to know what printers are available. Any printer you define in `/etc/printcap` is immediately available to Samba through the `[printers]` share in `smb.conf`.

Globally configuring printers

All printers available to Samba follow the parameters set in the `[printers]` share. This allows administrators to set global parameters across all print shares.

Exa***m***ple

```
[printers]
     comment = Linux printer
     path = /var/spool/samba/
     printable = true
```

The `[printers]` share must, obviously, have the `printable` parameter set to `true`; otherwise, Samba will not offer any printer services.

Macros substitutions are allowed, and several related to printing are listed below:

- `%s` and `%f` are replaced with the spool file name.
- `%p` is replaced with the primary printer name.
- `%j` is replaced with the Linux job number.

Customizing individual printers

You can also set parameters that apply only to a specific printer. This may be useful for printers that need to be restricted to a certain set of users, such as a color printer or a printer in a restricted location.

To specify additional parameters for a printer, create a section in the `smb.conf` file using the primary printer name as the share's name, then add any additional parameters.

Example

If your `/etc/printcap` had entries such as:

```
lp|standard|bw-laser:\
 :sd=/var/spool/lpd/standard:\
 :mx#0:\
 :sh:\
 :rm=peepy:\
 :rp=raw:\
 :if=/var/spool/lpd/standard/filter:

color|color-laser:\
 :sd=/var/spool/lpd/color:\
 :mx#0:\
 :sh:\
 :rm=peepy:\
 :rp=raw:\
 :if=/var/spool/lpd/color/filter:
```

Then you could configure these printers individually as:

```
[lp]
      comment = Basic black and white printer
      guest ok = true

[color]
      comment = Restricted color laser
      guest ok = false
      valid users = manager leads
```

Managing Print Jobs

Though print job management is the responsibility of the administrator outside of Samba, Samba administrators should be aware that Samba can also manage print jobs. Print job management includes tasks such as pausing, resuming, and removing jobs from the print queue. The table below shows the `smb.conf` file parameters used to implement this functionality:

Name	Default	Description
`printing`		Sets the system's printing subsystem type. Linux will use either `BSD` or `LPRNG`
`print command`	`lpr -r –P %p %s`	Command to print a file once the Windows client has spooled the entire file to the Samba server
`lpresume command`	`lpr -i %p-%j -H resume`	Continues the printing of a previously stopped or suspended job
`lprm command`		The command used to remove a completed print job. Sometimes unnecessary when the print command removes the job

Samba is not responsible for removing printer spool files once the job completes; that is the responsibility of the print command.

Chapter Summary

Configuring and managing printers under Samba can be accomplished at either a global or individual level. This allows you to set the majority of your printers up quickly and also allows you to override the global parameters for special purpose printers. Samba may also be used to manage print jobs.

This page intentionally left blank.

Chapter 5:
Password Management

Chapter Overview

Because Samba is used in environments with many different password management facilities, it can manage passwords in many different ways. You will find this flexibility very useful because Samba can grow with you as your use of Samba grows.

Chapter Objectives

After completing this chapter, you will be able to:

- describe how the SMB protocol provides password authentication.
- configure Samba to authenticate against the `/etc/passwd` file.
- configure Samba to use encrypted passwords.
- configure Samba to allow password changes from Windows clients.
- configure Samba to authenticate against another SMB server.
- configure password synchronization between Samba and Linux.
- configure your system to use the `/etc/smbpasswd` for all authentication.

The SMB Protocol and Passwords

When a client accesses your server, the SMB protocol transfers the client's authentication in a SMB request. This request carries the client's username and password in plain text, which is very insecure since anyone with a sniffer can capture your passwords.

The SMB protocol also supports encrypted passwords in later versions, such as found in Windows 98, NT 4.0, and 2000. Encrypted passwords use a challenge-response procedure, which Samba emulates faithfully.

In the final count, Windows supports four types of authentication:

Method	Description
Plain-text	The client sends a plain-text password
LMHash	The client computes the LAN Manager hash of the user's password and encrypts all server challenges with that hash
NTLM, NTLMv1	The client computes the NT hash of the user's password and encrypts all server challenges with that hash
NTLMv2	The client computes the HMAC_MD4 hash of the user's username and domain name (encrypted with the NT hash of the user's password) and encrypts all server challenges with that hash

TIP:

Samba fully supports changing passwords from Windows clients. You can change passwords from Windows 95/98, NT, and with the newest versions of Samba, 2000. You can change your password when the Samba server is receiving plain text and encrypted passwords.

Standard Authentication

By default, Samba supports plain-text passwords. In all but the smallest networks, this is a definite security hazard.

With this method, Samba receives the user's password from the SMB protocol and encrypts it using the system's encryption scheme. The encrypted value is then compared against the value stored in `/etc/passwd` for authentication.

This mode is useful because all passwords are kept in the single `/etc/passwd` file. Unfortunately, this method also requires all users to have an account on the system, so username maps can't be used with any real benefit.

Samba and Encrypted Passwords

Configuring Samba to use encrypted passwords is easily accomplished:

1. Add the following to the `smb.conf` file's global share:

```
encrypt passwords = yes
smb passwd file = /etc/smbpasswd
```

2. Create the `/etc/smbpasswd` file with the `mksmbpasswd.sh` script. This file translates the `/etc/passwd` file into the correct format for Samba to use:

```
$ mksmbpasswd.sh < /etc/passwd > /etc/smbpasswd
```

When you translate the passwords into Samba's format, you must be aware of several important factors:

- The encryption used in the Samba password file is different than used in the system password file. Therefore, it's impossible to simply copy the encrypted password over, so you'll need to have Samba users create and update their Samba passwords.
- The script moves entries from `/etc/passwd` over directly, including special and non-user accounts like `root`, `bin`, and `daemon`. You should remove these entries.

TIP:
It might not be appropriate to migrate all your users from the `/etc/passwd` file and reset their passwords in one day. This method is probably only appropriate for small systems or newly created systems.

Extracting hashed passwords from NT

It is possible to extract NT and LM password hashes kept in the Windows NT registry with a tool called `pwdump2`.

After you have downloaded and extracted the file, you will need to run the command on an existing Windows NT server:

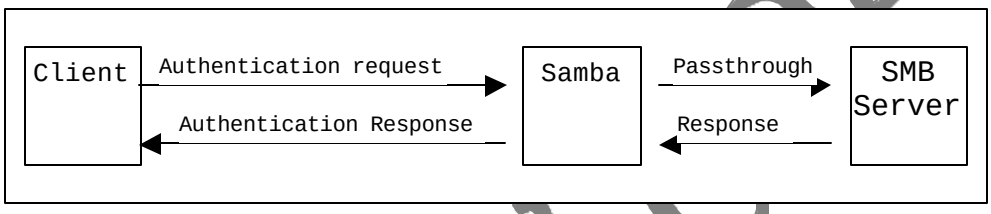
```
[C:\] pwdump2 pid > smbpasswd
```

You must replace "pid" with the process ID of the `lsass.exe` program running on the NT password server. Then transfer the created `smbpasswd` file to your Samba server and your users will not lose their passwords.

Authentication Against Another SMB Server

Samba can be integrated into NT-based networks in two different ways. It can act as an SMB server doing pass-through authentication or it can be a domain member, with a primary or backup domain controller performing all authentication.

The following diagram shows a Samba server operating in either of these modes:



The following steps that must be followed for both methods:

1. Your Samba server must be configured for encrypted passwords. You will need to add the following parameter to the global share of your `smb.conf` file:

```
encrypt passwords = yes
```

2. If you are using `security = domain`, you must also ensure that your Samba server is in the same domain or workgroup as the server you are authenticating against. You will need to include the appropriate `workgroup` parameter in the global share of your `smb.conf` file.

```
workgroup = the-workgroup
```

3. The `password server` parameter must be set up, which specifies the NetBIOS names of one or more password servers your Samba server will authenticate. For example:

```
password server = server1 server2
```

Pass-through authentication configuration

To pass authentication on to another server, you must use the `security = server` parameter in the global share.

Example

```
[global]
      workgroup = myworkgroup
      security = server
      encrypt passwords = yes
      password server = server1 server2
```

Domain controller configuration

In order for Samba to authenticate against domain controllers, your Samba server must be a domain member in the domain in which the controllers exist. This is a two-step process on the Samba server:

1. Use domain security, as below:

```
[global]
      workgroup = mydomain
      security = domain
      encrypt passwords = yes
      password server = my-pdc, my-bdc
```

2. Join the domain in which the domain controllers exist with the `smbpasswd` command:

```
$ smbpasswd –j mydomain
```

In addition, you must have added your Samba server as a domain member, by using either *User Manager for Domains* if your domain controllers are Windows NT servers or the following if your domain controller is a Samba server:

```
# note: Windows requires the ending $
$ useradd machine$
$ smbpasswd –a –m machine
```

Password Synchronization

One of the biggest problems with using the more secure encrypted password approach is that your Samba server keeps password hashes in two places: the `/etc/passwd` file and the `/etc/smbpasswd` file.

Eventually users will change their passwords from Windows or Linux without realizing that their password in the other environment remains unchanged.

Using Samba to update Linux passwords

Samba allows you to synchronize passwords between the SMB and Linux environments on your server.

This is managed by the following `smb.conf` parameters:

Parameter	Description
`unix password sync`	Specifies whether encrypted passwords are synchronized with Linux passwords when users change their passwords
`passwd program`	Specifies the commands or scripts to be used by Samba to change a user's password under Linux
`passwd chat`	Specifies a series of expect/send sequences that help manage the password changing process
`passwd chat debug`	Specifies that Samba should log all strings sent to and received from the `passwd` program at a debug level of 100. This should not be used because it means that the log file will contain the user's passwords in plain text

Authenticating Linux services with Samba

You can also have Linux services, such as login, authenticate with the Samba password file with PAM modules. The `pam_smblib.o` module, which should replace the `pam_pwdb.o` module in PAM configuration files, allows users to authenticate against and change their SMB and Linux passwords in unison.

Parameters Affecting Authentication

Because security should be a primary concern for all administrators, Samba or otherwise, it's important to know what parameters affect security. This list below summarizes those parameters that affect your Samba server's security:

- add user script
- allow trusted domains
- delete user script
- encrypt passwords
- map to guest
- min password length
- null passwords
- passwd chat debug
- passwd chat
- passwd program
- password level
- password server
- restrict anonymous
- security
- smb passwd file
- unix password sync
- update encrypted
- use rhosts
- username level
- username map

Chapter Summary

Maintaining separate passwords for each machine or system you need to access can be a cumbersome task. As a Samba administrator you wish to keep your users happy. One way to do this is by providing them with the means to have only one username and password for the system.

This chapter introduced you to several methods of creating user accounts that can be used on both Windows and Linux. The method you choose will depend entirely on the size and system needs.

This page intentionally left blank.

Chapter 6:
Serving Logons and Profiles

Chapter Overview

Allowing your users mobility makes their lives easier, because their configuration follows them regardless of where from they access the system. Samba provides administrators with the ability to configure per-user logon and use profiles that activate when a user authenticates him or herself to Samba. This flexibility will make serving a large number of users easier, because all user configuration remains in a single location.

Chapter Objectives

After completing this chapter, you will be able to:

- describe the purpose of a logon server.
- configure Samba as a logon server.
- configure Samba to support profiles.

Logon Servers

Using a logon server to centralize user authentication eliminates the need to maintain user accounts and passwords on every network workstation. The logon server can provide logon script processing and roaming profile storage, allowing users their own customized drive mappings and Windows desktop settings, regardless of the machine they are on. Using these features can improve your user's network experience.

User authentication

The logon server's primary purpose is authenticating network users, which consists of verifying the user exists in the network domain and that the correct password has been entered.

After a successful logon to a domain logon server is complete, the client's authentication information is used for all other network resource accesses. The network logon server enables a single network logon regardless of whether network resources are being accessed on the logon server or on other services within the domain.

On most Samba and Windows based networks, the primary domain controller (PDC) and the logon server are the same system. However, Samba has the capability to service Windows 95/98 domain logons without necessarily being configured as a true domain controller. Windows NT domain logons require a primary domain controller to be present on the logon server.

Logon scripts

Logon scripts are executed immediately after a successful logon. For Windows clients, the scripts are DOS-style batch files that are used to perform a variety of tasks, including synchronizing the client PC's time to the server and setting up network drive and printer connections.

Roaming profiles

Windows systems support a roaming profile server, which stores specific user configuration settings for a desktop PC. These settings are then available to users when they log into other network desktop PCs.

Configuring Samba as a Logon Server

When configured as a logon server, Samba also supports network logon scripts and roaming profiles.

Network Logon Support

To process network domain logons, your Samba server must be configured to process domain logons. Part of this requirement is that the logon server (Samba) must be able to resolve client names. Doing so requires that a WINS lookup mechanism exists.

Configuring a WINS server

A WINS server resolves the NetBIOS network names into IP addresses. The WINS server does not have to be the same as your Samba server, but it often is.

`Samba logon server as a WINS server`

If you want the Samba logon server to also be the WINS server, add the following option to the server's `[global]` share:

```
wins support = yes
```

`Samba Logon Server as a WINS Client`

If your Samba server will be utilizing another system as the WINS server, add the following entry to the `[global]` share, where `x.x.x.x` represents the network WINS server's IP address:

```
wins server = x.x.x.x
```

Enabling Logon Support

For Samba to support network domain logons add the following parameter to the `[global]` share:

```
logon server = yes
```

Logon scripts

In order to provide logon scripts for network clients, you must first enable the proper support in the `smb.conf` file. Then you must provide the logon scripts by using a special share on the Samba logon server.

Configuring logon script support

For network clients to execute the network logon script, the Samba server must provide a special network share. It must be named `[netlogon]` and should be readable by all users.

Example

```
[global]
      logon script = /shared/logon.bat

[netlogon]
      comment = Netlogon share
      path = /shared/netlogon
      writable = no
      guest ok = no
      browseable = no
```

This establishes the network logon share and will execute the `logon.bat` file during a client's login. It's important to remember that this file must be saved in DOS, not Linux, format.

TIP:

You can create customized logon scripts with the variable macros discussed earlier. For example, a logon script customized for each user could be set up with `logon script = %U.logon.bat`.

How Samba Supports Profiles[1]

Windows 95, Windows 98, and Windows NT all support the roaming profile server for storing user-specific configuration information, though differences exist in how each handles roaming profiles. With these differences come a penalty; Samba's roaming profiles configuration is operating system dependent.

Configuring Samba to support profiles

All Windows clients ask the logon server for their profiles' location, but use different mechanisms to retrieve the information.

Windows 95/98 clients obtain the information by sending a "NetUserGetInfo" request to their logon server. The "NetUserGetInfo" response only contains the user's home directory location, allowing very little control over where clients get their profiles.

Windows NT clients retrieve the appropriate information by making a "NetSamLogon" RPC. The response contains a separate field for the profile location. The logon home parameter can control where Windows NT client profiles are kept.

Windows 95/98 roaming profiles

For Windows 95/98, a roaming profile enables each network user to store the desktop contents and Start menu options in the logon path, which for these operating systems is always the user's home directory.

Several subdirectories representing the desktop as well as a `USER.DAT` file containing configuration settings are stored there.

Window NT Roaming Profiles

Windows NT is more flexible because it allows you to specify where the roaming profiles reside on the Samba server. This allows you to isolate your user profiles into a central location and not clutter up home directories.

[1] The latest version of Samba, 2.0.7, doesn't support Windows 2000 logon scripts. The next major release, 3.0, will.

Details of the Required smb.conf Parameters

The following table lists all of the `smb.conf` parameters that may be required in order to process user logons:

Parameter	Description
`logon drive`	Sets the drive letter to which Windows NT clients will map the user's home directory during the Windows NT logon process
`logon home`	Sets the home directory for the logon process
`logon path`	Enables roaming profile support and configures the path used to store user profiles
`logon script`	Configures the logon script's path and filename for domain logon script processing
`logon server`	Configures Samba to support domain logon request from clients
`root postexec`	Configures a command that runs on the Samba server when a client disconnects from a share
`root preexec`	Configures a command that runs on the Samba server when a client first connects to a share
`wins server`	Configures Samba to utilize another WINS server for resolving NetBIOS names into IP addresses
`wins support`	Enables WINS server support in Samba
`workgroup`	Sets the workgroup and network domain in which the Samba server will participate

Chapter Summary

The addition of a logon server to your Samba system provides your users the ability to maintain customized logon scripts, either on a system level or an individual level. The logon server allows for roaming profiles so user specific configurations are maintained on any workstation the user logs into. These features make logon servers a valuable addition to your heterogeneous network.

This page intentionally left blank.

Chapter 7: Accessing SMB Shares from Linux

Chapter Overview

Samba normally allows Windows users the ability to access Linux from a Windows client, but the reverse works as well. Samba provides services that allow easy access to SMB shares, whether they reside on Windows or Samba servers.

Chapter Objectives

After completing this chapter, you will be able to:

- use the `smbclient` program to access SMB shares.
- use a Linux client to access SMB printer shares.
- back up a SMB share onto a Linux system.
- mount a SMB share onto a Linux system.

Accessing SMB Shares

The `smbclient` program is a command-line Linux client that operates very much like the command-line `ftp` client.

Basic use

In the simplest form, the `smbclient` command accesses SMB servers by share name in several equivalent forms:

```
# access the share using Windows syntax
$ smbclient '\\hostname\sharename'

# access the share using Linux syntax
$ smbclient //hostname/sharename

# access the share by escaping the slashes
$ smbclient \\\\hostname\\sharename
```

If the share required a password, `smbclient` will prompt the user. If the user has access to the share, he or she will be allowed in and given the `smb: \>` prompt:

```
$ smbclient '\share\tmp'
Added interface ip=10.1.1.1 bcast=10.0.0.255
namsk=255.255.255.0
Got a positive name query from 10.1.1.2 (
10.1.1.2 )
Password:
Domain=[NTDOM] OS=[Unix] Server=[Samba 2.0.6]
smb: \>
```

The `smbclient` program provides commands including ones that allow you to move around a filesystem, retrieve files, and place files. Some of the more common commands are described in the table below:

Command	Description
`dir [mask]`	Prints a list of the files in the current directory. If a mask argument is specified, then only the files or directories matching the mask are displayed. The * and ? wildcard characters match one or more characters or just one character, respectively
`cd [directory]`	Changes the SMB server's current directory to the specified directory. If no directory is specified, then the current directory is displayed
`lcd [directory]`	Changes the local machine's current directory to the specified directory. If no directory is specified, then the current directory is displayed
`get rfile [lfile]`	Fetches the remote file `rfile` from the SMB server and saves it in the local machine's current directory. Specifying a second filename will rename the file on the local machine
`put lfile [rfile]`	Puts a copy of the local file `lfile` onto the SMB server's current directory. Specifying a second filename will rename the file on the SMB server
`del rfile`	Deletes the remote file `rfile` from the SMB server
`!command`	Executes `command` on the local system

Commonly used options

Option	Description
`-h`	Print a usage message.
`-c "string"`	Execute the semicolon-separated commands in "`string`". This is useful for scripting `smbclient` and not using it interactively
`-d level`	Set the debug level, which you might find useful if you're having problems connecting
`-E`	Write messages to standard error instead of standard out
`-I IP_address`	Specify the IP address to connect to. This is needed when the NetBIOS name doesn't correctly map to an IP address, or if you don't have access to the NetBIOS name (for example when the server is on a different network)
`-L hosts`	List the available shares
`-M hosts`	Send a popup message to the host. If permissions allow, the message will appear in a dialog box on the server's console
`-P password`	Password required to access the server's share. If no password is supplied and the share is password-protected, then the user is prompted to enter one
`-U username`	Specify the remote system's user log in
`-W workgroup`	Specify the workgroup name

Printing to SMB Printer Shares

To print to a remote SMB printer share, you will need to set up an `/etc/printcap` definition for the printer. The definition instructs Linux's printing subsystem to forward output to that printer through the `smbprint` program.

Printer capability definition

You will need to add, at a minimum, a printer definition like below to your `/etc/printcap` file:

```
smbprinter:\
      :lp=/dev/null:\
      :sd=/var/spool/lpd/smbprinter:\
      :if=/usr/local/bin/smbprint:
```

Because the printer is actually remote, the `lp` directive shouldn't point to a valid printer; usually `/dev/null` is sufficient. If you want a duplicate copy sent to a local printer, then you may specify a printer to the `lp` directive.

The input filter (`if`) directive receives all input destined for the printer and forwards it to the appropriate SMB printer.

Configuring document destination

In the SMB printer's spool directory (`sd`), create a `.config` file that holds connection information. The `smbprint` client will use the configuration in this file to connect to the remote printer and print:

```
$ cat /var/spool/lpd/smbprinter/.config
server='server'
service='printer'
password='foo'
```

Backing Up SMB Shares

It's possible to use the `smbclient` to retrieve all files from a SMB share, but Samba provides a more convenient command that will be familiar to all Linux system administrators: `smbtar`. This shell script is similar to the `tar` command, and it allows you to store all files from a SMB share onto a tape or a single, local file.

To use this command, you will need to:

1. share the directory that you want to back up on the Windows system, if not already shared.
2. run `smbtar`:

```
smbtar –s server –x share –t device/file
```

Command Line Options

Option	Description
`-r`	Restore to the SMB share rather than saving from it
`-a`	Reset the archive bit on backed up files
`-v`	Specifies verbose mode
`-s server`	Server to back up
`-p password`	Password for server
`-x share`	Specifies the share to backup
`-N file`	Back up all files that are newer the than given file's timestamp
`-b blocksize`	Specifies the block size
`-d dir`	Specifies the directory to back up from the share
`-l log-level`	Specifies the Samba log level
`-u user`	Specifies the user’s logon
`-t device`	Specifies the device or file to back up to

Working with SMB Shares from the Filesystem

Linux is the only non-Windows operating system that allows you to mount SMB shares. This makes the SMB share appear as a part of the local filesystem, where you can use normal Linux commands (such as `cp` and `mv`) to modify the files.

The `smbmount` command allows you directly mount a SMB share onto the Linux filesystem. For most Linux distributions, the `smbmount` program is called automatically by `mount` when the filesystem's type is `smbfs`.

Mounting SMB shares

Syntax

```
smbmount //server/share mp -o opt1,opt2,...
```

Command Line Options

Option	Default	Description
`username=user`	The contents of the `USER` environment variable	Specify the user to connect as
`password=pwd`		Specify the password
`uid=name`	User ID of caller	Specify that all files mounted will be owned by this user ID (username or numeric value)
`gid=name`	Group ID of caller	Specify that all files mounted will be owned by this group ID (name or numeric value)
`debug=level`		Set the debug level
`ip=address`		Specify the IP address of the remote host
`workgroup=name`	The value of the `domain` parameter in `smb.conf`	Specify the workgroup or domain name
`ro`		Mount the file share read-only

Automatically mounting SMB shares

If you want to have a SMB share mounted on your Linux machine regularly, you can add an entry into the system's mounting table, `/etc/fstab`. You will need to specify:

- The server and share name.
- The local mount point.
- `smbfs` as the file system type.
- A username and, most likely, a password.

Example

Mount the temporary space share onto your Linux machine at boot, accessing the SMB share as the user `john`:

```
//server/tmp /mnt/tmp defaults,username=john
```

Unmounting SMB shares

You can use the `smbumount` command to unmount a SMB share. The `umount` command automatically calls the `smbumount` program when the filesystem is a `smbfs` type.

Example

```
smbumount /mnt/tmp
```

Chapter Summary

Accessing SMB shares from a Linux client gives you the ability to easily retrieve files from a Windows or another Linux SMB server. It also gives you the flexibility to use alternative printers. You can also create a backup of a Windows server easily with `smbtar`.

Finally, in a feature exclusive to Linux, you can mount a SMB share, such as on a Windows NT server, onto a local filesystem. Even if you don't use Samba to provide Windows users access to your Linux files, you can provide your Linux users access to SMB shares.

This page intentionally left blank.

Chapter 8:
Samba Performance

Chapter Overview

Correct configuration and good maintenance are necessary for a Samba server that will meet your users' needs. However, there are other factors that need to be considered outside of Samba that can drastically affect the performance of your Samba system.

Chapter Objectives

After completing this chapter, you will be able to:

- implement Samba options that will improve performance.
- fine tune your operating system to increase performance.
- choose a network topology that improves server performance.
- choose the amount of memory that will optimize your system.
- choose an I/O subsystem that will optimize your system.

Tuning Samba

One of the great strengths of Samba is also one of its great weaknesses. Samba has an enormous number of configurable options, both at compile-time and runtime. This provides one of the best file and print server systems available. However, knowing which option to change to improve performance can be a difficult task.

The following describes some of the options that can be tuned to improve your performance. Many of these can be used in either the `[global]` share of your `smb.conf` file or specified per share.

TCP socket options

The SMB protocol uses TCP, and as such, any improvements made to the way your server handles TCP protocol connections will improve Samba's performance.

You can set specific TCP socket options using Samba's `socket options` option, which is put into the `smb.conf` file's `[global]` share in the following format:

```
socket options = option1 option2
```

Linux supports a wide array of options, as shown in the table below:

Socket Options	Parameters	Use
TCP_NODELAY	None	Tells the TCP layer to send segments as soon as data is written by the application without trying to combine multiple writes from the application into one segment
SO_KEEPALIVE	None	Sends periodic keep-alive packets on the connection. If the remote end ever fails to respond, the smbd process for that connection can exit
SO_SNDBUF	bytes	Sets the transmit buffer size
SO_RCVBUF	bytes	Sets the receive buffer size
SO_SNDLOWAT	bytes	Sets the minimum number of free bytes that must be available in the socket send buffer before a select or poll call will return a writable for the socket
SO_RCVLOWAT	bytes	Sets the minimum number of data bytes that must be available in the socket receive buffer before a select or poll call will return a writable for the socket
IPTOS_LOWDELAY	None	Tunes the connection for low delays or latency. Typically used on a LAN connection
IPTOS_THROUGHTPUT	None	Tunes the connection for maximum throughput. Typically used on a LAN connection

On a LAN, the TCP_NODELAY option has the most impact on raw performance. If the TCP layer delays sending data written by smbd (because it is waiting for more) throughput is affected.

A commonly used setting for the socket options is:

```
socket options = TCP_NODELAY SO_KEEPALIVE
```

File caching and locking options

Samba's file caching and locking options have a definite impact on performance. Normally, opportunistic locking (oplocks) leads to speed increases for Windows clients.

Using oplocks can greatly improve Samba's performance. However, there may be a few instances where they decrease performance, especially when many clients are accessing a file in read/write mode.

You have the ability to disable oplocks on specific files or file types by using the `veto oplocks` option. The following will disable oplocks on database files:

```
veto oplock files = /*.dbf/*.DBF/
```

Logging

Samba permits a great deal of log information to be captured. Although this can be a great aid in diagnosing network problems, it can have a negative impact on performance, especially if you are logging at a level of 3 or greater. In this case, information is logged for every single network packet between the client and the Samba server.

For normal network operation a log level of 1 is sufficient. This will only log connects and disconnects from network shares. Adding the following line to the `[global]` share of the `smb.conf` file can change the logging configuration:

```
log level = 1
```

Raw read and writes

Some SMB clients support the use of raw read and write operations. If enabled, raw reads and write allow up to 65,535 bytes to be transferred in a single packet, which can dramatically speed up the transfer of large files.

Theses are controlled by the Samba `read raw` and `write raw` global options, both of which default to enabled in Samba 2.0.x. To enable them in other versions, use the following in your `smb.conf` file:

```
read raw = true
write raw = true
```

Wide links

The Samba wide links option controls whether a SMB client is allowed to follow links in the underlying Linux file system and is enabled by default.

If wide links are disabled for security reasons, it will negatively impact performance. This is because the Samba server must check each share's file and directory access ensuring it does not involve an outside link.

Tuning Linux

To provide the maximum possible performance from your Samba server, you will need to tune all aspects of the system, and not just the Samba software package. You will need to consider general Linux and hardware tuning, as well as tuning to obtain optimum performance from your server's storage subsystem and network interfaces.

File system caching

You need to devote the maximum amount of server RAM to performing file system caching and to keep the cached data in memory as long as possible. This ensures that the server is not bound by disk I/O speed when serving up commonly used files.

File and inode limits

Linux limits, like all other UNIX filesystems, the maximum number of files that can be open simultaneously. Linux also limits the number of inodes in the inode cache. If you have a large number of users and open files, you may exceed the system defaults and run out of file handles or inode cache entries. In these cases, you will need to modify the Linux kernel and increase the limits.

TIP:
If users are unable to open files, examine the server's log files. If any messages indicating that an inode or file handle limit has been reached, you need to increase the system's limits.

Network Topology Considerations

Your tuning should also extend to considering your network topology, and evaluation of whether network topology changes are warranted for performance reasons.

In fact, the network is probably the most critical factor in the end user experience. Server speed is irrelevant if the physical network is unable to process user's requests quickly. The network's raw bandwidth, the physical topology, and the amount of traffic are all factors determining how quickly data can be moved from server to client.

Bandwidth

The network's raw speed, or *bandwidth*, mainly determines network file system performance. The applications running on your Samba server determine the bandwidth requirement.

If the server's main purpose is casual data file storage, like documents, or moderate size data files, a 10Mbps network usually suffices. However, if the server is intended to run applications, store and access multimedia video clips, or run applications with large network-based files, a faster medium may be required.

Ethernet segmentation and switches

Segmenting an Ethernet installation with a switch can reduce the amount of collisions, or interferences in network traffic, on that network. Ethernet networks operate most efficiently when the number of collisions is zero or close to it.

Many switches allow a directly connected Ethernet adapter to run in full duplex mode. This means that the receive and transmit pairs no longer share the same wire inside a hub. It also allows simultaneous transmits and receives, potentially doubling a connected system's network bandwidth.

TIP:
If your network is more than 50% utilized and has numerous collisions, you should consider breaking the network into multiple segments.

By installing multiple Ethernet cards on the server performance can be improved further. These Ethernet cards with separate IP address can then be put on separate Ethernet segments, splitting the network load between the server and clients.

Server Filesystem Considerations

Linux's current filesystem, ext2, is a fairly high performance filesystem that allows you to configure, at creation, the block and inode size. It is also a fragmentation-resistant file system, meaning the kernel file system driver attempts instantaneous file storage. Finally, the filesystem is designed to limit the amount of data that can be lost after a crash, which offers a high degree of data protection.

Filesystem limits need to be considered for your Linux Samba servers. Specifically, the 2.2 series kernel release supports terabyte-sized drives, but imposes a 2GB file size limit.

Also, whenever an abnormal system shutdown occurs or the maximum number of filesystem mounts has been reached, error checking performed by `fsck` can be time consuming. The length of time depends on the drive size and on large drives could take as long as 20 minutes.

Memory Considerations

To achieve optimal performance from your Samba server, the system needs enough RAM to accommodate system operation without excessive disk swapping. Several factors contribute to the memory needs of a Samba server.

Samba processes requirements

Samba can be run in daemon mode or using the `inetd` "super server". There is at least one instance of the `smbd` process for each client connection, along with some shared memory overhead and, the initial `smbd` process if running in daemon mode.

It is important to note an instance of `smbd` is created for each client, not instance per share. The `smbd` process handles access to all shares for a connected client. The amount of memory each `smbd` process uses varies by system and also is dependent on what each client is doing, since `smbd` internal data structure's memory allocation is dependent on client activity.

On Linux, with shared libraries, memory consumption can be as low as 250KB per process but averages at least 500KB. The only way to get a good idea of a process's memory use is to examine the memory use for active `smbd` processes using the `ps` command.

TIP:
Do not run Samba in `inetd` mode. The Samba team has deprecated its support for performance and maintenance reasons.

Operating system requirements

Considering the way the system will be running determines the Samba server RAM sizing requirements. How much RAM the operating system needs to load the kernel and all the required device drivers must be reflected in the total system RAM requirements.

You check how much memory Linux itself uses with the `free` command. Use this command immediately after the system reboots and before users have logged in and Samba is running.

Calculating total server memory requirement

The table below sets some general guidelines that can be used for sizing a Samba server's memory needs. Most Linux systems can perform well with limited amounts of RAM, especially when not running X11, which is not required if the system will only be a Samba server:

Component	Recommendation
Base operating system	16 MB to 32 MB for the typical Linux system
Samba base `smbd`/`nmbd` processes	1MB
Shared memory for shared modules	1MB, default
Samba per-client `smbd` processes	0.5 MB per client
Other Processes	System dependent, but probably somewhere between 16MB to 64MB
File system cache	As much as possible

I/O Considerations

Your Samba system not only needs to provide access to files and printers, but must also provide a storage space that provides fast and reliable service to your users. Although you want to provide the best possible hardware for your situation, you don't want to waste money on unnecessary capabilities.

When choosing your Samba server's I/O subsystem components, you must consider the intended use of the system and your system's infrastructure.

Consider the number of users

The number of concurrent user accesses to the Samba server is important when choosing an I/O subsystem. There are two choices of Linux storage technology: either SCSI or IDE. SCSI technology permits multiple outstanding I/O requests, but can be expensive. IDE technology is less expensive, but only allows one outstanding request per device.

IDE's lack of I/O request caching makes it a poor choice for disk-intensive, heavily loaded Samba servers; in these cases, use SCSI. For smaller organizations or when the Samba server is only lightly used, IDE is the most cost-effective choice.

Consider expandability

In time, the storage space will be used up so you need a plan that will easily expand your system. Current IDE controllers permit a maximum of four drives. Though you can add additional IDE controllers, these require free slots on your server's motherboard. SCSI, on the other hand, can have up to 15 devices for "wide" controllers.

Consider reliability

The required reliability and acceptable downtime for the server should be an influencing factor in choosing an I/O subsystem. If the server is performing mission-critical functions and no downtime is acceptable, than a combination of highly reliable SCSI technology with a RAID-1 or RAID-5 real-time error-correction mechanism is needed.

However, if the Samba server is acting as a workgroup server where your down time is not as critical and you can afford to be down for several hours, you can probably get by with a single low-cost SCSI or IDE drive.

Chapter Summary

There are many factors to consider when you size your Samba system. Unfortunately, there is no hard rule for sizing a particular system with a given number of users or bandwidth. As a Samba administrator, you'll probably need to consult your network and system use policy to determine acceptable use guidelines, and then experiment with an array of hardware to come up with the best configuration that meets your current and future needs.

This page intentionally left blank.

Chapter 9: Troubleshooting Samba

Chapter Overview

Maintaining a successful Samba system requires the ability to recognize common problems and quickly correct them. By creating an iterative process of diagnosing problems you can reduce the amount of time required to eliminate the problem.

Chapter Objectives

After completing this chapter, you will be able to:

- describe the troubleshooting process.
- utilize the troubleshooting tools.
- recognize common Samba problems.

The Troubleshooting Process

This is usually an iterative process, very much like the scientific method, which has the form:

1. List the problem's symptoms, either by asking those who've experienced the problem or by observing first hand.
2. Formulate a hypothesis about the problem's cause, including whether the problem is local or remote, related to Samba or not, and so on.
3. Test your hypothesis on the supposed troubled system using standard diagnostic tools. Check the system logs, look at file permissions, free memory, logged on users, hung or unusual processes, etc.
4. Correct any problems you see and check if the system works as expected now.
5. If the problem isn't fixed, go back to step 2. If it is, document the cause and resolution.

Troubleshooting Tools

There are many tools available to aid in diagnosis and problem solving between Samba and Windows clients. The following lists some of the tools:

- `testparm` checks your `smb.conf` file and verifies it does not contain any invalid parameters.
- `smbclient` can verify that you can reach your Samba server and the desired shares are defined.
- Samba log files can be used to search for error messages relating to each Samba component. In Linux, theses logs are held in `/var/log/samba`.
- `smbstatus` allows you to check who has connections open to the Samba server and what shares they are accessing.
- `nmblookup` can check what names have been registered on the network and which system is the master browser.
- DOS's `net view` command will check from a Windows system to see what server shares are available or what network servers are available.
- DOS's `nbtstat` command can check on NetBIOS names and adapter status information.
- `ping` will show if a system is available and to ensure that name translation is working.
- `tcpdump` will dump packets off the wire and checks to see whether clients are talking to servers and vice versa.
- `ethereal` can analyze packets in detail and can show SMB packet corruption.
- `netmon` is a Microsoft program that is similar to `ethereal`.

Common Problems with Samba

There are several common problems with Samba, explained below.

Browsing problems

When a user tries to open up a share from a Windows client, he or she will use a browser, such as *Network Neighborhood*.

You can't browse the network

This may happen for several reasons:

- There is no master network browser possibly because none has been configured and your Windows system isn't configured to become a master browser. You will need to change the `local master` and `preferred master` parameters in the `smb.conf` file. When you restart your Samba server it will become a local master browser.
- The client's broadcast address doesn't agree with the server's broadcast address, so the server can't see requests from the clients. You need to ensure that the broadcast addresses agree.
- Your clients are configured to use WINS, but they cannot contact the WINS server to resolve names into IP addresses. This may be caused by the WINS server address being incorrect, packets not reaching the WINS server, or `nmbd` is not running. You need to ensure that your client is properly configured and the WINS server is reachable. Also, if Samba is acting as your WINS server, verify that `nmbd` is running.
- The guest account does not exist on the Samba server or the guest account parameter uses a non-existent account. You need to ensure that that the guest account exists.
- `smbd` is not running, so you should verify that Samba is starting properly.

You can't see any servers or can only see your own client

This can be caused by:

- A connectivity problem where the client is not seeing your Samba server's announcements. You should use `ping` to ensure that packets can travel in both directions between the client and server.
- The server is in a different workgroup from your client. You will need to change the server's `smb.conf` file `workgroup` parameter and restart Samba.

You can't browse a server

This occurs when the client can't talk to the server and may be caused for a couple reasons:

- `smbd` is not running on the server you are trying to browse.
- The server you are trying to browse has not registered with the WINS server you use for NetBIOS name to IP address translation, or WINS is not working in your network.

Share access problems

When you can access servers but not shares, a different kind of problem needs to be addressed.

Network name not found

This occurs when you have set a share, via for example "Map Network Drive", but can't access it and you get an error of "Network not found." This can be caused by:

- The share's path doesn't exist, which could be because of a typo or forgetting to create the share.
- The account you are using does not have access to the directory that the share points to in its `path` parameter.

You must supply a password to make this connection

If a dialog continually appears prompting you for a password, it could because of several things:

- Your password on the Samba server is different than the client you are logged in on. This is easily fixed by making sure all passwords are the same.
- You're using a Windows client that insists on encrypted passwords, but your Samba server does not use encrypted passwords. You need to either configure Samba to use encrypted passwords or switch off encrypted passwords in Windows.
- You're using a Windows client that insists on plain text passwords, but your Samba server uses encrypted passwords. You need to either configure Samba to use plain text passwords or switch on plain text passwords in Windows.

The network is busy

This probably occurs because your Samba server is configured to have a `host deny` for your workstation or has not included the workstation in `host allow`.

Access is denied

This is most likely due to a permission problem on your Samba server.

Users cannot write to a share

Since Samba shares are created by default as read-only, you most likely just need to make the share read-write.

Logon Problems

If you're trying to log in with a profile, but the Samba server isn't allowing you to log in or is giving you the wrong profile, then you might need to check the following:

No domain server

This problem occurs for several reasons:

- The netmask or broadcast address on your client does not match the one used by the Samba server.
- `nmbd` is not running on your Samba server, so the client can't perform the standard logon processing.
- Use `smbclient` to see the status of your Samba server, by entering:

```
$ smbclient –N –L servername
```

  You should see a browse list for your samba server, which should include the `[netlogon]` share.

Domain password is incorrect

There are several reasons why this message may occur:

- Your password is incorrect.
- Your client insists on using encrypted passwords, but your Samba server does not support them.
- The Samba server you are trying to log in to has a `hosts allow` or `hosts deny` parameter set that does not allow you access to the server.

Logon script does not run

You should check the following:

- Make sure the client is configured to perform a domain logon.
- Make sure that the `[netlogon]` share is configured properly and that share's path points to a valid directory on the Samba server.
- Check the file permissions in the `[netlogon]` directory to make sure the user can read the logon script.
- The `[netlogon]` share is not accessible to your users, or the `netlogon` script is not accessible to your users. You should check the `hosts allow` and `hosts deny` parameters along with making sure the correct permissions are set for the logon script.
- Your `netlogon` script has erroneous commands that cause it to end prematurely. You should run the script in a DOS window to determine if the script executes correctly.
- Your `netlogon` script was constructed in a Linux editor and is missing the additional end-of-line markers. You can use the `todos` command to add additional end-of line characters to the script.

Troubleshooting Printing Problems

If you can't print from your Windows clients, it's very likely a problem with the system's configuration, not Samba. There are several items you should check to validate your Samba's printer configuration, though:

- Make sure you are able to print from the Linux command-line.
- Use `testparm` and `testprns` to verify the `smb.conf` file's configuration parameters' syntax.
- Use the `smbclient` program to print.
- Specify the full path to the Linux printing software used by Samba.
- Check Samba's log file for messages relevant to printing.

The testprns and testparm programs

The Samba `testprns` program is designed to check for correct syntax of printer shares. It performs several *simple* checks to determine the validity of a specific printer share. You invoke the `testprns` program by entering:

```
$ testprns printername [printcapfile]
```

The `printername` specifies the printer share to test and the optional `printcapfile` refers to the printer capabilities file, which if not supplies defaults to `/etc/printcap`.

The `testparm` program is useful for troubleshooting printer share problems. Because it shows the default value for all Samba configuration parameters, `testparm` can check the actual values of parameters against the intended values.

Troubleshooting printing problems using smbclient

To use `smbclient` to test printer shares, it must be invoked with the service name to connect to and the `-P` option to connect to a printer. You can invoke the `smbclient` from the command-line by entering:

```
smbclient '\\server\share' -P
```

If you have successfully connected to the share, then a Windows client will also be allowed to connect to the share.

You should then attempt to print a file by putting it onto the share with the `put` command.

Example

To print the `readme.txt` file:

```
smb: \> put readme.txt
```

If `put` is successful then the printer will print the text. If not, then there is probably a problem with the printing system.

Error messages

- There are a couple error messages produced by `smbclient` that can be helpful in determining the cause of the printing problem:
- `Invalid network name in tree connect` – The server or share name given to `smbclient` is incorrect. Make sure the server name is correct and verify the share name exists in `smb.conf`.
- `Bad password - name/password pair in Tree connect or Session Setup are invalid` – The user and workgroup name specified in the `smbclient` command-line is not allowed to access the printer share, or the password is incorrect.
- `Connection to servername failed` – The specified server name can't be found on the network. It might not be present if it is a Windows server, or it might not be running Samba if it is a Linux server.

Daemon problems

There are a couple problems that may cause the `nmbd` or `smbd` daemons from starting.

There should always be *at least one instance of* `smbd` *and* `nmbd` *running*. This can be verified by entering:

```
$ ps -ef | grep [sn]mbd
```

If these are not running, check the error logs to determine the problem.

If the following message occurs when you start the daemons, your hostname is incorrect or can't be resolved:

```
GetHostbyname: Unknown host
```

You will need to add your server's name either to the network's DNS database or to the server's `/etc/hosts` file.

Chapter Summary

Creating a troubleshooting process that is adapted to your system will greatly reduce the amount of time spent on diagnosis. Also, by using the Samba tools, you should be able to pinpoint problems much more quickly than doing so by hand or random guessing. Finally, by familiarizing yourself with the common problems, you will be able to solve them much more efficiently.

This page intentionally left blank.

Appendices

Lab 1: Introducing Samba

Part A (10 minutes)

Answer the following questions:

1. What services does Samba provide?

2. How does Samba use NetBIOS? What are the different NetBIOS classes and where would each be useful?

3. What requirements are there on NetBIOS names?

4. What differences exist between share-level security and user-level security?

5. List and briefly describe the major Samba components.

6. Where are the Samba logs located and what information do they provide?

Lab 2: Samba Configuration

Part A (5 minutes)

Answer the following questions:

1. Describe the `smb.conf` file's standard shares.

2. Why you would want to include other files within your `smb.conf` file?

3. What happens if you include a configuration file in `smb.conf` that does not exist? Experiment on your workstation if uncertain.

Part B (10 minutes)

This part asks you to create a simple share on your Linux Samba server.

1. Create a share called `John` with the following properties:
   - The share access's the files in the home directory (`/home/john`) of user `john`.
   - Cannot be browsed.
   - Can be written to by the users `john` and `mike` only.
   - Guest users should not be allowed.
2. How would you modify the path parameter in the share to allow users to connect their personal home directory?

Part C (5 minutes)

This part asks you to include alternate configurations.

1. How would you provide a per-architecture configuration that didn't overwrite the primary Samba configuration?

Lab 3: Samba File Shares

Part A (5 minutes)

Answer the following questions:

1. If you have a user named `arrow`, a printer named `arrow`, and a share named `arrow`, what will be returned when you access the "arrow" share and why?

2. Describe the differences in “mostly open” and “mostly closed” policies and provide situations where each could be used.

Part B (30 minutes)

This part asks you to create several shares all with different access attributes.

1. Create a file share called `scratch` that all users, including guests, can use as temporary file space. The share should access the `/export/tmp` directory.
2. Create the directory `/export/tmp`, if it's not already present, and copy some files from `/tmp` into it. For example, `cp -a /tmp /export/tmp`.
3. Configure the share to disallow access to links pointing outside the share, show all dot files, make the `/hidden` directory invisible, disallow access to the `/forbidden` directory, and disallow the deletion of the `/critical` directory.
4. Restrict access to the share with the following provisions:
   - Allow access only from machines on the 192.168.0 network, except 192.168.0.1.
   - Disallow access from the user `robber`.
5. Force files to be created mode 1644. What do these permissions allow?

Lab 4: Samba Printer Shares

Part A (5 minutes)

Answer the following questions:

1. Why may you not want to use the standard `/etc/printcap` file in your global printer share?

2. Could you set up a per-user printer configuration? If so, how? If not, would you ever need this ability?

Part B (20 minutes)

This part gives you the opportunity to configure Samba print shares.

1. Create a global print share that should:
   - have a descriptive comment of the printer's type; you may make something up or inspect the classroom's printer.
   - use the `/etc/smbprinters` file for a printer listing.
   - set the spool directory to `/var/spool/lpd/samba` and create this directory if it doesn't already exist.
2. In the `/etc/smbprinters` file, add the following:

```
lp|default:\
    lp=/dev/null:\
    mx#0:\
    sh:\
    if=/var/spool/lpd/samba/lp/filter
```

3. Create a print share for this printer. The share should:
   - set the spool directory to `/var/spool/lpd/samba/lp` and create this directory if it doesn't already exist.
   - allow only the users `karen`, `john`, and `lisa` to print to it.

Lab 5: Password Management

Part A (5 minutes)

Answer the following questions:

1. What types of password authentication schemes does Windows and Samba support? Are they compatible?

2. What problems are caused by creating the `/etc/smbpasswd` file by running a script against the `/etc/passwd` file?

Part B (10 minutes)

This part asks you to configure your Samba server to authenticate against another SMB server.

1. Configure your Samba server to:
   - Act as a pass-through SMB authenticator. The instructor's machine should be used as the primary SMB password server.
   - Become a member of the `training` workgroup.
   - Encrypt passwords.

Part C (15 minutes)

This part allows you to synchronize your Samba server's passwords into the system's `/etc/passwd` file.

1. Configure your Samba server to:
   - synchronize the `/etc/passwd` file every time a user changes their Windows password.
   - use the `/bin/passwd` program to do so. You will need to investigate the `passwd chat` option to see how to synchronize passwords for Red Hat Linux.

Lab 6: Serving Logons and Profiles

Part A (5 minutes)

Answer the following questions:

1. What services does a logon server provide?

2. What problems are there with the differences between Windows versions that makes servicing profiles consistently difficult?

Part B (10 minutes)

This part asks you to configure your Samba server to provide logon script support.

1. Create the network login share to pull profiles from the `/export/profiles` directory (create this directory if it isn't already present). The share shouldn't be accessible by guests, browse-able, or able to be written to.
2. Configure Samba to execute the `login.bat` file whenever clients connect.

Lab 7: Accessing SMB Shares from Linux

Part A (5 minutes)

Answer the following questions:

1. Why would you want to mount a Windows file system?

2. Which of the following scenarios would be the best way to make a backup of a SMB share: 1) mount the share and tar it with the `tar` command, or 2) use the `smbtar` program? Justify your answer.

Part B (10 minutes)

This part asks you to use and experiment with the `smbclient` program.

1. Use the `smbclient` program to access your neighbor's Samba server, connecting to the `scratch` share. Retrieve some files with the `get` command.
2. Type `!ls -ltr` at the `smb: \>` prompt and look at the retrieved files. What are their permissions? Does this surprise you?
3. Type `help` at the `smb: \>` prompt. What commands are present? Think about how you could use these commands. Try some of them out.
4. Exit the program by typing `quit`.

Part C (30 minutes)

This part asks you to experiment with the `smbmount` and `smbclient` commands.

1. Create the directory `/mnt/scratch`.
2. Mount your own Samba server's `scratch` share to `/mnt/scratch`.
3. Tar the contents, while timing the execution, into the `/tmp` directory by typing `time tar -cf /tmp/smbmount.tar /mnt/scratch`. How long did it take?
4. Tar the files, while timing the execution, with the smbclient program, by typing `time smbclient //localhost/scratch -I yourip -Tc /tmp/smbclient.tar`. Replace "yourip" with your computer's IP address. How long did it take?

Lab 8: Samba and Performance

Part A (5 minutes)

Answer the following questions:

1. What are the advantages and disadvantages of opportunistic locks?

2. How does logging affect system performance?

3. Should you enable the `wide links` parameter? Why or why not.

Part B (40 minutes)

This part asks you to gauge how many resources a Samba system might need given certain operating parameters and how the server should be configured. To complete this part, group with several of your classmates.

1. Suppose you have a Linux Samba server in a medium-sized business with approximately 75 users. Its characteristics are:
   - Provides Samba file and printer shares of approximately 1GB of text files (source code) and programs (which are never written to from Samba).
   - Exports all files, read-only, with NFS.
   - Uses CVS to access the source code.
2. Should you use oplocks on the source code share? The programs share? Why or why not?
3. Fill in the table below, using some values from your setup:

Component	*Size*
System base memory (reboot the system in single user mode and type `free`)	
Samba usage (type `ps -gxl | grep [sn]mbd` and add up the values in the 7th column)	
CVS server (approximate)	32MB
NFS (requires "0" memory because the memory is used in the kernel, not taken from user memory)	0MB
Swap (twice the sum of all previous rows)	
Total	

4. Is this a feasible amount of memory? Would you want more?

References

The materials below provide invaluable Samba administrative help and should be close at hand for Samba administrators:

- Kabir, M. **Red Hat Linux 6 Server**. M&T Books. 1999.
- `http://www.samba.org/`. Samba. The Samba project's primary page.
- `http://razor.bindview.com/tools/`. Razor Tools. The home page for the `pwdump2` utility discussed in Chapter 5.