



Introduction to Samba, Part 3

Getting Samba to samba: The configuration stage

Daniel Robbins
President/CEO, Gentoo Technologies, Inc.
July 2000

In [his previous article](#), Daniel Robbins guided you through the process of setting up Samba for the first time. Now it's time to configure Samba so that it does everything that **you** want it to do.

Contents:

- [Adding some shares](#)
- [An exciting share](#)
- [Sharing home directories](#)
- [Share parameters](#)
- [smb.conf options](#)
- [Printing from Samba](#)
- [Resources](#)
- [About the author](#)

Here's a listing of the smb.conf that we've been working with:

```
[global]

# set YOURWORKGROUP to the name of your workgroup

workgroup = YOURWORKGROUP
security = user
encrypt passwords = yes
guest account = guest

# enable *one* of the following two lines
# enable the first line if you want to use an existing
# WINS server on your subnet, otherwise, enable the
# second.

# wins server = IP address of WINS server
# wins support = yes

local master = yes
os level = 99
domain master = yes
preferred master = yes

# optional security options.  Customize for your site.

# hosts allow = 192.168.1. 127.
# interfaces = eth1

[tmp]

path=/tmp
writeable=yes
```

Adding some shares

While this smb.conf file is functional, all it does is share the /tmp directory with Windows by creating a share with the name of "tmp". Not too exciting. Let's create another share that could be more useful. Add the following lines to your smb.conf and restart Samba.

```
[ftp]

path=/path/to/ftp/root
writeable=no
```

If you have an ftp site or some kind of file archive on your Samba server, you can use something like this to share the data over

the network. The "writeable=no" parameter tells Samba that no one should be allowed to create or modify files on this share. Anyone who has a valid Samba account set up will be able to access this share.

An exciting share

OK, I know what you're thinking. This still isn't too exciting. How about sharing a home directory? Here's how:

```
[drobbins]
    comment=Home directory for drobbins
    path = /home/drobbins
    force user = drobbins
    read only = no
    valid users = drobbins administrator
```

This one is a lot more interesting. Adding something like this to your smb.conf will allow you to share a home directory. In this example, a "drobbins" share is created. This shares the contents of /home/drobbins over the network. Fortunately, however, thanks to the "valid users" line, not just anyone can access this share. This line causes Samba to reject access by anyone other than the "drobbins" or "administrator" account. Since I'm using Windows NT, I'm often logged in as administrator. In such situations, it's nice to still be able to access the "drobbins" share. Such a valid users line allows this to happen.

You'll also notice the use of the "read only" parameter. As you might guess, "read only" is the opposite of the "writeable" parameter. We could have just as easily replaced this line with "writeable=yes". This means that Samba will permit writing to this particular share, as long as you have the proper permissions to do so. Since the Samba "drobbins" user maps directly to the Unix "drobbins" user, and drobbins happens to be the owner of the /home/drobbins directory and its contents, writing and modifying files will be permitted.

Have you ever created a file in your home directory as root, and then tried to modify it when you're logged in as a normal user only to be denied write access? This seems to happen all the time to me. To fix the problem, I need to "su", "chown drobbins.drobbins filename" and then "exit" from root. Finally, I'm allowed to modify the file.

I bring this up because a similar problem *can* occur when you share out home directories and access them using different Samba users. Consider the following situation. I access a share as administrator and created a file. Normally, this file would be owned by administrator and it would not be modifiable by the drobbins user. If drobbins tried to modify it, access would be denied. Fortunately, Samba has the "force user" option that works around this situation. The "force user" option will cause all actions performed on files (on a particular Samba SMB/CIFS share) to be performed using a single Unix account. In my "drobbins" share example, this means that any files that administrator creates will actually be owned by drobbins, preventing any ownership conflicts. Since the "drobbins" share contains the contents of my home directory, I like to keep everything in it owned by the drobbins account.

Before we head on to the next topic, I should mention the "comment" parameter. This allows you to complement your share with a descriptive comment visible from Windows.

Sharing lots of home directories

So, we've covered how to share a single home directory. But what do you do if you happen to administrate a server that contains *hundreds* of users, all of whom want to be able to access their home directories from Windows? Fortunately, Samba has a special share just for this purpose called "homes". Here's how it works:

```
[homes]
    comment=Home directory for %S
    path=/home/%u
    valid users = %u administrator
    force user=%u
    writeable = yes
    browseable = no
```

As I mentioned, this is a "special" share. It doesn't work like ordinary shares. Samba recognizes the special identifier "[homes]" and treats this share differently.

One of the most unusual things about this share is the use of the "browseable=no" parameter. This particular option causes a share to be invisible under the Network Neighborhood, and it's normally used to deter those malicious users who may be tempted to "explore" any share they can see. But why use it here?

The answer is a bit tricky. You see, the "homes" share *does* create a share called "homes". But that particular share is of no use to us. It doesn't do anything, so we hide it. What the "homes" share does do for us is quite tremendous. It tells Samba to automatically create home directories on the fly for each individual user. For example, let's say our "drobbins" share wasn't defined in smb.conf and we explored the Network Neighborhood as NT user "drobbins". We would find a share called "drobbins"

that would behave identically to our original "drobbins" share. If we accessed Samba using the NT user "jimmy", we'd find a perfectly configured "jimmy" share. This is the beauty of homes. Adding one special share causes *all* home shares to be properly created.

Now, how does it work? When the "homes" share is set up, Samba will detect which NT user is accessing Samba. Then it will create a home share that's been customized for this particular user. This share will show up in the Network Neighborhood as if it's a normal, non-dynamic share. The NT user will have no idea that this particular share was created on the fly. Let's look at what each particular option does:

The comment parameter uses the %S wildcard, which expands to the actual name of the share. This will cause the "drobbins" share to have the comment "Home directory for drobbins", the "jimmy" share to have the comment "Home directory for jimmy", etc. The path parameter also contains the wildcard %u. %u expands to the name of the user accessing the share. In this particular case, %u is equivalent %S, so we could have used path=/home/%S instead. This allows Samba to dynamically map the share to the proper location on disk.

Again, we use macros in the "valid users=" line so that only the owner of the share and administrator are allowed to access it. "force user" uses a macro too, so that all file access will be performed by a single account. And of course we make the share writeable for any authenticated users. While we use the "browseable=no" parameter, the dynamically-created shares *will* be browseable when they are created. Again, this just hides the non-functional "homes" share.

Share parameters

We've seen a couple of handy techniques to use when creating shares. In this section I'll cover several popular options that allow you to customize Samba functionality on a per-share basis. All share-related options can also be placed in the [globals] section to set a default value for all shares.

comment=

The comment = parameter is a very handy option to make your Samba system look more professional from the Windows side. It allows you to specify a comment that accompanies a particular share intended to describe its contents. When specifying comments (especially when using "homes"), I often use the %S macro, which expands to the name of the share.

path=

path= is one of the most fundamental Samba share parameters. It allows you to set the path to the directory to be exported. Note that by default any symlinks in this directory tree can be followed. So it is possible for users to "jump out" of the directory tree. From the Windows side, they will have no indication that they are following a symlink. It will just appear as a regular file or directory. We'll look at several parameters that can change this behavior to make Samba more secure.

force user=

force user= is one of my favorite parameters. It forces all file modifications to be performed by the account of a single user. You'll want to use the valid users= option often with this one so that you can limit access to select users. Since all file operations are performed using a single user account, one of the side effects of force-user= is that you can't look at the Unix file permissions to figure out who did what. Thus for writeable shares, the force user= option should be accompanied by reasonable security defaults. Without this option, all file operations will be performed by the Samba user who is accessing the share.

force user example

```
force user=drobbins
```

browseable=

One simple way to enhance your security is to make certain shares invisible. Shares are browseable by default under the Network Neighborhood. Making them invisible can help to deter unwanted hacking attempts. But this should not be used as the only means of security. Just because a share isn't listed in the browse list does not prevent it from being accessed from Windows. It just decreases the amount of information you may potentially be providing to a malicious user. To access a hidden share, you can type its UNC name into the Run... dialog box. For example, the hidden share on myserver called 'test' can be accessed by typing "\\myserver\test" from Windows.

browseable example

```
browseable=no
```

available=

The available= option, which is 'yes' by default, is just a handy way of disabling a share without commenting it out or erasing it from the smb.conf entirely. Available=no will make the share inactive after Samba is restarted.

available example

```
available=no
```

valid users=

Definitely take advantage of the valid users= option to restrict access to certain shares. By default, any authenticated user will be allowed to access a Samba share. You can refer to a valid NIS netgroup or Unix group by appending an "@" to the group name.

valid users example

To allow drobbins and the members of the wheel group access to the shares:

```
valid users = drobbins @wheel
```

dont descend=

dont descend= specifies directories in the share that Samba should not enter. This can be handy to prevent Samba from entering a directory that contains recursive symlinks, or to restrict access to irrelevant directories like /proc and /dev. Be sure to test out your dont descend= settings to make sure they are working. You may need to switch "dont descend= /dev" to "dont descend= ./dev", for example.

follow symlinks=

Follow symlinks= normally defaults to 'yes' and will cause Samba to follow all symlinks, even if they redirect Samba to files or directories outside of the exported directory tree. Setting follow symlinks to 'no' will turn off this functionality, and prevent symlinks from being followed at all. Turning off follow symlinks does eliminate a potential security hole and should be done when symlinks are not needed or required.

follow symlinks example

```
follow symlinks=no
```

volume=

The volume= option can cause Samba to associate a "volume name" with the particular share. This is especially useful if you are using a Samba share to export the contents of a CD-ROM. Many installation programs will expect to find an exact volume name on the CD, without which they won't work.

volume example

```
volume=My Favorite CD
```

create mask=

Samba uses the create mask to set the proper permissions on newly created files. The create mask defines which permissions newly created files will allow. The supplied octal number will be combined with the desired permissions using a binary "and" operation. This causes any permissions not in the mask to be dropped from the new file's permissions.

create mask example

```
create mask= 0755
```

directory mask=

directory mask= works in a manner similar to create mask=. It specifies an octal number that defines those permissions allowed for the new directory.

The many options of smb.conf

In this section, we've covered only those smb.conf options that are the most essential in configuring a useful and secure Samba system. Samba itself has many additional configuration options that you may find useful. To find out more about them, check out the smb.conf main page, where they are listed and described in detail. (See [Resources](#).)

Printing from Samba

Samba's printer-sharing abilities come in handy and work well. To refresh your memory, Samba allows you to export existing lpd-based printers so that Windows clients can connect and print to them. One of the wonderful things about this arrangement is that all printer-specific code is generated on the Windows side. This means that your Unix system does not need to have explicit support for a particular printer. As long as your Unix system is able to dump raw data to the printer, it will work and work well.

This even allows you to share and use printers that are not fully functional in a pure Unix environment, such as my Adobe PrintGear-based NEC SuperScript 870.

How Samba printing works

To get printing to work, you first need to get lpd working. While lpd configuration is beyond the scope of this article, it's not too hard and is described in detail in the Printing FAQ available at linuxdoc.org. (See [Resources](#).) You'll want to configure your printers to be "raw" printers by default, so that any data sent to the printer using an lpr command is copied verbatim without any filtering or massaging. It's easy to test lpd to make sure that it is configured in "raw" mode. On the Windows side, install a printer driver for that particular printer that prints to FILE:. Print a page from your favorite Windows word processor and save it to file. Then copy it to your Unix machine and print it using lpr. If you get correct output, you're all set to configure Samba to use the printer automatically.

Samba print globals

To get Samba working properly for printing under a Linux system, you'll want to add the following parameters to your [global] section:

```
printcap name=/etc/printcap
printing=bsd
```

If your printcap is located somewhere else, adjust the printcap name= parameter accordingly. If you use a printing system other than standard BSD lpd, consult the printing= option on the smb.conf main page for more information on how to get Samba to work with your printing system.

Now, setting up the printer share. This is what I have in smb.conf for my printer. We'll use it as a model:

```
[nec]
#my NEC SuperScript 870
path=/var/spool/smb
print command=/usr/bin/lpr %s
lprm command=/usr/bin/lprm -P%p %j
printer=lp
public=yes
printable=yes
```

It is important to first understand the path parameter. When Samba accepts a print job from Windows, it needs to store it somewhere on disk before Samba can submit the job using lpr. The directory referred to by the path= parameter should have Unix permissions 1777 so that anyone can write to this directory. The print command= and lprm= lines are not usually needed. Include them if you want to specify the exact path for your print commands or if you need to pass any command-line parameters to lpr. Use the above macros as an example. %s expands to the temporary file name, %p expands to the printer name, and %j expands to the job number.

The printer= option, as you may guess, tells Samba which Unix printer to print to. Make sure this printer is set up in raw mode. public=yes allows even users with no password to connect to this printer. Eliminate this option later if you want to tighten up security (you may want to replace this line with a valid users= line to really tighten up security). printable=yes tells Samba both that this share should be configured as a printer, and to allow this share to accept print jobs.

After restarting Samba, you should be able to see the new printer from Windows. At this point you should be able to install this printer on the Windows side and fire off a test-page to your new, shared resource. (Windows will tell you that you are installing a driver for the NULL printer. Don't worry. Just select the correct printer driver from the list.) If for some reason printing doesn't work, be sure to check /var/log/log.smb for any error messages. I should also mention that there are tons of printer-related smb.conf configuration options that you may find useful. I've just touched on the most popular ones. Be sure to check out the smb.conf main page to get familiar with all of them.

Finishing up

In this article we've covered the key elements of Samba functionality, including sharing home directories and printing. I've also tried to highlight several security-related parameters. But don't get the idea that this is all there is to Samba. Samba is not only very powerful, but also very configurable. It's designed to let you, the administrator, decide how and to what extent it will be used in your organization. While quite a bit of manual smb.conf configuration is involved in a Samba setup, the results are well worth it because you can get everything working *exactly* how you want.

Samba has additional functionality that we haven't even touched on, including the ability to become part of (or even control!) an entire Windows NT domain. I encourage you to fully explore the potential of this extremely powerful tool.

Resources

- Download Samba at the main [Samba](#) Web site
- [frgpasswd](#) is a password synchronization utility to set Samba and shadow passwords concomitantly
- [GnoSamba](#) is a GUI tool for configuring Samba
- [KNetmon](#) is a front end for the Samba and smbfs packages
- [SambaLink/Q](#) is a version-independent editor for the smb.conf file
- [SMB Mode for Emacs](#) can help you edit Samba's configuration file, smb.conf
- See the Printing FAQ at [linuxdoc.org](#)
- Read [Samba](#) by Ed Weinberg
- IBM Learning Services offers a [2-day hands-on course](#) on Samba
- [Using Samba](#) (O'Reilly Publishing; 1999) is a comprehensive guide to Samba administration, including such recent additions as integration with Windows NT domains and the SWAT graphic configuration tool
- Visit the [SWAT](#) main page
- [Samba Notes for Redhat](#) provides the latest, but not quite finished, version of the notes, covering Samba 2 on Red Hat 6, and an older version, covering Samba 1 on Red Hat 5.x
- Check out [Samba/iX](#), a suite of programs that allow an HP e3000 running MPE/iX operating system to provide service using Microsoft's Message Block (SMB)
- See the [Kernel Cousin Samba](#) on Linuxcare
- [xSMBrowser](#) is a Samba browsing utility that supports both WINS and Broadcast networks
- Read [Samba Unleashed](#), by Steve Litt, with contributions from Daniel Robbins
- Subscribe to the [Amiga Samba](#) mailing list
- Check out a line of [Samba clothes](#) from Nerdgear

About the author

Daniel Robbins resides in Albuquerque, New Mexico. He is the President/CEO of Gentoo Technologies, Inc., the Chief Architect of the [Gentoo Project](#) and a contributing author of several books published by MacMillan: *Caldera OpenLinux Unleashed*, *SuSE Linux Unleashed*, and *Samba Unleashed*. Daniel has been involved with computers in some fashion since the second grade, when he was first exposed to the Logo programming language as well as a potentially dangerous dose of Pac Man. This probably explains why he has since served as a Lead Graphic Artist at SONY Electronic Publishing/Psygnosis. Daniel enjoys spending time with his wife, Mary, and his new baby daughter, Hadassah. You can contact Daniel Robbins at [drobbins@gentoo.org](mailto:d Robbins@gentoo.org).

What do you think of this article?

Killer!

Good stuff

So-so; not bad

Needs work

Lame!

Comments?

[Privacy](#) [Legal](#) [Contact](#)