

The Unofficial Samba HOWTO

David Lechnyr <david@lechnyr.com>

Mon Apr 7 16:57:39 PDT 2003



1. Introduction

- 1.1 [Background](#)
- 1.2 [Terminology](#)
- 1.3 [Related Projects](#)
- 1.4 [What's Not Covered](#)
- 1.5 [Miscellaneous](#)

2. Installation

- 2.1 [Downloading Samba](#)
- 2.2 [Verifying Samba's PGP signature](#)
- 2.3 [Installing Samba](#)
- 2.4 [Configuring Samba](#)
- 2.5 [Running Samba \(137/udp, 138/udp, 139/tcp\)](#)
- 2.6 [Running Samba \(445/tcp\)](#)
- 2.7 [Examining the Samba Package](#)

3. Using Samba

- 3.1 [Connecting to Samba from a Windows machine](#)
- 3.2 [Mounting an SMB share on a Linux machine](#)
- 3.3 [Caveats](#)

4. Troubleshooting

- 4.1 [Samba Logfile Settings](#)
- 4.2 [Marshalling/Unmarshalling](#)
- 4.3 [Using tcpdump](#)
- 4.4 [Troubleshooting the Server](#)

- [4.5 Troubleshooting the Client](#)
- [4.6 Troubleshooting Example](#)

5. Optimizing Samba

- [5.1 Oplocks](#)
- [5.2 Socket Options](#)
- [5.3 Additional Tuning Options](#)

6. Samba as a PDC

- [6.1 PDC Configuration](#)
- [6.2 Users and Groups](#)
- [6.3 Joining your Samba Domain](#)
- [6.4 Roaming Profiles](#)
- [6.5 Login Scripts](#)
- [6.5 Linefeeds \(CR/LF\)](#)

7. Security and Samba

- [7.1 Restricting Client Access](#)
- [7.2 Shadows and Light](#)
- [7.3 Samba and Firewalls \(iptables\)](#)
- [7.4 Tunneling SMB through SSH](#)
- [7.5 Samba and SSL](#)
- [7.6 Additional Techniques](#)

8. Appendix

- [8.1 SMB Methodology](#)
 - [8.2 Samba Startup Script](#)
 - [8.3 Windows Event Manager](#)
 - [8.4 Sample smb.conf](#)
 - [8.5 Additional Resources](#)
 - [8.6 Epilogue](#)
-

1. Introduction

"If you understand what you're doing, you're not learning anything." -- Anonymous

Samba is a file and print server for Windows-based clients using TCP/IP as the underlying transport protocol. In fact, it can support any SMB/CIFS-enabled client. One of Samba's big strengths is that you can use it to blend your mix of Windows and Linux machines together without requiring a separate Windows NT/2000/2003 Server. Samba is actively being developed by a loose-knit group of about 20 people from all over the world and was originally developed by Andrew Tridgell.

1.1 Background

Once long ago, there was a buzzword referred to as DCE/RPC. This stood for Distributed Computing Environment/Remote Procedure Calls and conceptually was a good idea. It was originally developed by Apollo/HP as NCA 1.0 (Network Computing Architecture) and only ran over UDP. When there was a need to run it over TCP so that it would be compatible with DECnet 3.0, it was redesigned, submitted to The Open Group, and officially became known as DCE/RPC. Microsoft came along and decided, rather than pay \$20 per seat to license this technology, to reimplement DCE/RPC themselves as MSRPC. From this, the concept continued in the form of SMB (Server Message Block, or the "what") using the NetBIOS (Network Basic Input/Output System, or the "how") compatibility layer. You can run SMB (i.e., transport) over several different protocols; many different implementations arose as a result, including NBIPX (NetBIOS over IPX, NwLknNb, or NWNBLink) and NBT (NetBIOS over TCP/IP, or NetBT). As the years passed, NBT became the most common form of implementation until the advance of "Direct-Hosted TCP" — the Microsoft marketing term for eliminating NetBIOS entirely and running SMB by itself across TCP port 445 only. As of yet, direct-hosted TCP has yet to catch on. And so the story goes...

Perhaps the best summary of the origins of SMB are voiced in the 1997 article titled, [CIFS: Common Insecurities Fail Scrutiny](#):

*"Several megabytes of NT-security archives, random whitepapers, RFCs, the CIFS spec, the Samba stuff, a few MS knowledge-base articles, strings extracted from binaries, and packet dumps have been dutifully waded through during the information-gathering stages of this project, and there are *still* many missing pieces... While often tedious, at least the way has been generously littered with occurrences of clapping hand to forehead and muttering 'crikey, what are they *thinking*?!'"*

Anyone even remotely considering getting involved in implementing a Samba server should review Linux Magazine's [Understanding the Network Neighborhood](#) article by Christopher R. Hertel.

1.2 Terminology

- **SMB**: Acronym for "Server Message Block". This is a Microsoft's file and printer sharing protocol.
- **CIFS**: Acronym for the "Common Internet File System". Around 1996, Microsoft apparently decided that SMB needed the word "Internet" in it, so they changed it to CIFS.
- **Direct-Hosted**: A method of providing file/printer sharing services over port 445/tcp only, using DNS for name resolution instead of WINS.
- **IPC**: Acronym for "Inter-process Communication". A method to communicate specific information between programs.
- **Marshalling**: – A method of serializing (i.e., sequential ordering of) variable data suitable for transmission via a network connection or storing in a file. The source data can be re-created using a similar process called unmarshalling.
- **NetBIOS**: Acronym for "Network Basic Input/Output System". This is not a protocol; it is a *method* of communication across an existing protocol. This is a standard which was originally developed for IBM by Sytek in 1983. To exaggerate the analogy a bit, it can help to think of this in comparison your computer's BIOS — it controls the essential functions of your input/output hardware — whereas NetBIOS controls the essential functions of your input/output traffic via the network. Again, this is a bit of an exaggeration but it should help that paradigm shift. What is important to realize is that NetBIOS is a *transport standard*, not a protocol. Unfortunately, even technically brilliant people tend to interchange NetBIOS with terms like NetBEUI without a second thought; this will cause no end (and no doubt) of confusion.
- **NetBEUI**: Acronym for the "NetBIOS Extended User Interface". Unlike NetBIOS, NetBEUI is a *protocol*, not a standard. It is also not routable, so traffic on one side of a router will be unable to communicate with the other

side. Understanding NetBEUI is not essential to deciphering SMB; however it helps to point out that it is not the same as NetBIOS and to improve your score in trivia at parties. NetBEUI was originally referred to by Microsoft as "NBF", or "The Windows NT NetBEUI Frame protocol driver". It is not often heard from these days.

- **NBT**: Acronym for "NetBIOS over TCP"; also known as "NetBT". Allows the continued use of NetBIOS traffic proxied *over* TCP/IP. As a result, NetBIOS names are made equivilant to IP addresses and NetBIOS name types are conceptually equivilant to TCP/IP ports. This is how file and printer sharing are accomplished in Windows 95/98/ME. They traditionally rely on three ports: NetBIOS Name Service (nbname) via UDP port 137, NetBIOS Datagram Service (nbdatagram) via UDP port 138, and NetBIOS Session Service (nbssession) via TCP port 139. All name resolution is done via WINS, NetBIOS broadcasts, and DNS. NetBIOS over TCP is documented in RFC 1001 (Concepts and methods) and RFC 1002 (Detailed specifications).
- **W2K**: Acronym for Windows 2000 Professional or Server
- **W3K**: Acronym for Windows 2003 Server

1.3 Related Projects

Currently, there are two projects that are directly related to Samba: SMBFS and CIFS network *client* file systems for Linux, both available in the Linux kernel itself.

- SMBFS (Server Message Block File System) allows you to mount SMB shares (the protocol Windows 95/98/ME, Windows NT/2000/XP and OS/2 Lan Manager use to share files and printers over local networks) and access them just like any other Unix directory. This is useful if you just want to mount such filesystems without being a SMBFS server.
- CIFS (Common Internet File System) is the successor to SMB, and is actively being worked on in the upcoming version of the Linux kernel (2.5/2.6). The intent of this module is to provide advanced network file system functionality including support for dfs (heirarchical name space), secure per-user session establishment, safe distributed caching (oplock), optional packet signing, Unicode and other internationalization improvements, and optional Winbind (nsswitch) integration. If you enable CONFIG_CIFS in the Linux kernel, be aware that it is currently in an early development stage and may not be as stable as the existing CONFIG_SMB_FS option.

Again, it's important to note that these are implementations for *client* filesystems, and have nothing to do with acting as a file and print server for SMB/CIFS clients.

1.4 What Isn't Covered

If there's anyone who wants to take any of these items on, send me a draft and I'll see about incorporating it with this document ;-)

- SWAT
- Printing
- Disk Quotas
- Kerberos support
- Winbind

1.5 Miscellaneous

- The official version of this document is located at <http://hr.uoregon.edu/davidrl/samba/>.

- According to recent statistics for this document, there is a 26% chance that you're reading this on a Linux (non-Windows) machine — if so, well done ;-))
- **Credits:** The logo at the top of the article was created by "Sluggite Bob". "Sluggie Freelance" is © Pete Abrams. "Using Samba" is © 1999 by O'Reilly & Associates, Inc. We are, of course, indebted to the Samba Team for their diligent and excellent work! Thanks to: Rob Muggridge on encrypted password issues and Win95; Kai Price for suggesting that .tgz versions of this document be made available.
- David Lechnyr is a Network Manager at the Human Resources department of the University of Oregon. He holds a Master's Degree in Social Work along with his MCSE+I, CNE, and CCNA certifications. He has been working with Linux for the past seven years, with an emphasis on systems security, network troubleshooting, and PHP/MySQL integration.
- No SGML tools were harmed during the creation of this document (as it should be).
- This document is Copyright © 2003 David Lechnyr. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation. A copy of the license is available at <http://www.gnu.org/licenses/fdl.txt>.

This document was lovingly handcrafted on a Dell Latitude C400 laptop running Slackware Linux 9.0, in case anyone asks.

2. Installation

"All parts should go together without forcing. You must remember that the parts you are reassembling were disassembled by you. Therefore, if you can't get them together again, there must be a reason. By all means, do not use a hammer." — IBM maintenance manual, 1975

2.1 Download Samba

This one is easy. You should feel quite proud ;-) If you'd rather use a precompiled binary or an RPM, this document probably isn't the one you want.

It's worth mentioning that you should be running the latest stable version of Samba. As all versions prior to 2.2.8a have a critical security flaw; you should at least be running this version or later.

Samba files are available at <http://www.samba.org>. You can retrieve the file with something like:

```
$ wget http://us1.samba.org/samba/ftp/samba-2.2.8a.tar.gz
```

2.2 Verifying Samba's PGP signature

In these days of insecurity, it's strongly recommended that you verify the PGP signature for any source file before installing it. According to Jerry Carter of the Samba Team, only about 22% of all Samba downloads have had a corresponding PGP signature download (a very low percentage, which should be considered a *bad* thing). Even if you're not downloading from a mirror site, verifying PGP signatures should be a standard reflex.

With that said, go ahead and download the following files:

```
$ wget http://us1.samba.org/samba/ftp/samba-2.2.8a.tar.asc
$ wget http://us1.samba.org/samba/ftp/samba-pubkey.asc
```

The first file is the PGP signature for the Samba source file; the other is the Samba public PGP key itself. Import the public PGP key with:

```
$ gpg --import samba-pubkey.asc
```

And verify the Samba source code integrity with:

```
$ gzip -d samba-2.2.8a.tar.gz
$ gpg --verify samba-2.2.8a.tar.asc
```

If you receive a message like, "Good signature from Samba Distribution Verification Key..." then all is well. The warnings about trust relationships can be ignored. An example of what you would *not* want to see would be:

```
gpg: BAD signature from "Samba Distribution Verification Key"
```

2.3 Installing Samba

Extract the package and change into the "source" subdirectory:

```
$ tar xf samba-2.2.8a.tar
$ cd samba-2.2.8a
$ cd source
```

Next, you need to configure the Samba package for your specific system. Some possible options you can specify include:

- **--with-smbmount** -- Create the [smbmount\(8\)](#) program and /sbin/mount.smbfs (which is actually just a soft link to smbmount). You'll still need to enable SMB filesystem support in the kernel (CONFIG_SMB_FS) for this to work. If you plan to mount other SMB file systems on your Linux box, you'll definately want to enable this option.
- **--with-smbwrapper** -- Create [smbsh\(1\)](#), a shell (aka command interpreter) that allows you to access an NT filesystem using UNIX commands such as ls(1) and egrep(1).
- **--with-automount** -- Include automount(8) support for SMB filesystems. If you don't use automount, you probably don't need this option.
- **--with-syslog** -- Enable support for logging directly to syslog(2).
- **--with-fhs** -- Install the files in FHS-compliant directories rather than in /usr/local/samba (see below).

If you're just starting out with Samba, you probably just want to use:

```
$ ./configure
```

If you're interested in a 100% [FHS](#) (Filesystem Hierarchy Standard) setup, you might try:

```
$ ./configure --with-fhs --prefix=/usr --sysconfdir=/etc
--localstatedir=/var
```

For the more experienced administrator, you might be interested in trying something like:

```
$ CFLAGS="-O2 -march=i686" ./configure \
--with-fhs \
```

```
--prefix=/usr \
--sysconfdir=/etc \
--localstatedir=/var \
--with-automount \
--with-smbmount
```

It's worth noting that if you get the error message:

```
bash: ./configure: No such file or directory
```

then you're 1) in the wrong directory and 2) haven't followed directions. We leave the answer as an exercise to the reader ;~)

A word about optimization: You can run configure with the CFLAGS option of `-O` (also known as `-O1`), `-O2`, or `-O3`. In theory, the higher the number, the faster your program will run (and the slower it will compile). During development, most programmers usually use no optimization and only activate it when the code is ready for release. By default, Samba compiles with basic optimization (`-O`) and debugging disabled. You can experiment with this option to see if optimization will work for your specific situation. If you're unsure or your system doesn't have a lot of memory for compiling programs, leave this option alone.

A word about architecture: If you'd like to generate code specific to your CPU type, you can pass the option to the `--march` parameter using one of the following (for gcc 2.95.3): `i386`, `i486`, `i586`, `i686`, `k6`. There are additional options for non-Intel architectures and gcc 3.2 (see ``info gcc``). Note that if you choose to compile for an `i686`, your code won't run on anything that's has an `i586` CPU or slower. For example, on our Intel Pentium III-Xeon servers running with gcc 3.2, we use `"--march=pentium3"`. If you don't know what any of this means, you can safely skip this step.

You should resist the temptation to include any parameters that are being deprecated in the upcoming Samba 3.0 version: `--with-codepagedir`, `--with-krb4-base`, `--with-msdfs`, `--with-ssl`, `--with-winbind-auth-challenge`, `--with-winbind-ldap-hack`. It's worth noting that the following list of new options (not exhaustive) are currently planned for inclusion in Samba 3.0: `--with-ldap`, `--with-mysql-prefix`, `--with-xml-prefix`, `--with-manpages-langs={en,ja,pl}`, `--with-python`.

Regardless of the options you use, you need to compile the package next, making sure to create the following FHS-compliant directories as well:

```
$ make

# mkdir -p /usr/share/samba/codepages
# mkdir -p /var/{spool,cache,log}/samba
# mkdir -p /etc/samba/private && chmod 700 /etc/samba/private
# touch /etc/samba/private/smbpasswd && chmod 600
/etc/samba/private/smbpasswd
# cp -f ../examples/smb.conf.default /etc/samba/
# touch /etc/samba/smb.conf

# make install
```

You should also make sure that you keep a copy of all the Samba documentation:

```
# mkdir -p /usr/doc/samba-2.2.8a
# cd ..
```

```
# cp -fa COPYING Manifest README \  
  Read-Manifest-Now Roadmap WHATSNEW.txt \  
  docs examples /usr/doc/samba-2.2.8a/
```

2.4 Configuring Samba

Samba's configuration file resides in `smb.conf`, which you will need to create in `/usr/local/samba/lib/` (default install) or `/etc/samba/` (FHS). A good way to determine what the default values for a particular version of Samba are is to create an empty `smb.conf` file, run `testparm` against it and `grep` for the parameter name you're curious about. For those that just want to get started, here's a quick & dirty (although not secure) `smb.conf` to try out that should work under most circumstances. For specifics, see [smb.conf\(5\)](#).

```
[global]  
  encrypt passwords = yes  
  guest account = smbguest  
  hide unreadable = yes  
  hide dot files = yes  
  log level = 1  
  log file = /var/log/samba/%m.log  
  netbios name = myserver  
  security = user  
  server string = "Is It Not Nifty?"  
  socket options = TCP_NODELAY IPTOS_LOWDELAY SO_RCVBUF=8192  
SO_SNDBUF=8192  
  time server = yes  
  wins support = yes  
  workgroup = myworkgroup  
  
[homes]  
  browseable = no  
  read only = no  
  create mode = 0600  
  directory mode = 0700  
  hide files = /*.pst/  
  
[public]  
  browseable = yes  
  create mode = 0666  
  directory mode = 0777  
  guest ok = yes  
  guest only = yes  
  path = /home/public  
  read only = no  
  veto files = /*.com/*.exe/*.scr/*.dll/*.{*}/  
  
[private]  
  browseable = yes  
  create mode = 0660  
  directory mode = 0770  
  force group = +users
```

```

path = /home/private
read only = no
valid users = +staff +students
veto files = /*.com/*.exe/*.scr/*.dll/*.{*}/
write list = +staff

```

You'll want to make sure that the following items exist for the above configuration file to work correctly:

```

# mkdir -p /home/private /home/public
# mkdir -p /etc/samba/private
# touch /etc/samba/private/smbpasswd
# touch /etc/samba/private/secrets.tdb
# chmod 0770 /home/private
# chmod 1777 /home/public
# groupadd smbguest
# groupadd staff
# groupadd students
# groupadd users
# useradd -d /home/smbguest -g smbguest -s /bin/false -m smbguest
# useradd -d /home/fred -g users -G staff -s /bin/false -m fred
# chmod 0700 /home/fred /home/guest
# smbpasswd -a fred
# testparm

```

If you'd like to use an `lmhosts(5)` file, it should be created in the same directory as your `smb.conf` file.

2.5 Running Samba

To run a traditional installation of Samba, edit your `/etc/services` file to include:

```

netbios-ns      137/udp      #NETBIOS Name Service (used by nmbd)
netbios-dgm    138/udp      #NETBIOS Datagram Service (used by nmbd)
netbios-ssn    139/tcp      #NETBIOS Session Service (used by smbd)

```

And launch the daemons with:

```

# /usr/sbin/smbd -D
# /usr/sbin/nmbd -D

```

Alternately, you can run Samba from `inetd` by modifying your `/etc/inetd.conf` file to include:

```

netbios-ssn stream tcp nowait.400 root /usr/sbin/smbd smbd
netbios-ns dgram udp wait root /usr/sbin/nmbd nmbd

```

2.6 Running Samba (445/tcp)

If you're feeling daring, you can try running Samba via port 445/tcp only using a method called "Direct-Hosted". Note that this method is not as widely tested (or supported) as running a traditional install of Samba. Only Windows 2000/XP/2003 support this method of connecting to Samba, so if you have other clients this option is not one that you should use. Modify your `/etc/services` file to include:

```
microsoft-ds 445/tcp #Direct-Hosted Service (used by smbd)
```

And modify your `/etc/inetd.conf` file to include:

```
microsoft-ds stream tcp nowait.400 root /usr/sbin/smbd smbd
```

Please be aware that NetBIOS naming conventions do not work under this method; you must use the explicit IP address of the server instead. For example:

```
NET USE F: \\192.168.1.1\sharename
```

A few other items to be aware of when using Samba to listen to port 445/tcp:

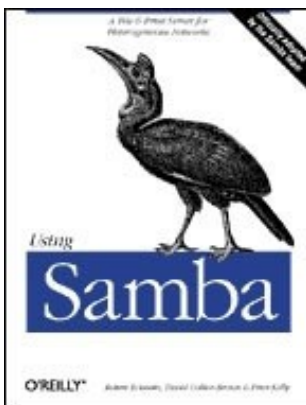
1. Logon scripts no longer reliably run when logging into a Samba PDC server.
2. Shortcuts to programs on the network will issue a security warning when run, as Windows believes you are trying to run a program across the Internet. Adjusting security settings in Internet Explorer does not seem to resolve this.
3. The Samba variable `%L` will no longer resolve to the NetBIOS name of the server.
4. The syntax "net use `* /homes`" will no longer work; you must explicitly use "net use `* \\192.168.1.1\homes`".
5. ...Probably many others...

Recommendation: Avoid using this method if your client base is not exclusively Windows 2K/XP and/or you are having Samba act as a PDC.

2.7 Examining the Samba Package

It's worth noting that the Samba source includes the following useful files and directories:

- **WHATSNEW.txt** – Changes since the last version. Of all the documents, this is probably the most important one to be aware of.
- **docs** – Directory with Samba documentation, both old and new
- **examples** – Directory with sample configuration files for Samba
- **source** – Directory containing the Samba source code



Also, you'll kick yourself if you haven't yet realized that the full, unabridged version of *Using Samba 1st Edition* by O'Reilly is included in the source tarball for free in `docs/htmldocs/using_samba/`. *Using Samba 2nd Edition* is not available free of charge, however a sample chapter is available for [download](#).

Once you've installed Samba, there are a few tools you definately want to be familiar with:

- [nmbd](#) – Server program that provides NetBIOS over IP naming services to clients
- [smb.conf](#) – Samba main configuration file
- [smbd](#) – Server program that provides SMB/CIFS file/print services to clients
- [smbmount](#) – Mount an smbfs filesystem
- [smbpasswd](#) – Change or create a user's SMB password
- [testparm](#) – Check an smb.conf configuration file's syntax

Additional tools at your disposal are:

- [findsmb](#) – list info about machines that respond to SMB name queries on a subnet
- [make_unicodemap](#) – construct a unicode map file for Samba
- [make_smbcodepage](#) – construct a codepage file for Samba
- [nmblookup](#) – NetBIOS over TCP/IP client used to lookup NetBIOS names
- [rpcclient](#) – Tool for executing client side MS-RPC functions
- [smbcacls](#) – Set or get ACLs on an NT file or directory names
- [smbclient](#) – ftp-like client to access SMB/CIFS resources on servers
- [smbcontrol](#) – Send messages to smbd, nmbd or winbindd processes
- [smbmnt](#) – Helper utility for mounting SMB filesystems
- [smbspool](#) – Send a print file to an SMB printer
- [smbstatus](#) – Report on current Samba connections
- [smbtar](#) – Shell script for backing up SMB/CIFS shares directly to UNIX tape drives
- [testprns](#) – Check printer name for validity with smbd
- [wbinfo](#) – Query information from winbind daemon
- [winbindd](#) – Name Service Switch daemon for resolving names from NT servers

Reference: [samba-2.2.8a/source/configure](#)

3. Using Samba

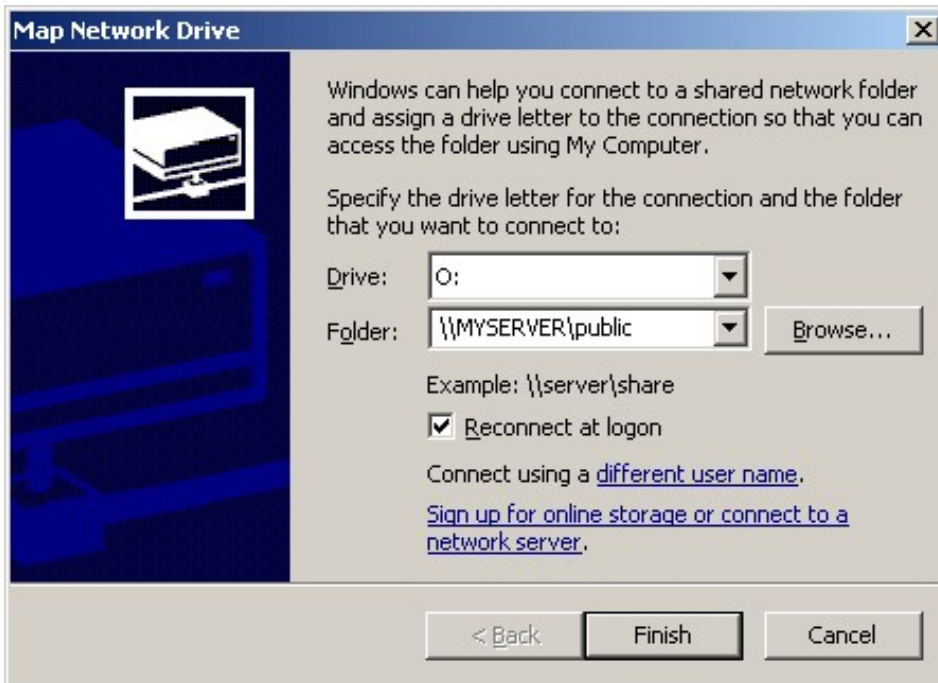
"How do you power off this machine?" -- Linus Torvalds, when upgrading linux.cs.helsinki.fi, and after using the machine for several months

Samba's goal is to provide a bridge between Windows and Linux. Which sometimes presents a problem -- not everyone versed in Linux understands Windows networking concepts. The reverse is also true. Hopefully, this section will provide some basics to get you started in actually using Samba from both sides of the coin.

3.1 Connecting to Samba from a Windows machine

Ironically, this is a frequent question from Linux administrators. Most Samba documents give vague instructions on "use Network Neighborhood" but never actually say how to connect. From this, we draw the feature of *mapping*.

Mapping is a Windows networking term which implies that you will be creating a *simulated* disk drive that actually points to a remote share on another computer. For example, I might have a F: drive that is actually a shortcut to one of my folders available on my Samba server.



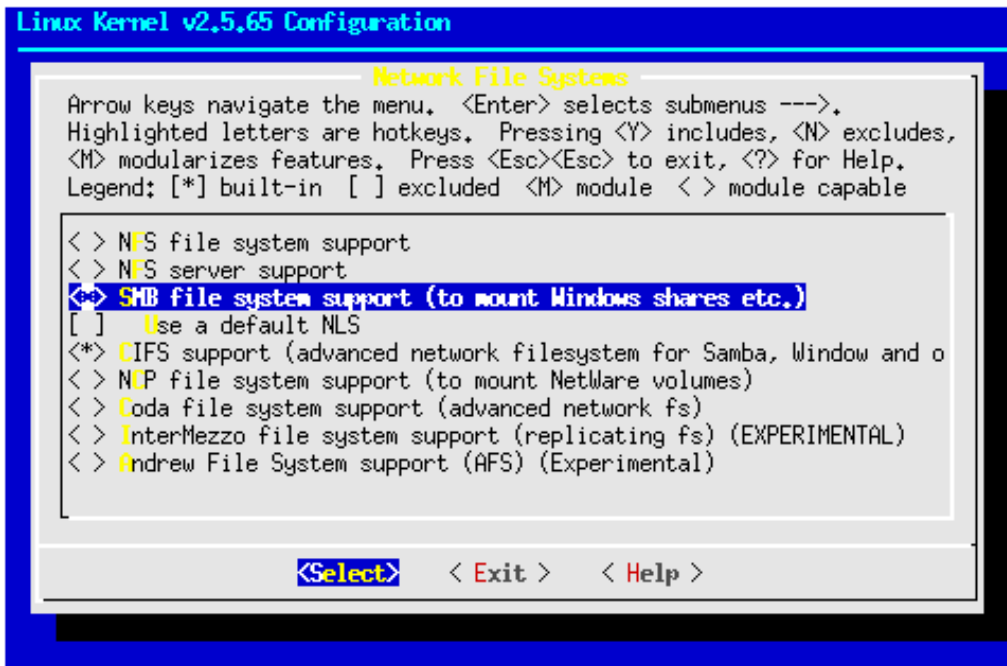
To create this simulated drive, you will probably want to use one of two methods: Using the Windows GUI (Graphical User Interface) or using the Command Prompt.

To map a drive using the *Windows GUI*, open up either *My Computer* or *Windows Explorer*. On the toolbar are many options; look for one that provides a list that includes the words "Map Network Drive". Under Windows XP, this option is under the *Tools* menu. Once selected, a new box will open up. In the *Drive* box, click a drive letter that you wish to use (e.g., F:). In the *Folder* box, type the UNC path for the server and shared resource you wish to connect to. For example, if your Samba server's NetBIOS name is "MYSERVER" and one of the shares it provides is called "public", you would type in "\\MYSERVER\public" (minus quotes). If you want this connection to be permanent across reboots, put a checkmark in the box.

To map a drive using the *Command Prompt*, click on the *Start* button, choose *Run*, and type in the command "cmd" (minus quotes). This will open up a command prompt (aka MS-DOS prompt). To connect to your Samba server using the same server name and share name in the previous example, you would simply type: "NET USE F: \\MYSERVER\public" (minus quotes).

3.2 Mounting a SMB share on a Linux machine

In order to mount an SMB share on your Linux machine, regardless of whether or not the share is on a Samba machine, you'll need to prepare a few things. First, you'll need SMB file system support (CONFIG_SMB_FS) in your Linux kernel. If you're running Linux 2.5/2.6, you can experiment with also enabling CIFS support. You can read about both of these filesystems in [smbfs.txt](#) (somewhat useless) and [cifs.txt](#) in the Linux kernel Documentation/filesystems/ directory.



You *may* need to compile Samba using the "--with-smbmount" parameter (and the --with-automount, if relevant). It's a little unclear what advantages this gives you over the kernel-built smbfs/cifs. What installing Samba with these options does is to create the `smbmount(8)` program and `/sbin/mount.smbfs`, which is actually just a soft link to `smbmount(8)`.

Once these two items are in place, you can simply issue something like:

```
# mount -t smbfs -o username=fred,password=secret //192.168.1.1/public
/mnt/public
```

If you want to use NetBIOS names instead of IP addresses, you'll need to make sure that you have created an `lmhosts(5)` file in the same directory as your `smb.conf` file (even if your `smb.conf` file is empty).

You could easily adapt this for use with your `/etc/fstab` file or for use with `automount(8)`.

If you don't compile "SMB file system support" into your kernel, you'll get the following error message instead:

```
mount: fs type smbfs not supported by kernel
```

3.3 Caveats

As with all things, you must be aware of a few Windows/Linux connectivity "ground rules":

- **Case sensitivity:** NetBIOS names, workgroups, domains, etc., are not case-sensitive as far as Samba is concerned. However, some clients (notably Win98/ME/XP) do odd things when there is a difference in upper/lowercase for such things. **Recommendation:** Save your sanity and either use all upper- or all lower-case NetBIOS names, workgroups, domains, etc., on both server *and* workstation.
- **Packet sniffing:** If you plan to diagnose why things aren't happening, become familiar with `tcpdump(1)` and its use and syntax. Otherwise, you're going to end up diagnosing things in the dark, which isn't very useful.
- **Mailing Lists:** If you plan on getting help, make sure to subscribe to the samba mailing list (available at

<http://www.samba.org>). Optionally, you could just search mailing.unix.samba at <http://groups.google.com>.

- **Samba is a connectivity tool:** It's meant to bridge the gap between Windows and Linux, not to solidify the place of Windows over Linux. As always, your mileage may vary (see figure 1).



Figure 1: The Windows Desktop

4. Troubleshooting

"You're reasoning is excellent -- it's only your basic assumptions that are wrong." -- Anonymous

4.1 Samba Logfile Settings

To stand a chance of determining the answer to the classic question of "what went wrong" you'll need to understand how to enable Samba logging and how to decipher its cryptic log messages as well. First, here are a few of the `smb.conf(5)` parameters that directly affect logging:

- **`debug hires timestamp = yes`**

Use microseconds, instead of seconds, in printing out the timestamps in the log messages. Chances are you won't

need this unless you're doing massive debugging.

- ***debug pid = yes***

Add the process-id of the specific samba process to the logfiles. This can help you if you're trying to determine which process outputs which log message. Requires the ***debug timestamp*** parameter to be enabled as well.

- ***debug timestamp = no***

Actually, this is the default setting and is generally quite useful. You might want to set this to 'no' if you're running at a high debug level and don't want the timestamps to be printed out as well, which can be distracting. It's also helpful when comparing logfiles with diff(1), although one caveat is that you don't get the name and function of the code that is generating the message (catch-22).

- ***debug uid = yes***

Print out the effective uid (euid), effective gid (egid), uid and gid along with the the timestamp message headers in the log files. Since Samba runs as root and at other times runs as the connected user, this can be helpful for certain cases. Requires the ***debug timestamp*** parameter to be enabled as well.

- ***debug level***

This is a synonym for ***log level***

- ***log file = %m.log***

The name and location of the Samba log file. The '%m' allows you to log machine-specific connection information in different log files.

- ***log level = 3***

Enables logging at the level of detail specified by an integer. Higher numbers mean more debugging information. It's generally a good idea to leave this at '1' to catch basic, day-to-day errors that can crop up. When researching a problem, you'll probably want to use a log level of '3' to gather enough details about what's happening. Note that it's often what the log *doesn't* say which is the real eye-opener when chasing down problems; being vaguely aware of how the log files should look will benefit you more often than not! When discussing in-depth issues with a Samba developer, a log level of '10' is generally preferred (although you won't win a lot of friends if you post the entire thing to the mailing list ;-)

- ***max log size = 5000***

This option allows you to specify the maximum size the log files can grow to. Files exceeding this value (in kilobytes) are renamed with a '.old' extension. Setting this value to '0' disables the log limit.

4.2 Marshalling/Unmarshalling

If you've ever seen something similar to the following in a logfile before, you've probably been a bit perplexed:

```
[2002/12/18 14:12:05, 0]
rpc_server/srv_samr.c:api_samr_set_userinfo(670)
  api_samr_set_userinfo: Unable to unmarshall SAMR_Q_SET_USERINFO.
```

By itself, *marshalling* means to take variable data, serialize (i.e., place in order) it, and send it in transmittable form across a network or to a file. Unmarshalling is the reverse process. In DCE/RPC terminology, marshalling and unmarshalling refers to the flattening and unpacking of a data stream between a client and server. A rough approximation of this process could be described as:

- Both computers negotiate a 'set of functions' and a 'data format' that is common to both of them, so that they can communicate using the same language
- The receiving computer authenticates the caller (if required)
- Both computers negotiate the security (if required)
- The calling computer *marshalls* (flattens) the data into a 'continuous data stream' and sends it to the receiving computer

- The receiving computer *unmarshalls* (unpacks) the data stream it has just received and processes the data
- The receiving computer then *marshalls* the results of processing the data and passes them back to the caller
- The calling computer *unmarshalls* the results and submits the received data to the original function call that started this process in the first place

4.3 Using tcpdump

When troubleshooting your Samba server with tcpdump, available at <http://www.tcpdump.org>, the best use is something like:

```
tcpdump -ln -vv host 192.168.1.1 | tee tcpdump.samba
```

While analyzing your data, you may come across an error message similar to:

```
WARNING: Short packet. Try increasing the snap length
```

This actually has nothing to do with Samba, however it comes up a lot. It's tcpdump itself complaining about snaplen (`-s <number>`) being shorter than at least one packet during its capture. Again, it has nothing to do with Samba; it's harmless and can be ignored.

4.4 Troubleshooting the Server

You've configured your server and can't seem to get it working. Chances are you're uncertain how to proceed. But don't panic; there are some useful things you can do to determine where the problem is. The following tests were adapted from Andrew Tridgell's Samba 2.2.0-alpha3 DIAGNOSIS.txt.

At a minimum, you should be running the latest stable version of Samba. Also, these tests assume that your smb.conf file contains the following information:

```
[global]
workgroup = MYGROUP
netbios name = MYSERVER
encrypt passwords = yes
log level = 3
log file = /usr/local/samba/var/samba.log
wins support = yes

[tmp]
path = /tmp
browseable = yes
writeable = yes
```

Test #1: smb.conf Syntax

Verify that your smb.conf configuration file exists and has valid parameters. You'd be surprised how often this occurs. Run the following command, noting whether any errors are reported:

```
testparm /usr/local/samba/lib/smb.conf
```

Test #2: Network-Level Connectivity

Try pinging your Samba server from *itself*, using both the loopback address, actual IP address and DNS name of your server. You should be running these commands from your Samba server itself. If the first test fails, getting Samba to work is the least of your worries ;-). The second test lets you know that you can get an ICMP response from your actual IP address. The third test is designed to let you know whether or not you have your DNS (or /etc/hosts file) configured correctly, which is needed for Samba to run correctly. You should also consider running these tests from both your Samba server and your Windows client.

```
ping 127.0.0.1      # ping our Loopback address
ping 192.168.1.1   # ping our server using its IP address
ping myserver.fluffygerbils.com    # ping our server using its FQDN
```

Test #3: Daemons

You can verify that `smbd` is running by examining `/usr/local/samba/var/samba.log` for something like:

```
[2002/10/31 14:54:21, 2] lib/interface.c:add_interface(81)
  added interface ip=192.168.1.1 bcast=192.168.1.255
nmask=255.255.255.0
[2002/10/31 14:54:21, 2] smbd/server.c:open_sockets(215)
  waiting for a connection
```

You can also verify that `nmbd` is running by examining `/usr/local/samba/var/log.nmbd` for something like:

```
[2002/10/31 14:57:31, 0] nmbd/nmbd.c:main(794)
  Netbios nameserver version 2.2.x started.
  Copyright Andrew Tridgell and the Samba Team 1994-2002
[2002/10/31 14:57:31, 1] lib/debug.c:debug_message(258)
  INFO: Debug class all level = 2 (pid 23684 from pid 23684)
[2002/10/31 14:57:31, 2] nmbd/nmbd.c:main(832)
  Becoming a daemon.
[2002/10/31 14:57:31, 2] lib/interface.c:add_interface(81)
  added interface ip=192.168.1.1 bcast=192.168.1.255
nmask=255.255.255.0
[2002/10/31 14:57:31, 2] nmbd/nmbd_subnetdb.c:make_subnet(192)
  making subnet name:192.168.1.1 Broadcast address:192.168.1.255
Subnet mask:255.255.255.0
[2002/10/31 14:57:31, 2] nmbd/nmbd_subnetdb.c:make_subnet(192)
  making subnet name:UNICAST_SUBNET Broadcast address:0.0.0.0 Subnet
mask:0.0.0.0
[2002/10/31 14:57:31, 2] nmbd/nmbd_subnetdb.c:make_subnet(192)
  making subnet name:REMOTE_BROADCAST_SUBNET Broadcast address:0.0.0.0
Subnet mask:0.0.0.0
```

Test #4: NetBIOS Name Resolution via Direct Query

From the Samba box itself, verify that you can query the nmbd server (substituting your server's NetBIOS name for `MYSERVER`):

```
/usr/local/samba/bin/nmblookup -B MYSERVER __SAMBA__
```

Next, verify that you can resolve any client NetBIOS names. This test is also done on the Samba box itself (substituting your client's NetBIOS name for `MYCLIENT`).

```
/usr/local/samba/bin/nmblookup -B MYCLIENT '*'
```

Test #5: NetBIOS Name Resolution via Broadcast

We want to verify the same NetBIOS name resolution as before, but this time sending the request to your subnet's broadcast address:

```
/usr/local/samba/bin/nmblookup -d 2 '*'
```

Test #6: Application–Level Connectivity

From the Samba box itself, try generating a list of valid shares (substituting your server's NetBIOS name for `MYSERVER`):

```
/usr/local/samba/bin/smbclient -L MYSERVER
```

Next, try doing the same thing from the Windows client:

```
NET VIEW \\MYSERVER
```

Test #7: Ability to Log In

From the Samba box, try connecting to the shares using the command line utility *smbclient* (substituting your server's NetBIOS name for `MYSERVER` and your smb username for `myname`):

```
/usr/local/samba/bin/smbclient //MYSERVER/tmp -Umyname
```

Next, try doing the same thing from the Windows client:

```
NET USE X: \\MYSERVER\tmp
```

Test #8: Master Browser

Verify that you can query the Domain Master Browser for the Samba server's IP address (substituting the workgroup name that your Samba server belongs to for `MYGROUP`):

```
/usr/local/samba/bin/nmblookup -M MYGROUP
```

Test #9: Network Neighborhood

The final test is to establish whether or not the Samba server can be seen via your Window Client's Network Neighborhood. Double-clicking of the mouse is all that is necessary here.

Test #10: Still having problems?

If you've tried to follow these steps and have addressed any issues as a result, you can always consider:

- Is there a firewall running on the Samba server? Note that this happens more often than not! ;-)
- Did you use SWAT, linuxconf, or some other GUI-based control panel to configure your smb.conf file? Try starting over from scratch using your favorite text editor first before you get fancy. Trust me, it's simpler, less can go wrong, and forces you to dive in and get your hands dirty ;-)
- Are you using the default smb.conf file that came with the source? Use this only as a template and create a *new*, *blank* smb.conf file with only the minimum statements you need. I.e., if you end up sending your smb.conf file to someone for assistance, it looks bad if 90% of it is commented out from the original sample template, but then again, that's just me talking...
- Did you turn on debugging to at least log level 3 and review the results?
- Sniffing the network using tcpdump, ethereal, or other sniffer program

Again, don't panic. The solution is there; you just have to do the work to determine the cause of the problem. If you're certain you've exhausted your options and have read, followed and addressed the above tests, you can always post your question to one of the Samba mailing lists located at <http://lists.samba.org>. If you haven't addressed and resolved the above tests, don't be suprised if nobody answers your questions ;-)

4.5 Troubleshooting the Client

This document is mainly about server-specific issues, but here are a few gotcha's that can happen to a Windows client:

- Is your Windows client on the same subnet as your Samba server?
- Does the WINS server your Windows client is using know about your Samba server? Better yet, is your Windows client configured to use the Samba server as its WINS server, and is the Samba server configured to run the WINS service via nmbd?
- Are you trying to see the shares via Network Neighborhood? This list is updated only once every fifteen minutes, and is not accurate. Use NET VIEW instead

Windows 95

Early versions of Windows 95 don't use encrypted passwords. Hopefully, this isn't your case. Your only choice is to tell the Samba server not to use encrypted passwords. Windows 95 OSR2 doesn't fall into this category. If you really want to enable plain-text passwords, you can make the following change to Windows 95's registry:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001
```

Windows XP Home Edition

Unfortunately, the most common problem with Windows XP Home Edition is *using* Windows XP Home Edition itself. There are several limits to its design, most noteworthy being the inability to join or access a domain-based network.

Windows XP Professional

When attempting to join a domain, you receive the following error message: "*Computer Name Changes: The following error occurred attempting to join the domain MYDOMAIN: The specified network password is not correct*". Additionally, your Samba logfile (at debug level 1) reveals: "*smbd/service.c:make_connection(): Can't become connected user!*". This is usually caused by improper registry settings in the client. Use Window's Group Policy Editor (gpedit.msc) to make the following changes in the "Local Computer Policy\Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options" branch:

```
Disable: Domain member: Digitally encrypt or sign secure channel data
Disable: Domain member: Digitally sign secure channel data (when
possible)
```

System error 6118 has occurred. The list of servers for this workgroup is not currently available.

This behavior can occur if you have enabled Windows XP's built-in Internet Connection Firewall (ICF), which disables SMB ports by default. The Master Browser attempts to reconnect to the client computer to send the Browse list, but the firewall prevents this connection. See Microsoft Knowledge Base Article 298804.

System error 53 has occurred. The network path was not found.

You have disabled NetBIOS over TCP/IP (NetBT), even if you are only using direct-hosted SMB over TCP/IP. This problem is resolved in Windows XP Service Pack 1. See Microsoft Knowledge Base Article 308246.

The Messenger service terminated with service-specific error 2270.

This behavior can occur if the computer's name is not unique on the network. In practice, it can also occur if your computer name, username, and domain/workgroup name are not all set to unique (different) values. See Microsoft Knowledge Base Article 314094.

4.6 Troubleshooting Example

This is, perhaps, the best example of a real-life approach towards troubleshooting, which was posted on samba-technical:

```
From: jht@samba.org (John H Terpstra)
Subject: Re: Eureka! Slow Samba transfers resolved!
Date: Sun, 19 Jan 2003 21:17:39 +0000 (UTC)
```

On Sun, 19 Jan 2003, Jean-Marie White wrote:

```
> I just spent so much time troubleshooting why
> *some types* of samba transfers were slow that I thought
> I would share how I resolved the problem.
>
> This problem had been going on for ever. As far as I could
> tell, it was only happening with winamp (MP3 player). All
```

> my MP3s reside on my Linux box and I was trying to access
> them from my PC running Windows 2000. What was throwing me
> for a loop was that Windows Media player seemed to be more
> successful at playing them ! What would happen is that Winamp
> would freeze and nothing would happen. I had then decided to
> just copy my MP3s back to my PC.
>
> Then, last night I started using FrontPage for some reason or
> another; and when I tried to publish my web site to my Linux
> box, then behavior I had observed with my MP3s started
> plaguing FrontPage as well.
>
> I went through all the troubleshooting guides, learned about
> every samba configuration options: I was able to successfully
> transfer a 100M file in either direction (linux->pc or
> pc->linux) using the File Explorer without any problem.
> But I just couldn't for the life of me understand why
> winamp and FrontPage were having problems.
>
> Then I decided to use ethereal and try and understand what
> was actually happening on the wire. I realized that when
> winamp or FrontPage were trying to transfer files, Linux
> was ending up retransmitting the same frame over and over
> without ever receiving a TCP acknowledgment for it. Then the
> PC would eventually tear down the session and re-establish the
> session and restart the transfer. An article that I read
> <http://www.dd.iij4u.or.jp/~okuyamak/Documents/tuning.english.html>
> talking about how Windows could mysteriously drop packets,
> and I started thinking that maybe packets were being lost
> due to some hardware problem.
>
> So I looked at my PC network card configuration (Linksys LNE
> 100TX Fast Ethernet) and saw that the "Connection Type" was
> set to auto-detect. (My PC is connected to my Linux box through
> a Linksys 8-port Workgroup Switch - Ehterfast 10/100) So I
> thought, maybe having to ends that "auto-detect" each
> others speed is not good. So I changed my "Connection Type"
> to 100 TX Full Duplex and VOILA! All my problems solved, winamp
> is happily playing away and FrontPage can publish my whole web
> site in seconds.

The problem and solution you describe here is very common. It also happens to be the least expected, thus we have had many queries on this list regarding samba problems that were in fact hardware issues like yours.

Thank you for sharing your experience. This is valuable feedback.

John H Terpstra
Email: jht@samba.org

5. Optimizing Samba Performance

"Interfere? Of course we should interfere! Always do what you're best at, that's what I say." -- Doctor Who

5.1 Oplocks

Opportunistic locking essentially means that the client is allowed to download and *cache* the file on their hard drive while making changes; if a second client wants to access the file, the first client receives a break and must sync the file back to the server. This can give significant performance gains in some cases; in others, some programs insist on syncing back the contents of the entire file for a single change.

Level1 Oplocks (aka just plain "oplocks") is another term for opportunistic locking.

Level2 Oplocks is a fancy way of saying that you are providing opportunistic locking for a file that will be treated as "read-only". Typically this is used on files that are read-only or on files that the client has no intention to write to (at least, not initially).

Kernel Oplocks are essentially a method that allows the Linux kernel to co-exist with Samba's oplocked files, although this is simplifying things a bit. SGI IRIX and Linux are the only two UNIXes that are oplock aware at the moment.

Unless your system supports kernel oplocks, you should disable oplocks if you are accessing the same files from both Unix/Linux and SMB clients. Regardless, oplocks should always be disabled if you are sharing a database file (e.g., Microsoft Access) between multiple clients, as any break the first client receives will result in the *entire* file needing to be sync'd (not just the single record), which will result in a noticeable performance delay and, more likely, problems accessing the database in the first place. Notably, Microsoft Outlook's personal folders (*.pst) react very badly to oplocks. If in doubt, disable oplocks and tune your system from that point.

If client-side caching is desirable and reliable on your network, you will benefit from turning on oplocks. If your network is slow and/or unreliable, or you are sharing your files among other file sharing mechanisms (e.g., NFS) or across a WAN, or multiple people will be accessing the same files frequently, you probably will not benefit from the overhead of your client sending oplock breaks and will instead want to disable oplocks for the share.

Another factor to consider is the perceived performance of file access. If oplocks provide no measurable speed benefit on your network, it might not be worth the hassle of dealing with them.

You can disable oplocks on a per-share basis with the following:

```
oplocks = False
level2oplocks = False
```

Alternately, you could disable oplocks on a per-file basis within the share:

```
veto oplock files = /*.mdb/*.MDB/
```

If you're having problems with oplocks as evidenced by Samba's log entries, you may want to play it safe and disable oplocks and level2oplocks.

5.2 Socket Options

Socket options allow you to control the networking layer of your connection with your SMB clients, which you can tune to optimal performance for your local network. Each local network is different, so there is no "hard and fast" rule for this, so you'll have to do some experimenting. Reading up on socket options in general might not be a bad idea as well (try *man setsockopt*).

A good way to determine what's best for you is to create both a single 100MB dummy file and 100 multiple 1MB dummy files and test the time involved in sending/receiving both the large file and small files to/from the server (not at the same time, of course). You'll want to make sure to stop and restart Samba each time, to prevent caching. By charting out different response times for `SO_RCVBUF` and `SO_SNDBUF` you can find the optimal value for your specific network. To create a single 100MB dummy test file and 100 multiple 1MB dummy files, you could simply use:

```
#!/bin/sh

# Create a single 100MB test file
dd if=/dev/zero of=testfile count=10240 bs=10240

# Create 100 multiple 1MB dummy files
count=1
until [ "$count" -gt 100 ]
do
    let "count += 1"
    dd if=/dev/zero of=testfile${count} count=1024 bs=1024
done
```

It is not necessary to make `SO_RCVBUF` an exact multiple of MTU or MSS to avoid performance degradation due to packet fragmentation, as is commonly claimed. This is just another urban myth ;-). With that said, you'll find the best performance gain by modifying the value for `SO_RCVBUF` to your particular network setting. For the curious, MTU is the size of the entire packet. MSS is the size of the payload. The header of a TCP packet is normally 40 bytes if there aren't any TCP options expanding the header.

Regardless, a good general starting point for you to try is:

```
socket options = TCP_NODELAY IPTOS_LOWDELAY SO_KEEPAIVE
SO_RCVBUF=8192 SO_SNDBUF=8192
```

After you've gathered your data, you'll want to chart it to find the optimal values for your specific setting.

It's worth noting that if you're filtering inbound traffic with a packet filter (such as iptables), you'll cause yourself quite a bit of a headache if you've been playing with Samba's socket options. Specifically, ICMP type 3 code 4 (Fragmentation Needed) should *not* be blocked. Otherwise, your server/client will be unable to negotiate a valid MTU window size. A good rule of thumb is to avoid blocking ICMP traffic while tuning your Samba server.

5.3 Additional Tuning Options

Log Level is a useful parameter for debugging, but when it is set higher than 2 you'll no doubt suffer a large penalty in performance. This is mainly due to the amount of information being logged, along with the sync command sent after each log file operation.

Wide Links tends to cause a performance penalty when turned off.

Read Raw and *Write Raw* tend to help or hinder depending on the client. Experimentation with these options will possibly prove enlightening.

Regardless of the tuning options you use, it's worth nothing that one of the best means to measure and optimize Samba is to implement testing over a period of weeks or months to isolate trends. This reflects the real usage of the network and is almost impossible to beat in the information it reflects. Discovering the maximum amount of work that a server can perform is invaluable in combination with actual test results to help in determining what tuning options are actually beneficial to a system.

6. Samba PDC

"I tried to get some documentation out of Digital on this, but as far as I can tell even they don't have it ;-)" -- Linus Torvalds, in an article on a dnsver

6.1 Configuration

To run as a Primary Domain controller (PDC), you will need to add the following entries to your smb.conf file:

```
[global]
  domain logons = yes
  logon drive = p:
  logon home = \\BIGSERVER\%U
  os level = 99
  preferred master = yes
  security = user
  wins support = yes

[homes]
  read only = no
  create mode = 0600
  directory mode = 0700

[netlogon]
  path = /home/netlogon
```

6.2 Users and Groups

To effectively manage your Samba/Windows collection of computers, it is *critical* that you understand how Windows NT/2000/2003 Servers provide security through the use of *Users* and *Groups*. Specifically, you must be aware that there is a difference between the following four different types of Users and Groups:

- 6.2.1 *Local Users*
- 6.2.2 *Local Groups*
- 6.2.3 *Domain Users*

- 6.2.4 *Domain Groups*

6.2.1 Local Users

Local Users are accounts that are unique to a single workstation. Each Windows 2000/XP workstation has two built-in Users: *Administrator* and *Guest*. Local Users are not available on Domain Controllers.

6.2.2 Local Groups

Local Groups are groups that are unique to a single workstation. Each Windows 2000/XP workstation has several built-in Groups: *Administrators* have complete and unrestricted access to the workstation; *Power Users* have most administrative powers with some restrictions and can run legacy applications in addition to certified applications; *Users* are prevented from making accidental or intentional system-wide changes, and can run certified applications but not most legacy applications; the *Guests* Local Group actually has the same access permissions as the Users group (something that is often overlooked). Local Groups are not available on Domain Controllers.

6.2.3 Domain Users

Domain Users are accounts that are created on a Windows NT/2000/2003 Server that is acting as a Primary Domain Controller. Samba mimics this behavior by treating every user in smbpasswd(8) as a Domain User. If the user *root* is defined in smbpasswd, it is equivalent to the Administrator account on Windows NT/2000/2003.

6.2.4 Domain Groups

Domain Groups are groups that are created on a Windows NT/2000/2003 Server that is acting as a Primary Domain Controller. Samba technically only has two Domain Groups: the *root* user, and everyone else. However, Samba mimics additional group behavior by honoring filesystem permission restrictions using the group membership information in /etc/groups — see groups(1) for more information.

6.2.5 Local Users/Groups vs. Domain Users/Groups

Samba has no knowledge of Windows 2000/XP Local Accounts; as far as it is concerned, there are only multiple users (defined in smbpasswd) and a single Administrator named *root* (if defined in smbpasswd). Samba also has no knowledge (nor does it care) about Windows 2000/XP Local Groups; all Samba group memberships are defined in /etc/groups.

The reverse is also true — Windows 2000/XP has no direct knowledge of Samba Domain Users or the /etc/groups file on the Samba box. Therefore, it is possible to log into the domain as root (not advised for the security conscious) yet not have any administrative authority over the Windows 2000/XP box!

Under Windows NT/2000/2003 Servers, you can add Local Users, Global Users, and Global Groups to Local Groups. With Samba, you can add Local Users and Global Users to Local Groups. However, with both Windows NT/2000/2003 Servers and Samba Servers, you cannot add Local Users and Local Groups to Global Groups.

One option to get around this is to select one account from Samba to add it to each Windows 2000/XP workstation's Administrators Local Group. This needs to be done on a per-workstation basis since Local Groups are just that — local to the box itself, and to no one else. There is no way to do this from the Samba server (nor should there be) or even from a Windows 2000 Server.

Regarding setting up security and the Group Policy Editor, see Microsoft Knowledge Base Article 307882.

Microsoft's description of Local/Domain Users/Groups is available in their helpfile [%SYSTEMROOT%\Help\LOCALSEC.CHM](#).

6.3 Joining your Samba Domain

6.3.1 Windows 95/98/ME

It's fairly easy to join a domain with Windows 95/98/ME. You can join the domain by adjusting the properties to "Client for Microsoft Networks" in the Network Control Panel. This is not a big area of interest for most, so it's probably the shortest section ;-) Please note that Windows 95 and 95A don't use encrypted passwords, so this option must be disabled in your smb.conf to support these clients (and your non-Win95 clients will need a patch then; see /docs/Registry/ in the Samba source code).

6.3.2 Windows 2000 Professional

When establishing your machine as a member of a domain, you'll need to add each Windows machine into your Linux machine's password file using vipw(8) or an equivalent. Regardless, the entry in /etc/passwd should look something like this (below). Note the dollar sign (\$) at the end of the workstation name; if your workstation's name is PC-LAB15, your entry in /etc/passwd should begin with PC-LAB15\$ — this is commonly overlooked. For information on the fields in /etc/passwd itself, try ``man 5 passwd``.

```
pc-lab15$:x:1200:300:Workstation:/dev/null:/bin/false
```

You should also have a group account specific to workstation-only accounts, using vigr(8) or an equivalent:

```
workstation::300:
```

And your entry in /etc/shadow (hint: try ``vipw -s``) should look like:

```
pc-lab15$:*:9797:0:0:0:0:
```

Finally, you need to add this account to the Samba password file. Note the *lack* of a dollar-sign here! The `-m` switch tells Samba that this is a machine account, and the `-a` switch tells it to add this account to the database. Note that this account won't have an initial password, so it's important to run this step immediately prior to having the workstation join your domain.

```
smbpasswd -m -a pc-lab15
```

To allow anyone to actually join, however, you'll need to add a smbpasswd entry for the user "root" — this is required!

Log in with an account that is a member of the Local Administrator's group. Note that this account should not exist on your Samba box.

Next, drop to a MS-DOS prompt and delete any existing NetBIOS connections. This will ensure that no ghosted connections to NETLOGON or \$IPC exist:

```
NET USE * /DELETE /YES
```

In your System Properties, locate the Computer Name section, and click Change. From here, you can join your Samba domain, which is the "Workgroup" parameter from your smb.conf file. You'll be prompted for a name and password of an account with permissions to join the domain; only the user account "root" will work here, and the account must exist both on your Linux box (of course) and in your smbpasswd file.

Possible error messages you might encounter when trying to join the domain are:

- ***The following error occurred attempting to join the domain "MYDOMAIN": The network path was not found.*** Ironically, this is usually a Windows XP specific error where it apparently attempts to contact your PDC via a stale (and incorrect) gateway/route entry in the registry. Change your IP address, subnet, and gateway to something different, and then change it back again. This usually removes any stale entries. It's unclear why this happens, but this behavior can be verified by a packet sniffer.
- ***The following error occurred attempting to join the domain "MYDOMAIN": No mapping between account names and security IDs was done.*** This obscure error has reportedly been fixed by using lower-case names for the workstation name in /etc/passwd and smbpasswd and on the Windows XP client. It's unclear why this matters as Samba shouldn't care about this.
- ***The following error occurred attempting to join the domain "MYDOMAIN": Access is denied.*** There isn't a machine account entered in smbpasswd for the computer you're attempting to have join the domain, or the machine account is currently disabled. It's also possible that you're trying to join the domain using an account name other than "root", which is required.
- ***The following error occurred attempting to join the domain "MYDOMAIN": Logon failure: unknown user name or bad password.*** Either "root" doesn't exist in the smbpasswd database, or you've typed in an incorrect password.
- ***A domain controller for the domain "MYDOMAIN" could not be contacted.*** You've typed in a domain name, however it's not the one that your Samba server is using. You want to use the value of the "workgroup" parameter from smb.conf as the name of your domain. Another possibility is that nmbd is not running on your server, and therefore can't answer NetBIOS name queries.
- ***The Account Used is a Computer Account. Use Your Global User Account or Local User Account to Access the Server.*** This reportedly happens if the machine account information (account, UID, GID) in /etc/passwd does not match its same credentials in /etc/samba/private/smbpasswd. Usually, this is the result of modifying either /etc/passwd or smbpasswd for the machine account, and not sync'ing the changes in both. It can also be caused by using an account name that does not match the NetBIOS name of the Windows machine (thanks to Sherwood Botsford for the info!).

6.3.3 Windows XP Home

Due to limitations built into Windows XP Home by Microsoft, you won't be able to join any domain -- Samba or Windows.

6.3.4 Windows XP Professional

To allow Windows XP Professional to join a Samba Domain, you will need to first make the following changes to your registry and reboot:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters]
"requiresignorseal"=dword:00000000
"signsecurechannel"=dword:00000000
```

Next, to join your Samba domain, follow the instructions (above) for Windows 2000 Professional.

If you find that Notepad starts with "[.ShellClassInfo] LocalizedResourceName=@%SystemRoot%\System32\Shell32.dll,-21787" when you log into the Samba domain, please see [Microsoft Knowledge Base Article 330132](#). This is a non-Samba-related issue, although it comes up frequently.

6.4 Roaming Profiles

If you'd like to enable roaming profiles for Windows 2000/XP, make the following changes to your smb.conf file.

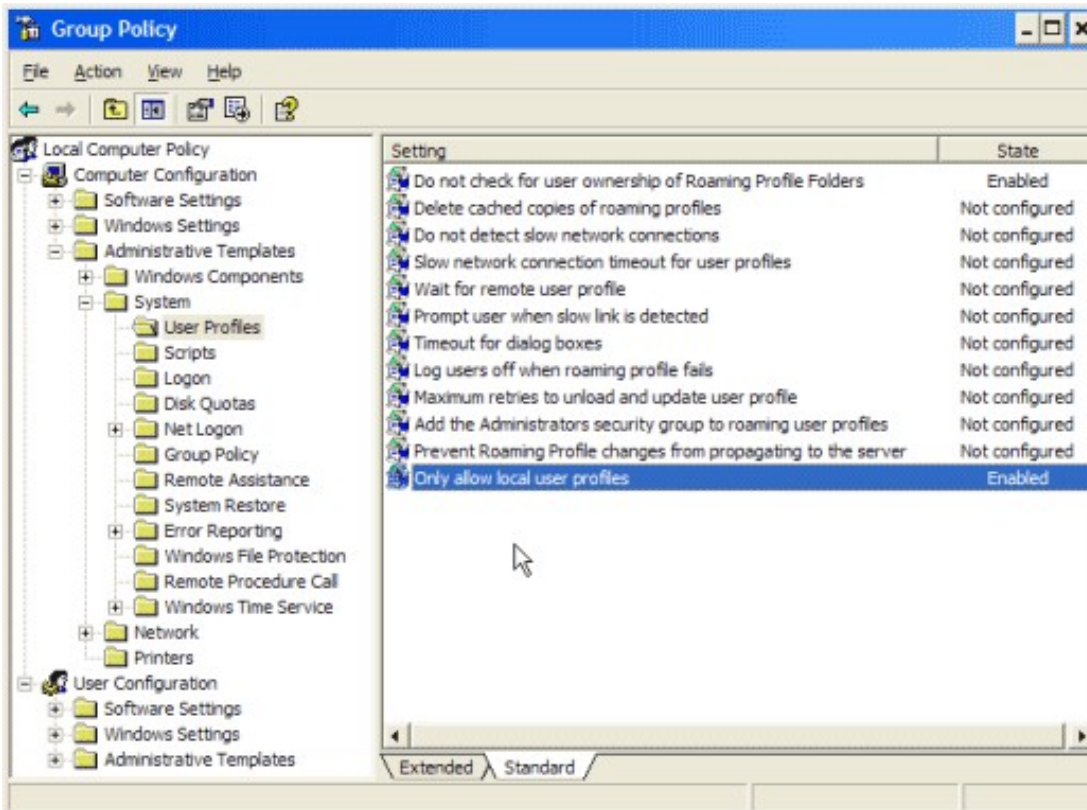
```
[global]
logon path = \\MYSERVER\profile\%U

[profile]
create mode = 0600
csc policy = disable
directory mode = 0700
path = /home/profile
profile acls = yes
read only = no
```

6.4.1 Roaming Profiles and Windows XP

Due to some changes introduced with Windows XP Service Pack 1/1a, you must choose one of the following methods regarding how you intend to implement Roaming Policies:

1. Use the statement "profile acls = yes" in your smb.conf file (see above). If you want to enable roaming profile support under Windows XP and you have Samba 2.2.6 or later installed, this is the recommended option.
2. Use the Group Policy Editor (gpedit.msc) and enable "Computer Configuration\Administrative Templates\System\User Profiles\Do not check for user ownership of Roaming Profile Folders".
3. Use the Registry Editor and edit the HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System\CompatibleRUPSecurity to have the DWORD value of 1
4. If you would like to *disable* roaming profiles under Windows XP --- and avoid most of the headaches it (usually) causes --- use the Group Policy Editor (gpedit.msc) and enable "Computer Configuration\Administrative Templates\System\User Profiles\Only allow local user profiles". This is my personal favorite.



6.5 Login Scripts

If you're running your Samba server to act as a Primary Domain Controller (PDC) for Windows clients, you can have a batch file execute upon login. This can be enabled by adding the following to your smb.conf file:

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

[netlogon]
path = /home/netlogon
```

All login scripts must reside in the [netlogon] share and should be formatted to be readable by your Windows clients (see *Line Feeds CR/LF* below). Here's a sample script which I've found to be quite useful:

```
@echo off
REM Synchronize the client's clock with the server.
REM Requires Power User rights, however
NET TIME \\MYSERVER /SET /YES

REM Map some drives
NET USE F: \\MYSERVER\PUBLIC /YES
NET USE I: \\MYSERVER\INSTALL /YES

REM Map the user's home directory
NET USE P: /HOME /YES
```

```

REM Run Symantec Liveupdate (silently) for Antivirus definitions.
REM Does not require the user to have local administrator rights ;-)
IF EXIST "C:\Program Files\Symantec\LiveUpdate\Luall.exe" "C:\Program
Files\Symantec\LiveUpdate\Luall.exe" -s

REM Run WinNT/2K/XP/2003 specific stuff
IF NOT "%OS%"=="Windows_NT" goto noxp

IF NOT EXIST "\\SERVER%\%USERNAME%\xpbackup" mkdir
"\\SERVER%\%USERNAME%\xpbackup"
REM /c ignore errors, /h copy hidden/system files, /r copy read-only
files, /y no prompt, /d only copy newer files
XCOPY "%HOMEDRIVE%\%HOMEPATH%\*.*" "\\MYSERVER%\%USERNAME%\xpbackup\*.*
/S /E /C /H /R /Y /D
IF ERRORLEVEL 0 GOTO services
ECHO There was an error backing up your files

:services
REM Adjust some services
NET STOP "Protected Storage"
NET START "Automatic Updates"

REM Back up the client's registry
REM Why anyone would want to do this is beyond me
REGEDIT /E P:\Registry.bak

REM Update (silently) the client's registry
REM Depending on what you import, you need the appropriate user rights
REGEDIT /S "\\SERVERNAME\NETLOGON\winxp.reg"

:noxp
echo Done!

```

6.6 Linefeeds (CR/LF)

Another issue to become aware of with login scripts is *linefeeds*. If you're working in a Linux/Windows mixed environment, you're bound to come across files that are littered with '^M' at the end of every line. This is the Windows command for 'newline', which uses a combination of CR/LF, whereas Linux uses just the single LF. To convert a text file from DOS to Linux, thereby removing the '^M' newline characters in the file, use the *fromdos* command:

```
fromdos <dosfile >linuxfile
```

To convert a text file from Linux to DOS, thus adding the '^M' newline characters so that the file can be read by your Windows clients, use *todos*:

```
todos <linuxfile >dosfile
```

Alternately, if you find you don't have the *fromdos/todos* utilities, you can try this simple trick:

```
cat dosfile | col -b > linuxfile
```

It's worth noting that if you're running into this problem in the first place, it means you're working on a file that is being edited in both your Linux and Windows environments. As such, you should probably read the section on *Oplocks* to prevent any data loss...

If you'd rather take care of things from the Windows side of the coin, you can simply use a text editor that can convert between DOS/Linux. My personal favorite is Jean-Pierre Menicucci's Editeur which includes syntax highlighting, although you're encouraged to look around for what fits your needs best.

7. Security and Samba

"It turned out that the worm exploited three or four different holes in the system. From this, and the fact that we were able to capture and examine some of the source code, we realized that we were dealing with someone very sharp, probably not someone here on campus." -- Dr. Richard LeBlanc, associate professor of ICS, in George Tech's campus newspaper after the Internet worm.

Regardless of the security measures you incorporate into your Samba design, you should never run SMB services across an untrusted network (e.g., the Internet). The SMB/CIFS protocol is vulnerable to many attacks. Ports 137/udp, 138/udp, 139/tcp and 445/tcp should never be exposed to untrusted networks.

7.1 Restricting Client Access

If you run your samba server on a machine that has a valid IP address to the Internet, or an an untrusted LAN, you'll probably want to limit who can connect to your Samba shares. Assuming your server runs on 192.168.0.1, your netmask is 255.255.255.0 and you wish to deny access to a host in your network on 192.168.0.8, your smb.conf file should look like:

```
hosts allow = 192.168.0.0/255.255.255.0 127.0.0.1 EXCEPT 192.168.0.8
hosts deny = ALL
```

It's also useful to limit the interfaces on which Samba will run, if you have a multihomed (more than one IP address) server. A common mistake is to set the interfaces line to the specific IP address of the box, when it is actually the IP *subnet* that your interface is on that you want to use. Assuming your server runs on 192.168.0.1 and your netmask is 255.255.255.0, your smb.conf file should look like:

```
interfaces = 192.168.0.0/255.255.255.0 127.0.0.1
bind interfaces only = Yes
```

7.2 Shadows and Light

These are simple smb.conf configuration settings that make Samba a little more transparent to the watching eye.

```
# Encrypt all passwords stored in /usr/local/samba/private/smbpasswd
encrypt passwords = yes

# Prevent the administrative user from logging in
invalid users = root @wheel
```

```
# Files that have their Linux permissions set to prevent access
shouldn't even appear
hide unreadable = yes

# Prevent browsing by default
browseable = no

# Don't allow access to any of the following files.
# Useful for preventing the spread of virus infections on your server
# should a Windows-connected client become infected.
# The last match bit prevents accessing files with a CLSID in its file
extension
veto files = /*.exe/*.dll/*.com/*.vbs/*.{*}/
# You could uncomment the next line to use a more restrictive set of
# unallowed file extensions (see Microsoft KB 290497)
# veto files =
/*.scr/*.scf/*.sct/*.pif/*.exe/*.bas/*.bat/*.com/*.ade/*.adp/*.asp/*.asx/*.chm/*.c

# Hide the following files; the client can still choose to alter their
view settings to show hidden files.
hide files = /example.txt/*.ico/
```

7.3 Samba and firewalls (iptables)

NBT (NetBIOS-over-TCP): This method of communication relies on three specific services: NetBIOS Name Service (port 137/udp), NetBIOS Datagram Service (port 138/udp), and NetBIOS Session Service (port 139/tcp). If you're using Windows 95/98/ME or Windows NT, this is the only method available to you.

Direct Hosted: This is the new Microsoft marketing term used to describe a new method of file/printer sharing which uses only port 445/tcp. Currently, only Windows 2000 and Windows XP support this new method. It is "NetBIOS-less" in that all name resolution is done via DNS (which removes the dependence on WINS). By default, Windows 2000/XP will attempt to use both NBT and Direct Hosted; whichever method receives a response first is the one that will be used. With Direct Hosted traffic, a four-byte header precedes all SMB traffic. The first byte of this header is always 0x00, and the next three bytes are the length of the remaining data.

Samba does have the ability to accept connections on port 445, but it does not listen on this port when started with the `-D` option. Instead, it must be started from `inetd` (without the `-D` flag) and `inetd` must be configured to accept smb connections on port 445. Please be aware that this is a very new and untested feature, so your mileage may vary.

Assuming your server has an IP address of 192.168.0.1, a netmask of 255.255.255.0 (aka "/24"), and is using the Traditional NetBIOS method, your iptables bit might look like:

```
iptables -A INPUT -p udp -s 192.168.0.0/24 --sport 137 -d
192.168.0.1/32 --dport 137 -j ACCEPT
iptables -A INPUT -p udp -s 192.168.0.0/24 --sport 1024:65535 -d
192.168.0.1/32 --dport 137 -j ACCEPT
iptables -A INPUT -p udp -s 192.168.0.0/24 --sport 138 -d
192.168.0.1/32 --dport 138 -j ACCEPT
iptables -A INPUT -p udp -s 192.168.0.0/24 --sport 1024:65535 -d
192.168.0.1/32 --dport 138 -j ACCEPT
```

```
iptables -A INPUT -p tcp -s 192.168.0.0/24 --sport 1024:65535 -d
192.168.0.1/32 --dport 139 -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --dport 1024:65535 -m state --state RELATED
-j ACCEPT
iptables -A OUTPUT -p udp --dport 1024:65535 -m state --state RELATED
-j ACCEPT
[... insert the rest of your rules here ...]
iptables -A INPUT -j DROP
iptables -A OUTPUT -j DROP
```

As a sidenote, if you're running a 2.4.x Linux kernel and you're still using ipchains(8), you should consider upgrading to iptables which is considered a *stateful* firewall and a lot easier to configure. If you have an old ipchains or ipfwadm based firewall, you should be able to apply the same logical layout of the above script and adapt it to your needs.

7.4 Tunneling SMB through SSH

One method of encrypting SMB traffic over a network is to "tunnel" SMB through SSH using a method known as *port forwarding*. This is a frequently asked question by system administrators wishing to secure remote SMB traffic. While this is possible, it does have some serious drawbacks which we will touch on as well.

It's important to be aware that running SMB by itself without SSH over a 56k dialup line is still *terribly* slow to the point of frustration. If you don't have a high speed link or at least a lot of patience, you probably don't even want to deal with tunneling over SSH.

The other unfortunate bit of news is that due to a design limitation in the GUI API of Windows 9x/ME, you'll only be able to perform your tunneled work in a MS-DOS window. Once you step outside of this and attempt to interact with your remote server via the GUI, you'll find 30-60 second periods where the computer will pause/hang, after which it will complain that the path is invalid or unavailable. One possible explanation is the 16/32-bit nature of this type of Windows OS, however there has yet to be a confirmation of this by either Microsoft or the Samba team. Those using the 32-bit Windows 2000/XP systems will not have this limitation whatsoever.

That being said, the good news is that tunneling SMB through SSH is indeed possible. Name services, or anything relying on UDP, can't be forwarded via SSH due to a limitation in how SSH forwards ports (TCP only). So, we'll focus on port forwarding only TCP/port 139. Since UDP tunneling is not available under SSH, your first step involves adjusting the lack of WINS/broadcast name resolution.

Windows provides two different files, HOSTS and LMHOSTS. The former is for Hostname-to-IP Address resolution (similar to DNS), and the latter is for NetBIOS-name-to-IP Address resolution (similar to WINS). LMHOSTS originally stood for "LAN Manager". These files are provided as a "backup" in the case that DNS or WINS are not available. Since NetBIOS name resolution only works via UDP, which can't be tunnelled via SSH, the first step is to edit the LMHOSTS file:

```
REM Under WinNT/2k/XP, this is c:\windows\system32\drivers\etc\LMHOSTS
REM Under Win9x/ME, this is c:\windows\LMHOSTS

127.0.0.1      FAKENAME      #PRE
```

Where FAKENAME is a bogus NetBIOS name that you will use to refer to your Samba server. The #PRE statement tells Windows that this name should be cached into memory, otherwise it won't always be read. The LMHOSTS file will not

be processed by Windows until you reboot or you issue the following command, which forces a reload of the NetBIOS name cache (note the uppercase-R):

```
nbtstat -R
```

Configure your client's SSH program to forward port 139/tcp on the localhost to port 139/tcp on the server, and then connect via SSH. Once done, open up a MS-DOS window and issue these commands:

```
NET VIEW \\127.0.0.1
NET VIEW \\FAKENAME
```

Viola! Both commands work, and you can confirm the encryption with a packet filter.

7.5 Samba and SSL

For better or worse, SSL support is being deprecated in Samba 3.0. This is mainly due to the lack of implementation of Windows support for SMB/SSL. While the code still exists in the 2.2.x branch, it hasn't been updated for some time. Those interested in the specifics can review the following posting to samba-technical:

```
From: vorlon@netexpress.net (Steve Langasek)
Subject: Re: Impending Removal of --with-ssl
Date: Fri, 3 May 2002 15:51:55 +0000 (UTC)
```

On Fri, May 03, 2002 at 07:56:43AM -0700, abartlet@samba.org wrote:

```
> This message is a warning:
> --with-ssl will die.
> Ok, thats enough with the dramatics, but the general consensus
> amongst the samba team is that --with-ssl really isn't a
> particulary smart idea, and it is better implmented by
> external tools. So what is --with-ssl exactly? And why
> kill it? --with-ssl allows Samba to tunnel SMB inside an
> SSL connection. Unfortunately there are only 2 clients:
> smbclient and sharity. Windows clients simply don't know how
> to use SSL. So why kill it? It might be useful to somebody?
> While some small minority of users might find it handy, it
> confuses many more including a supprising number of our
> distributors. Users actually using this functionality will
> find that they can achive almost the same effect by creative
> use of 'stunnel' both as an inetd wrapper as as a 'LIBSMB_PROG'
> program. Finally, it is intrusive and ugly, with large #ifdef
> sections in what should be simple code. If somebody can come up
> with both reasons to keep this code, and time to maintain it,
> then I would like to hear it.
```

Though I don't object to --with-ssl's presence if someone is willing to maintain it, there are a variety of reasons why Debian has never enabled this option, and probably never will. Having gotten past the obstacle of US export law, it's now been pointed out[1] that the GPL does not permit us as a distributor to ship GPLed binaries linked against OpenSSL together with the OpenSSL libraries themselves; unless

all copyright holders in Samba are willing to grant an explicit exemption for linking with OpenSSL, Debian is not willing to expose itself or its mirror operators to the legal risk.

Assuming everyone was ok with the legal minutiae, we would still have to decide if SSL was really worth enabling. I've always been lukewarm about this option, because setting up an SSL tunnel on the Unix side has always been the /easy/ part: it's configuring all of your Windows clients (of varying flavors) to use SSL for SMB connections that takes doing. So the savings of having SSL support compiled into Samba are minimal, but the potential headaches are numerous. I'd almost say you'd be doing users a favor by removing this option.

Steve Langasek
postmodern programmer

[1]
<http://lists.debian.org/debian-devel/2002/debian-devel-200203/msg01569>

7.6 Additional Techniques

The following information is written by Andrew Tridgell, the leader of the Samba Team, on additional ways to protect an unpatched pre-2.2.8 Samba server. It's also a good overview of some basic Samba security techniques as well:

This is a note on how to provide your Samba server some protection against the recently discovered remote security hole if you are unable to upgrade to the fixed version immediately. Even if you do upgrade you might like to think about the suggestions in this note to provide you with additional levels of protection.

Using host based protection

In many installations of Samba the greatest threat comes for outside your immediate network. By default Samba will accept connections from any host, which means that if you run an insecure version of Samba on a host that is directly connected to the Internet you can be especially vulnerable.

One of the simplest fixes in this case is to use the 'hosts allow' and 'hosts deny' options in the Samba smb.conf configuration file to only allow access to your server from a specific range of hosts. An example might be:

```
hosts allow = 127.0.0.1 192.168.2.0/24 192.168.3.0/24
hosts deny = 0.0.0.0/0
```

The above will only allow SMB connections from 'localhost' (your own computer) and from the two private networks 192.168.2 and 192.168.3. All other connections will be refused connections as soon as the client sends its first packet. The refusal will be marked as a 'not listening on called name' error.

Using interface protection

By default Samba will accept connections on any network interface that it finds on your system. That means if you have a ISDN line or a PPP connection to the Internet then Samba will accept connections on those links. This may not be what you want.

You can change this behavior using options like the following:

```
interfaces = eth* lo
bind interfaces only = yes
```

that tells Samba to only listen for connections on interfaces with a name starting with 'eth' such as eth0, eth1, plus on the loopback interface called 'lo'. The name you will need to use depends on what OS you are using. In the above I used the common name for ethernet adapters on Linux.

If you use the above and someone tries to make a SMB connection to your host over a PPP interface called 'ppp0', they will get a TCP connection refused reply. In that case no Samba code is run at all as the operating system has been told not to pass connections from that interface to any process.

Using a firewall

Many people use a firewall to deny access to services that they don't want exposed outside their network. This can be a very good idea, although I would recommend using it in conjunction with the above methods so that you are protected even if your firewall is not active for some reason.

If you are setting up a firewall then you need to know what TCP and UDP ports to allow and block. Samba uses the following:

```
UDP/137 – used by nmbd
UDP/138 – used by nmbd
TCP/139 – used by smb
TCP/445 – used by smb
```

The last one is important as many older firewall setups may not be aware of it, given that this port was only added to the protocol in recent years.

Using a IPC\$ share deny

If the above methods are not suitable, then you could also place a more specific deny on the IPC\$ share that is used in the recently discovered security hole. This allows you to offer access to other shares while denying access to IPC\$ from potentially untrustworthy hosts.

To do that you could use:

```
[ipc$]
hosts allow = 192.168.115.0/24 127.0.0.1
hosts deny = 0.0.0.0/0
```

this would tell Samba that IPC\$ connections are not allowed from anywhere but the two listed places (localhost and a local subnet). Connections to other shares would still be allowed. As the IPC\$ share is the only share that is always accessible anonymously this provides some level of protection against attackers that do not know a username/password for your host.

If you use this method then clients will be given a 'access denied' reply when they try to access the IPC\$ share. That means that those clients will not be able to browse shares, and may also be unable to access some other resources. I don't recommend this method unless you cannot use one of the other methods listed above for some reason.

8. Appendix

"I don't have any solution but I certainly admire the problem." -- Ashleigh Brilliant

8.1 SMB Methodology

Traditionally, SMB uses UDP port 137 (NetBIOS name service, or netbios-ns), UDP port 138 (NetBIOS datagram service, or netbios-dgm), and TCP port 139 (NetBIOS session service, or netbios-ssn). Anyone looking at their network with a good packet sniffer will be amazed at the amount of traffic generated by just opening up a single file. In general, SMB sessions are established in the following order:

1. TCP Connection – establish 3-way handshake (connection) to port 139/tcp or 445/tcp.
2. NetBIOS Session Request – using the following "Calling Names": The local machine's NetBIOS name plus the 16th character 0x00; The server's NetBIOS name plus the 16th character 0x20
3. SMB Negotiate Protocol – determine the protocol dialect to use, which will be one of the following: PC Network Program 1.0 (Core) – *share level security mode only*; Microsoft Networks 1.03 (Core Plus) – *share level security mode only*; Lanman1.0 (LAN Manager 1.0) – *uses Challenge/Response Authentication*; Lanman2.1 (LAN Manager 2.1) – *uses Challenge/Response Authentication*; NT LM 0.12 (NT LM 0.12) – *uses Challenge/Response Authentication*
4. SMB Session Startup. Passwords are encrypted (or not) according to one of the following methods: Null (no encryption); Cleartext (no encryption); LM and NTLM; NTLM; NTLMv2
5. SMB Tree Connect: Connect to a share name (e.g., \\servername\share); Connect to a service type (e.g., IPC\$ named pipe)

A good way to examine this process in depth is to try out [SecurityFriday's SWB program](#), which allows you to enable the SMB(CIFS) session setup without depending on the version and the registry setting of your Windows machines.

8.2 Samba Startup Script

If you want to easily start and stop your smbd/nmbd daemons with a minimum of fuss, you can use the following script. Note that any active clients won't like you for it, especially if they're running Microsoft Access on an open networked file ;-)

```
#!/bin/sh
# Start/stop/restart samba

samba_start() {
    if [ -x /usr/sbin/smbd -a -x /usr/sbin/nmbd -a -r
/etc/samba/smb.conf ]; then
        echo -n "Starting: smbd"
        /usr/sbin/smbd -D
        echo " nmbd"
        /usr/sbin/nmbd -D
    else
```

```
        echo "$0: Cannot start Samba"
    fi
}

samba_stop() {
    echo "Stopping: smbd nmbd"
    killall smbd nmbd
}

samba_restart() {
    samba_stop
    sleep 2
    samba_start
}

case "$1" in
'start')
    samba_start
    ;;
'stop')
    samba_stop
    ;;
'restart')
    samba_restart
    ;;
*)
    echo "usage $0 start|stop|restart"
esac
```

8.3 Windows Event Manager

This technically isn't related to Samba, but it does involve some significant Windows/Linux integration and it's quite nifty (and therefore somewhat relevant). Often, it's desirable to have Windows NT/2000/XP send Event Log messages to Linux via syslogd. One such free and simple tool is *evtsys* from [Purdue University](#). Essentially, it installs itself as a Windows service and simply forwards all Event Log traffic to a designated server's syslog daemon. Here's a sample output from syslog:

```
Nov 13 15:33:23 pc-lab14 Norton AntiVirus: Download of virus
definition file failed.
Nov 13 15:35:20 pc-lab16 Service Control Manager: N/A: The Parallel
port driver service failed to start
Nov 13 15:35:41 pc-lab13 MsiInstaller: N/A: Microsoft Office 2000 --
Installation successfull.
Nov 13 15:44:43 pc-lab12 Norton AntiVirus: N/A: Symantec AntiVirus
Realtime Protection Loaded.
Nov 13 15:45:23 pc-lab14 Cdrom: N/A: The device, \Device\CdRom0, has a
bad block.
Nov 13 15:46:03 pc-lab12 Norton AntiVirus: N/A: Virus Found:
W32.Badtrans.B@mm in File: info.DOC.scr
```

8.4 Sample smb.conf

A sample smb.conf file exists in the Samba source, in the examples/ directory. If you're just looking for a configuration file to get started with, see Section 2.4.

8.5 Additional Resources

- [CIFS: Common Insecurities Fail Scrutiny](#) by Hobbit
- [SMB/CIFS by The Root](#) by ledin
- [Doing the Samba on Windows](#) by Financial Review
- [Implementing CIFS](#) by Christopher R. Hertel
- [Just what is SMB?](#) by Richard Sharpe
- [Opening Windows Everywhere](#) by Mike Warfield
- [SMB HOWTO](#) by David Wood
- [The Story of Samba](#) by Christopher R. Hertel & Luke Leighton
- [Using Samba as a PDC](#) by Andrew Bartlett

8.6 Epilogue

"What's fundamentally wrong is that nobody ever had any taste when they did it. Microsoft has been very much into making the user interface look good, but internally it's just a complete mess. And even people who program for Microsoft and who have had years of experience, just don't know how it works internally. Worse, nobody dares change it. Nobody dares to fix bugs because it's such a mess that fixing one bug might just break a hundred programs that depend on that bug. And Microsoft isn't interested in anyone fixing bugs -- they're interested in making money. They don't have anybody who takes pride in Windows 95 as an operating system.

People inside Microsoft know it's a bad operating system and they still continue obviously working on it because they want to get the next version out because they want to have all these new features to sell more copies of the system.

The problem with that is that over time, when you have this kind of approach, and because nobody understands it, because nobody REALLY fixes bugs (other than when they're really obvious), the end result is really messy. You can't trust it because under certain circumstances it just spontaneously reboots or just halts in the middle of something that shouldn't be strange. Normally it works fine and then once in a blue moon for some completely unknown reason, it's dead, and nobody knows why. Not Microsoft, not the experienced user and certainly not the completely clueless user who probably sits there shivering thinking "What did I do wrong?" when they didn't do anything wrong at all.

That's what's really irritating to me."

— Linus Torvalds, from an [interview](#) with BOOT Magazine, Sept 1998

