Using Samba as a PDC

Skill Level: Introductory

Tom Syroid (dwcomments@syroidmanor.com)
Contract writer

03 Apr 2002

Open-source Samba turns a UNIX or Linux system into a file and print server for Microsoft Windows network clients. Tom Syroid dishes up a juicy tutorial that shows you how to configure Samba as the primary domain controller on an xSeries server.

Section 1. Introduction to Samba

Tutorial roadmap

Although you don't need to be a Samba expert to complete this tutorial, a basic knowledge of common administrative tasks is helpful, including tasks such as copying and moving files, creating user accounts, and setting permissions.

Upon completion of this tutorial you will know how to:

- Install Samba using the Redhat package manager (RPM) or by compiling the program from source.
- Configure the Samba server as a PDC (Primary Domain Controller).
- Configure the server to support roaming profiles.
- Configure the server to support netlogons.
- Create the required administrative directories on the server, and set their permissions as appropriate.
- Create user and machine (trust) accounts on the domain controller.
- Configure the client machines to join the domain.
- And troubleshoot any basic problems that may arise in the above process.

What is Samba

Samba is arguably one of the most successful Open Source projects ever created. It began like so many other similar projects: as a need. Andrew Tridgell needed a fileserver program for his local network that supported an old DEC protocol from Digital Pathworks. Unbeknownst to Andrew, this same protocol later became SMB--the language of Microsoft filesystems. In the 11 years since its inception, Samba has evolved into a stable, reliable product, and has been ported to numerous platforms (AIX, HP-UX, BSD, Linux, DigitalUnix, IRIX, SCO, VMS, OSF, and Solaris). Once installed and configured, Samba provides an almost seamless integration with an existing Windows networks, and for all intents and purposes from the user's perspective, is simply another Windows fileserver, printer server, or domain controller. Best of all, Samba is free and there are no traditional licensing issues to contend with. Regarding PDC functionality, the topic of this tutorial, Samba currently supports:

- Domain logon from Windows NT/2000 clients.
- User level security for Windows 9x/ME clients (Win9x/ME clients have no concept of domain-level security; they simply know how to logon to a domain controller).
- Roaming profiles.
- NT4-style system polices.
- Browse lists.
- Retrieve of user and group lists stored on a Samba PDC.

The current stable release of Samba at the time of this writing is version 2.2.3a, which is used as the basis of this tutorial. Note that the current SAMBA_2_2 CVS development tree is close to release (slated for mid-April 2002), which will become 2.2.4. Check Samba's Web site (www.samba.org) for release notes and details; if Samba 2.2.4 is avaiable, use it--it includes several important bug fixes relating to printing and domain controller functionality.

Current program constraints

While the functionality in Samba compares favorably with Windows NT/2000 server, Samba does have some limitations you should be aware of:

- Samba cannot function in the role of a BDC (Backup Domain Controller). There is currently no way to establish a "trust" relationship between a Samba domain controller and a Windows NT/2000 controller. Having said that, Scott Merrill has devised a way to simulate a PDC/BDC environment using two Samba machines. See the Resources section for details and a URL.

- Along similar lines, Samba cannot replicate SAM (user authentication information) with a Windows NT/2000 server (or visa-versa).
- Samba cannot yet read or utilize Windows Active Directory (AD).
- Samba does not support adding users via the Windows User Manager for Domains (primarily due to the above limitations).

Stay tuned, however. Many of the above limitations (primarily the ability to talk to an AD tree and SAM replication) will vanish with Samba 3.0, which is currently in active development (release is slated for late-summer 2002). In addition to connectivity with Windows AD, Samba 3.0 will also (likely) feature: Microsoft Kerberos authentication, a completely re-written and configurable authentication subsystem, a Microsoft-like "NET" command for terminal access to a variety of network resources, improved printing support, and the usual bug fixes and stability enhancements.

For more details on the status of Samba 3.0 and other developmental projects, see www.samba.org/samba/.

Hardware considerations

On the whole, Samba is an extraordinarily efficient program, and consumes very few system resources for the amount of work it does. But like any front-line software program, it really comes down to scale--you cannot expect to run 200 Samba users off a Pentium 166 with 64MB of RAM and a 10GB IDE hard drive without complication (and a whole slew of upset users). On the other hand, that same Pentium 166 acting as a Samba file server for a small home office will probably do an admirable job given the file transfer load is light.

Generally speaking, from a hardware perspective I recommend the following:

- Lots of memory--the more the better. Every user connection spawns a Samba daemon, so the more users you plan to host off a machine, the more memory it should have.
- Disk access is critical. I've built perfectly adequate Samba file servers from old workstations (PII-400) by simply adding a SCSI subsystem and a good quality SCSI hard drive. Again, think scale--the more users you plan to serve, the higher quality your disk subsystem should be.
- My personal preference is for SMP file servers. Multiple processors do not necessarily make for a faster machine, but it does make for a machine that does not slow down under load. And when you consider the fact you can build a dual Celeron system for about the same cost as a high-end single P4 processor system, to me the choice is a no-brainer.

For the record, the hardware used to write and test this tutorial consisted of one of IBM's new xSeries servers: An xSeries model 220, dual Pentium 3 1.2GHz CPUs, an IBM ServeRAID hardware RAID controller, 1GB of SDRAM, and three 10,000 RPM 18GB hard drives in RAID5 configuration (netting approximately 36GB of useable hard disk space). The machine performed admirably in all regards, and continues to do so without error or failure.

For more details on IBM's server product line, see the eServer Developer Domain or for xSeries specific information, the xSeries Intel processor based servers site. There is also an excellent IBM Redbook titled *Samba Installation, Configuration, and Sizing Guide* (SG24-6004-00) that has a good section on hardware requirements and sizing.

Tutorial prerequisites

Installing Samba and configuring it as a PDC requires that certain steps must be followed. Although the reader does not need to be a Samba expert to benefit from this tutorial, a basic knowledge of common administrative tasks such as copying and moving files, creating user accounts and setting permissions is helpful.

If you want to follow through the examples you need the following configuration and tools:

- This tutorial is based on Redhat 7.2 with all the current errata applied (see Redhat's Errata page for details). You can, of course, use any UNIX or Linux distribution you choose (heck, you can even use HP-UX or Solaris!)--just be aware that file locations and paths noted are Redhat specific. Make sure, however, that whatever distribution you choose is working as advertised before attempting to install and configure Samba.
- Working knowledge of a text editor like vi is a good skill to have and/or develop. There is a browser-based configuration tool discussed (SWAT), and information on this tool can be found in the Troubleshooting and SWAT information section.
- If you plan to compile Samba from source, which is the recommended and demonstrated approach, ensure that gcc is installed and correctly configured on your machine.

Section 2. Building and configuring a Samba PDC

Server-side configuration

This section covers server-side details of building and configuring a PDC using Samba. When building and configuring the PDC it is important to pay attention to details. When a project fails to work as advertised, nine times out of ten it's human error: a missing compile-time option, a spelling mistake in the configuration file, a permission bit set incorrectly, etc. The message here is to be methodical.

The following general steps are required to bring a Samba PDC online:

- Build/install Samba (if not already installed)
- Configure the `smb.conf` file
- Create the required directories on the server (if logon scripts or roaming profiles are to be used)
- Create the user and machine accounts
- Test the configuration and start the daemons

The final step is to configure the individual clients and join them to the domain; this is discussed in the next section, Client Configuration.

Installing Samba

The first step is to ensure Samba is installed on your system. If not, you need to install it. The easiest way to check this (again, we're assuming Redhat 7.2 in this tutorial) is to type:

```
rpm -qa | grep samba
```

which will either produce a list of Samba-related RPM packages installed, or generate a blank line.

If Samba is installed, note the version number. When building a Samba domain controller, experience suggests using at least version 2.2.2. Better yet, use the latest stable code, which at the time of this writting is 2.2.3a. A lot of work has been done on domain controller functionality in the last two or three revisions. In addition, print functionality has been seriously reworked, the LDAP backend is now stable enough to be considered "production" capable, and scores of small bugs have been eradicated. All-in-all, version 2.2.3a is a stable, refined, and well-tested release.

If you plan to install or upgrade, there are two ways to go about it: compile from source, or install a pre-built RPM package. We'll take a look at the compile from source option in the next panel.

When installing via RPM, there are a couple caveats to keep in mind.

First, Redhat split their binary Samba RPM's into three parts: `samba-common-xxxxx.rpm`; `samba-server-xxxxx.rpm`; and `samba-manual-xxxxx.rpm`, where the x's represent the release version. The Samba team also builds binary packages for a variety of platforms (see http://us1.samba.org/ftp/Binary_Packages), but he chose to create a single installation RPM for Redhat distributions. So to ensure everything gets upgraded properly, uninstall any existing Samba packages before installing a new version! (`rpm -e samba`).

Second, if you have an existing configuration file that you want to keep, back it up somewhere safe. It is possible to lose a valued configuration file to an installation gone awry.

Type man rpm for an explanation of the various options available for installing RPMs. The command `rpm -ivh` samba-package should serve most needs.

Building from source

Many administrators like to build all key server programs (for example, Apache, BIND, Samba, and so forth) from source. The reason is simple: building from source means you can tailor a program's features and components to your needs. You also know exactly what you've got on your system in terms of "clean, untouched" source code. Building Samba on a Redhat distribution can get a little tricky, however, due to variances in directory/file placements. The folks at RH designate /usr as the base *installation* directory. This means user-accessible utilities go in /usr/bin, and administrator utilities go in /usr/sbin; configuration files are placed in `/etc/samba`. A default Samba configuration (meaning built from source with no configuration options), on the other hand, uses `/usr/local/samba` as the base. So binaries are installed in `/usr/local/samba/bin`, secure binaries go in `/usr/local/samba/sbin`, and, rather unintuitively, Samba's configuration files end up in `/usr/local/samba/lib`.

The "gotcha" comes when someone unfamiliar with the above conventions installs Samba from source without uninstalling the Redhat version. He end up with two different versions on his system, and cannot figure out why -- after installing a shiny new Samba release -- his old version is still running. The old version is invoked because:

- The old version is on the "path"; the new version is not.
- The old version (RH built) is automatically started by a script in `/etc/init.d` called samba (on some installations, this file may be called smb); an installation from source does not always create or correct the paths in the script.

TIP: You can determine Samba's version by typing `/usr/local/samba/bin/smbd -V` (or the path to wherever your Samba binaries reside).

So when building from source the first decision that must be made is: Where will Samba reside?

There's nothing wrong with installing Samba in the `/usr/local tree`; you just have to remember it's there, and remember to type the full path when running one of the program's utilities (or add the file locations to the path). The latter is really not a big issue, as once Samba is configured, started, and running, it just goes, and goes, and goes...

On the other hand, if your policy is to keep all program files in the bin/sbin directories, that too is possible. The next panel contains a step-by-step checklist for building Samba from source, plus an example `./configure` command to do just that.

Building from source: A checklist

The following checklist should get anyone unfamiliar with building from source up and running with minimal fuss and frustration:

1. If you have an existing Samba installation (in other words, from a RH RPM package), go to `/etc/init.d` and copy the file samba (or smb) to a safe place for future reference. Also, as noted earlier in the tutorial, make a copy of your existing `smb.conf` "just in case".

2. Remove any existing Samba installations.

3. Download the source tarball from your favorite Samba site. The latest stable release is always called `samba-latest.tar.gz`. See the Resources section for the requisite URLs.

4. I prefer to build all source code under `/usr/local/src`, so this location is used for reference--substitute as desired. Copy the `tar.gz` file to `/usr/local/src` (or better yet, download the tarball directly there) and type tar `xvzf samba-latest.tar.gz`. CD to the source directory (`/usr/local/src/samba/source`). Note that many applications are configured to `./configure` from the unzipped root directory (`/usr/local/src/samba`). This is not the case with Samba, which is a common mistake made by people unfamiliar with the program. If you're comfortable with Samba installing under `/usr/local/samba`, then `--prefix= and xxxdir= options` are not necessary.

5. Type `./configure --help` to see Samba's configuration options. Below is a Redhat configure option recipe. The directory placement options shown will put Samba in the same locations as a Redhat built RPM.

6. Once you've achieved an error-free configuration, type `make`.

7. To copy the binaries to his final resting place, type make `install`.

8. A word of caution: If you ever have to return and reconfigure Samba with different options, be sure to delete the `/usr/local/src/samba/source/config.cache` file before re-running the `./configure` command. If you don't, Samba will configure itself using the previous option-set.

That's it! You should now have a functional Samba installation. Now to configure it to do something...

Listing 1. Samba installation

```
./configure \
--prefix=/usr \
--bindir=/usr/bin \
--sbindir=/usr/sbin \
--libexecdir=/usr/libexec \
--datadir=/usr/share/samba \
--sysconfdir=/etc/samba \
--localstatedir=/usr/local/samba/var \
*** CHECK THE ABOVE ***
--libdir=/usr/lib \
--with-lockdir=/var/locks/samba \
--with-swatdir=/usr/share/samba/swat \
--with-codepagedir=/etc/samba/codepages \
--with-configdir=/etc/samba \
--with-smbwrapper \
--with-automount \
--with-smbmount \
--with-pam \
--with-pam_smbpass \
--with-winbind
```

Introducing smb.conf

The power and flexibility of Samba is controlled by a single configuration file, `smb.conf`. As noted earlier in this section, `smb.conf's` location is dependent on the program location options used when the program was built. Two typical locations for smb.conf are `/usr/local/samba/lib` and `/etc/samba`. The first thing anyone new to Samba should do is: (a) make a copy of `smb.conf` so you always have a "clean" original; and (b) print it out and READ IT! `smb.conf` is extensively documented, and a goldmine of valuable information for novices and seasoned veterans alike.

The structure and layout of `smb.conf` is extremely simple. It consists of two main sections: [global] contains option statements that apply "globally" to the program, and a "shares" section. Each share begins with the name of the share enclosed in square brackets (for example, [homes]) and a list of option statements that apply to that share. Here's an important note to file away for future reference: Every Samba option statement has a default value. So specifying a different value in the global section overrides the default for the server as a whole, and specifying a value in a shares section overrides both the global option and the server's default option (if different).

A simple example is in order. By default (for example, the global default option) Samba allows anyone who passes the authentication process--typically a valid username/password combination--access to a listed share. An administrator can, however, restrict user access to a share by using the valid users = option. For example:

Listing 2. An example

```
[homes]
        comment = Home Directories
        valid users = tom, leah, suzie, bilbrey
        read only = No
        browseable = No
```

The above share can only be accessed by the users tom, leah, suzie, and bilbrey, effectively overriding any other options (implied or otherwise) specified in the [global] section of the configuration file.

Enough theory. Let's move on to the fun stuff--configuring a Samba domain controller.

Basic server settings

Too many admins create initial configuration files that are six- or seven-page, and find themselves pulling out copious quantities of hair if their configurations fail to provide the expected results. All of this leads to an unnecessarily complex six- or seven-page. As you're about to see, a "basic" configuration that provides domain authentication, a logon script option, roaming profiles, and a few basic shares, is really quite simple to build. So in the spirit of keeping it simple, we're going to put our smb.conf file together in small, discrete pieces. This allows you to see which options do what, and digest the process in chewable chunks. The completed, fully functional `smb.conf` can be downloaded from the Resources section.

Rather than mess around deleting and editing the six or seven page configuration file installed with Samba, we're going to start with a blank slate and add the following domain/machine options:

Listing 3. Domain/machine options

```
# /etc/samba/smb.conf
# samba configuration file
# last updated: 2/28/2002 by tms

[global]

  ;basic server settings
  workgroup = syroidmanor
  netbios name = phoenix
  server string = Samba PDC running %v
  socket options = TCP_NODELAY IPTOS_LOWDELAY SO_SNDBUF=8192 SO_RCVBUF=8192
```

Comments begin with a hash (#) or semicolon (`;`) and are ignored; so too is whitespace. For those who don't already, providing liberal comments in your configuration files is an excellent habit to get into (especially when you make configuration changes; note the change and why it was made). The workgroup option, in the case of a domain controller, sets the domain name the controller will serve. Yes, I could have used `syroidmanor.com`, but `syroidmanor` works and it's less typing for both user and admin. The netbios option is the host name. So the FQDN (Fully Qualified Domain Name) of this system is `phoenix.syroidmanor[.com]`. The `server string` provides a description that appears next to the share when browsing your "Network Neighborhood" from a Windows client. Next we need to tell Samba this machine is the PDC. Finally, the `socket options` setting controls TCP/IP performance. The settings shown are known to work well with Linux-based systems. If you're using another OS, consult one of the documentation references listed at the end of this tutorial.

PDC and master browser settings

First, a word of caution: There can be only ONE PDC on the network. Having more than one PDC on a network will cause you no end of head-scratching and unexplained weirdness--don't go there.

To our existing smb.conf file we add the following options to the global section:

Listing 4. smb.conf file -- Global section

```
[global]
        ...
        ;PDC and master browser settings
        os level = 64
        preferred master = yes
        local master = yes
        domain master = yes
        ...
```

To quote from O'Reilly & Associates' seminal title *Using Samba* (Eckstein, Collier-Brown, and Kelly; see the Resources section for more details): ...one machine in each subnet always keeps a list of the currently active machines. This list is called the browse list and the server that maintains it is called the *local master browser*. As machines come on and off the network, the local master browser continually updates the information in the browse list and provides it to any machine that requests it." Given that local master browser could itself become unavailable, *election* are called on a routine basis to determine who's authority is absolute (that is, which machine holds the most accurate local master browse list). The winner of an election is determined by a number of factors: election protocol (meaningless at this point in time), os level, the preferred master setting, the time online, and finally, alphabetically according to the machine's NetBIOS name. The first three settings in the above configuration ensure this machine is always consulted first regarding the current browse list. The `domain master` option is the "switch" that tells Samba to be the primary domain controller.

Security settings

The next step is to add some security and logging options:

Listing 5. Security and logging options

```
[global]
        ...
        ;security and logging settings
        security = user
        encrypt passwords = yes
        domain logons = yes
        log file = /var/log/samba/log.%m
        log level = 2
        max log size = 50
        hosts allow = 127.0.0.1 192.168.1.0/255.255.255.0
```

...
Samba supports four security options: share, user, server, and domain. The security option must be set to user on a Samba PDC. For an explanation of the other security modes, see page 164 of the previously referenced *Using Samba* title. The `encrypt passwords = yes` option is also mandatory for a PDC. We'll touch briefly on the implications of this setting in the Client Configuration section of this tutorial. The `domain logons` option simply tells Samba to support domain logons on this machine (versus authenticating via the PDC and then sending the client to another machine for logon scripts, home directories, etc.). The next option determines where log files are kept; %m is a *substitution variable*. It is replaced with the netbios name of the connecting machine (see *Using Samba*. Acceptable variables range from 1 to 10. Running a log level above 3 is not recommended unless very detailed debugging information is required; a high logging option will slow your system dramatically and produce copious output. In order to keep log files under control (filling your /var partition has very "ungood" consequences), we set `max log size = 50`. This constrains each log file to a maximum of 50 kilobytes; when the file reaches this size, new entries displace the oldest entries. Last but not least, we ensure that only machines from our internal 192.168.1 subnet can connect to the server. Don't forget to add the *localhost*entry, especially if you plan to use SWAT.

Roaming profiles

Next up, we're going to add support for *roaming* profiles. A *local* profile consists of all the files and settings (Desktop, application configuration files not stored in the registry, Internet cache files, etc, etc, etc.) stored on the user's machine under `C:/Documents and Settings/username`. With roaming profiles (or just profiles) enabled, these settings are stored on the local machine in a different folder (for example `C:/Documents and Settings/tom.SYROIDMANOR`) and on logout, saved to a directory on the PDC. When the user logs back in, the server settings are then restored to the local host. The point of all this fanciness is to allow a user to logon to the domain from any machine on the network, and have their saved Desktop settings, Start Menu, and various configuration settings appear like magic. There is one caveat to profiles: They can consume a LOT of bandwidth synchronizing the required files. If your network is already stretched thin, roaming profiles are definitely not a good idea. First, let's populate our configuration file, then we'll explain what the various options do.

Listing 6. Configuration file

```
[global]
        ...
        ;user profiles and home directory
        logon home = \\%L\%U\.profile
        logon drive = H:
        logon path = \\%L\profiles\%U
        ...

#===Shares===

[homes]
        comment = Home Directories
        browseable = no
        writeable = yes

[profiles]
        path = /home/samba/profiles
        writeable = yes
        browseable = no
        create mask = 0600
        directory mask = 0700
```

The most important point to note with the above global option block is that the directories referenced must be accompanied by a matching share (in this case, profiles). And ensure you spell the global reference and the share name the same. A common error is to spell one "profile" and one "profiles". When this occurs, users are denied access to the PDC. In addition, the directories must exist and must have the appropriate permissions set on them. Directory creation and permissions are detailed later in this section.

As you can see, the logon path = option uses variable substitution and pairs with the `[profiles]` share. Assuming my username is tom and I'm trying to connect to the PDC named phoenix, the logon path line would substitute out as \\phoenix\profiles\tom (%L = the Samba server's netbios name = phoenix; %U = the username requesting the share = tom). The profiles name in the proceeding references the `[profiles]` share. Putting it all together, we come up with the following: "When user tom tries to log into phoenix, his profile can be found in the `/home/samba/profiles/tom` directory (for a first-time logon, Samba will create the tom directory). For security we make the `[profiles]` share non-browseable (hidden to anyone browsing Network Neighborhood), writeable (mandatory if a user's profile is to be updated/kept in sync), we set the create mask to 0600 (rwx-xxx-xxx--only the user can read/write the files there), and we force any directories created to 0700 (rwx-xxx-xxx--directories must be executable if he is to be navigated).

One other thing to be aware of: Windows NT/2000 clients implement profiles differently than Windows 9x/ME clients, hence the two different approaches to the "logon" path. The logon home is Win9x/ME specific; Win9x/ME restricts profiles to the user's home directory. The logon path is Windows NT/2000 specific, and has no such restrictions. Having both option in your configuration file simply broadens its applicability. Of course, if you have no Win9x clients on site, feel free to delete as appropriate.

Next, we're going to look at the special `[homes]` share and how it works.

The [homes] share

The `[homes]` (Caution! That's "homes", not "home") share is one of three special sections within Samba's configuration file (the other two are `[global]` and `[printers]`). If a client attempts to connect to a share that doesn't exist, and the `[homes]` share is present in smb.conf, Samba assumes the user is trying to connect to their home directory. The program then searches its user database (smbpasswd; stay tuned for more on this topic) for a username and password combination the same as the user requesting the invalid share. If found, the username is substituted into the logon home option statement. Using our friend tom as an example, the global option would expand to `\\%L\%U`, which in turn translates to `\\phoenix\tom`. If no share called `[tom]` exists, Samba creates one using the path = option supplied in the `[homes]` share and assigns the share the Windows drive letter... Yep, you guessed it... H: (per the global option logon drive = H:; this can be any drive letter not already in use on the client machine). If no path = statement exists under `[homes]`, /home/username is assumed. Pretty smart program, wouldn't you say?

We have one more global/shares option pairing to consider: netlogon.

The netlogon option

The final piece we're going to add to smb.conf (well, at least for now...) is the logon script option. Like the [homes] and [profile] shares, there are two pieces to the puzzle:

Listing 7. Netlogon option

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

# === shares ===
        ...
[netlogon]
        path = /home/netlogon
        read only = yes
        write list = tom
```

The `[netlogon]` share is administrative tool used primarily for globally updating client machines with items like registry patches, anti-virus updates, program updates, etc. Anything you want to "push" out to the client, can be done via netlogon. In addition, you can use the share to enforce a system policy on a client or clients or perhaps backup a select group of files every time the user logs on. Creating scripts to run from netlogon is beyond the scope of this tutorial, but you'll find plenty of information on the Samba.org site and by searching Google with the string "samba scripting".

Here's how netlogon works: Any time a user logon onto the PDC and the `logon script = option` and `[netlogon]` share are present, Samba goes to the indicated path and executes the file referenced by logon script. Once again, some brief caveats are in order:

- The file referenced by logon script can be called anything as long as Windows recognizes it as an executable file.
- If the file referenced is not found, Samba continues on its merry way and connects the user to the requested share(s).

- Be sure to set the UNIX-side permissions to executable.
- To delegate responsibility for scripts in the netlogon share, create an "admin" group and add this group to the `write list = option` in the form: `write list = @admin` (the '@' notation signifies a group).
- Be sure to test your scripts thoroughly in a non-production environment. Trashing the registry on 600 client machines is not a good career move.

The final cut

Below, for reference, is the fully assembled Samba PDC configuration file:

Listing 8. Samba PDC configuration file

```
# /etc/samba/smb.conf
# samba configuration file
# last updated: 2/28/2002 by tms

[global]

   ;basic server settings
   workgroup = syroidmanor
   netbios name = phoenix
   server string = Samba PDC running %v
   socket options = TCP_NODELAY IPTOS_LOWDELAY SO_SNDBUF=8192 SO_RCVBUF=8192

   ;PDC and master browser settings
   os level = 64
   preferred master = yes
   local master = yes
   domain master = yes

   ;security and logging settings
   security = user
   encrypt passwords = yes
   log file = /var/log/samba/log.%m
   log level = 2
   max log size = 50
   hosts allow = 127.0.0.1 192.168.1.0/255.255.255.0

   ;user profiles and home directory
   logon home = \\%L\%U\
   logon drive = H:
   logon path = \\%L\profiles\%U
   logon script = netlogon.bat

# ==== shares ====

[homes]
  comment = Home Directories
  browseable = no
  writeable = yes

[profiles]
  path = /home/samba/profiles
  writeable = yes
  browseable = no
  create mask = 0600
  directory mask = 0700

[netlogon]
  comment = Network Logon Service
  path = /home/netlogon
  read only = yes
```

```
   browseable = no
   write list = tom
```

Next is the topic of authentication: adding user and machine accounts to the domain controller.

Section 3. Directories, accounts, and authentication

Creating the PDC administrative directories

Now that we have a PDC configuration file in place, it's time to finish up the remaining server-side administrative items; namely, creating the requisite directories required by the shares in `smb.conf`, setting the appropriate permission on these directories, and adding the necessary user/machine accounts.

But first, let's create two groups to help manage the domain:

```
[root@phoenix root]# group -g 200 admins
[root@phoenix root]# group -g 201 machines
```

The first command creates the admin group with a GID of 200 (chosen so as to not conflict with any other existing groups); the second creates a machine group with a GID of 201. The first group will contain users who are allowed to administer certain aspects of the PDC. The second group is a convenient way to organize the machine accounts we'll be creating shortly.

Now that the above groups are in place, we can go ahead and create the two required directories and set the correct ownership.

```
[root@phoenix root]# mkdir -m 0775 /home/netlogon
[root@phoenix root]# chown root.admins /home/netlogon
[root@phoenix root]# mkdir /home/samba /home/samba/profiles
[root@phoenix root]# chmod 1757 /home/samba/profiles
```

Setting the correct permissions and ownership on the above directories is a critical step in securing your server. Don't forget, when a client logs onto the server, any file in the `/home/netlogon` directory named (in our example configuration) `netlogon.bat` will be automatically downloaded and executed. It doesn't take a brain surgeon to see how easy it would be to write a destructive script or plant a trojan on every client on your network. The "chown" command set on the `/home/samba/profiles` is equally important to user security. In effect, this command sets the directory permission such that root owns everything down to the "profiles" branch of the tree, and the user owns everything below that (the directory created to hold their profile and all the information it contains). This mean a user cannot "climb out" of their profile directory and accidently (or intentionally) mess with any other user's files.

Double-check that you've typed in the above commands exactly as shown -- this step is an important one.

Authentication: user and machine accounts

Now that Samba is configured and the required directories are in place, it's time to add user and machine accounts to the domain. Unfortunately, this is a rather complex topic, and once understood, a bit of a tedious process. The crux of the problem lies in the fact that Windows passwords and UNIX/Linux passwords are different beasts. The "bridge" between the two incompatible formats is the Samba password file (in our compiled example, `/etc/samba/smbpasswd`). The Samba password file, however, demands a corresponding UNIX account on the same machine. So what we end up with, is two kinds of accounts (user and machine) in two different passwords files (UNIX and Samba, or `/etc/passwd` and `/etc/samba/smbpasswd`, respectively).

The user accounts are self-explanatory and easy to grasp. Machine accounts--which also require entries in both the UNIX and Samba password files--are known as *trust accounts* (in Windows parlance, *computer accounts*). When a trust or machine account is created, a "secret" is automatically generated (similar in concept to a unique machine name/password combination). This secret is used as a means of secure communication between the client and the domain controller, and to prevent an unauthorized machine with the same NetBIOS name from joining the domain and gaining access to data stored there.

Here's where things get a little tricky. Windows NT/2000/XP clients fully support the concept of trust accounts; Windows 9x/ME clients do not. As a matter of fact, the only concept of domains supported by Windows 9x/ME is the logon mechanism whereby the client will download a system policy if there is one present, and store a profile on the server. Windows 9x/ME has no concept of machine trust or security, which makes it remarkably easy to "spoof" a domain controller into accepting a logon from a Win9X machine that is not who it claims to be. That's why Windows 9x/ME clients are particularly unsuited to domain-type networks. Now that we have that little bit of trivia out of the way, let's move on to creating machine or trust accounts on the PDC. There are two methods available:

- Manual creation where both the UNIX and Samba passwords are added "by hand".
- Automatic creation via an "add user" script in the `smb.conf`.

Machine accounts: the manual approach

As noted, Samba will not allow you to add an entry to the smbpasswd file (user or machine) unless there is a existing UNIX account for that user. So the first step is to create an entry for the client in `/etc/passwd`:

```
[root@phoenix root]# /usr/sbin/useradd -g machines
     -d /dev/null -c "machine id" -s /bin/false machine_name$
[root@phoenix root]# passwd -l machine_name$
Changing password for user machine_name$
Locking password for user machine_name$
```

The first command creates the user machine_name (don't forget the dollar-sign; it's required and identifies the entry as a trust account)), as a member of the group machines (`-g`), with no home directory (`-d /dev/null`), a descriptive entry (`-c`; for example, "Tom's Notebook"), and no shell access (`-s /bin/false`). The second command creates a "secret" for the machine to authenticate against.

With the UNIX account created, we can now add the machine to `/etc/samba/smbpasswd` as shown below:

```
[root@phoenix root]# smbpasswd -a -m machine_name
Added user machine_name$
```

Two things to note in the above command: One, if you installed Samba under `/usr/local/samba`, you'll probably have to provide the complete path (ie, `/usr/local/samba/bin/smbpasswd`). Two, when entering the machine_name, do not append a dollar-sign; it's not required with smbpasswd.

WARNING: Once a trust account has been created on the PDC, it's good policy to connect the client ASAP (which, in effect, changes the machine "password" and syncs the secret between the server and the client). Until the client formally connects to the PDC, the domain is vulnerable to another machine connecting with the same NetBIOS name.

Machine accounts: an automated approach

The second approach to creating machine/trust account on the PDC is to allow Samba to create them as needed when the client first joins the domain. This little bit of magic is accomplished by adding an `add user script` option to `smb.conf`. This creates the UNIX trust account, and tells Samba to automatically create a corresponding entry in smbpasswd. The following is an example of an entry based on a Redhat distribution:

```
[global]
...
add user script = /usr/sbin/useradd -d /dev/null -g machines -s /bin/false -M %u
...
```

The important thing to note in the above command is that the command to add users may vary across operating systems and/or distributions, so tweak accordingly.

Adding user accounts

The last piece of information we need to provide the PDC is a means of authenticating users. As discussed at the beginning of this section, this is accomplished by adding user accounts to both `/etc/passwd` and `/etc/samba/smbpasswd`. Unfortunately, there are no cute configuration options to automate this process (a shell script might ease the tedium). Here are the three commands necessary to create the two required user accounts:

```
[root@phoenix root]# useradd leah
[root@phoenix root]# passwd leah
New password:
Retype new password:
passwd: all authentication tokens updated successfully
[root@phoenix root]# smbpasswd -a leah
New SMB password:
Retype new SMB password:
Added user leah.
```

Note that you'll need to create a root user account in order to join Windows NT/2000 machines to the domain. Treat the password you use with the same care and security as you would the UNIX root password; it has all the same authority.

TIP: I make it a policy to make UIDs the same for all systems on my network; it saves a whole bunch of authentication issues down the road. I also make sure that every user has only one password--that is, I use the same password for UNIX and Samba. Again, it makes for easy system and user management.

Keeping user accounts in sync

The last topic of this section is password synchronization. One of the challenges with using Samba is trying to keep user passwords in sync between UNIX and Samba. There is another way of accomplishing this other than going to a backend user-management system like LDAP or NIS. The options statements below will allow a user to change their Samba password from a Windows client, which will in turn update their UNIX password to match the new Samba entry. If the UNIX password is changed, however, the same technique does not work in reverse; the Samba password will have to be manually sync'd.

```
[global]
  ...
  ;sync UNIX passwords
  unix password sync = yes
  passwd program = /usr/bin/passwd %u
  passwd chat =
    *New*UNIX*password* %n\n *Retype*new*UNIX*password* %n\n *Enter*
  new*UNIX*password* %n\n *Retype*new*UNIX*password* %n\n *passwd: *all*
  authentication*tokens*updated*successfully*
  ...
```

About the only thing that bears mentioning in the above statements is that the `passwd chat` option, despite how it might display here, is entered on one line. Note also that some options use "password" and others use "passwd".

Configuration of the Samba PDC is complete. The only thing left to do is join the clients to the domain.

Section 4. Client configuration

Joining clients to the domain

Unfortunately, there's no easy way around it--joining Windows clients to a domain (assuming, of course, they were not a member of a domain previously) is a hands-on job. If you're lucky enough to be working with a single client across all users, then perhaps a HOWTO could be constructed and circulated, but most administrators are not so blessed. The difficult issue in a mixed-client environment is finding any sort of consistant approach. Every time Microsoft releases a new version of Windows, they also seem to introduce a new way to configure networking. A new dialog here, an extra checkbox there, some systems require you to go through the Control Panel, others by right-clicking on My Computer--all of which makes for a lot of confusion from the user perspective.

The process is actually relatively painless and mechanical, but it does differ across the gamut of Windows releases. With that in mind, the best approach is to "divide and conquer", which is exactly what we've done here.

Client configuration: Windows 95/98/ME

To join a Windows 95/98/ME client to the domain:

1. First check that Client for Microsoft Networks is installed; if not, install it (**Control Panel > Network > Client for Microsoft Networks**). To install, place your Windows CD in the drive and select Add from the afforementioned dialog, then: **Client > Add... > Microsoft > Client for Microsoft Networks**.

2. Make sure Client for Microsoft Networks is the primary network protocol (**Control Panel > Network > Primary Network Logon**).

3. Next, go to **Control Panel > Network > Client for Microsoft Networks > Properties > Logon to NT Domain**.

4. If you've employed the add `user script` option, select the checkbox Create a Computer Account in the Domain; otherwise you'll need to ensure a machine account already exists for the client.

5. Fill in the domain, and click **OK**.

Client configuration: Windows NT/2000

Under Windows NT:

1. Go to **Control Panel > Network > Identification > Change option**. If the machine is currently configured under the Workgroup option, select the Domain radio button and enter the domain name.

2. Select **Create a Computer Account** in the Domain as necessary.

3. Now, logon to the domain using the *username root* and the appropriate *password*. This is necessary to initialize the "secret" between the server and client machines. From here forward, any authenticated user can logon from this machine.

4. A message should appear welcoming you to the *domain_name* domain.

The steps are the same for Windows 2000 except the network settings are found under **Control Panel > System > Network Identification** (or right-click the My Computer icon on your desktop, choose **Properties**, **Computer Name**, and select the **Change** button).

Client configuration: Windows XP

Enter Windows XP, and the most complex beast of the lot. But first a word of warning for those unaware: Windows XP Home Edition *cannot* join a Windows domain. For domain functionality, you *must* use Windows XP Professional. Second, sometimes joining an XP machine to a Samba PDC involved all the steps below; on other occasions, however, you can get away with just the registry patch. Don't ask--I haven't a clue.

To join a Windows XP machine to a domain:

1. Open the Local Security Policy editor (**Start > All Programs > Administrative Tools > Local Security Policy**).

2. Locate the entry "Domain member: Digitally encrypt or sign secure channel (always)". Disable it.

3. Locate the entry "Domain member: Disable machine account password changes". Make sure it's disabled as well.

4. Locate the entry "Domain member: Require strong (Windows 2000 or later) session key". Disable it.

5. Next, download the WinXP_SignOrSeal registry patch from www.samba.org or collect it from the Resources section at the end of this

tutorial. Apply it by double-clicking and answering **Yes** to the dialog prompt.

6. Now join the domain the same as you would for Windows NT or 2000. Right-click **My Computer**, select **Properties**, **Computer Name**, and **Change**. Or click the **Network ID** button and run the Network Wizard.

Section 5. Troubleshooting and SWAT information

When things go bump in the night...

When Murphy crashes your party, it's time to do some troubleshooting. The list below is by no means complete or seminal. It's simply a compliation of years of working with Samba, and many many hours tracking user's cries for help on the Samba mailing list.

- First and foremost, determine where exactly your problem lies and how it occurred. Is it on the server end, or the client? Did it occur in conjunction with something else? Can you isolate the problem? Did you check your network cables? Are you SURE it's Samba related (this one in particular has bitten me several times), can you perform the same action without difficulty from another workstation? Are the Samba daemons running?
- If you start the Samba daemons and they unexpectedly die, run the Samba *testparm* utility on your configuration file. Chances are you have a syntax error somewhere. As a matter of fact, experience has taught me that the two most common errors that cause Samba to stop running or lose functionality are: (1) typos in `smb.conf`, and (2) incorrect permissions on a file or directory.
- The client can't save a profile to the PDC? Read the above again. And check your directory permissions.
- The client can't join the domain? Check to ensure user and machine accounts exist on the controller. If necessary, create them manually.
- If you try and join the domain, and get a "Cannot join domain..." or "Cannot create account, you already have a connection to the domain" message, check to ensure there are no existing mapped drives to the server. If there are, kill them by typing net use `* /d` in a command prompt window.
- If you can't join the domain, and you created the machine account manually, check to ensure you didn't forget to add the dollar-sign (`'$'`) after the machine name.

- If you can't join the domain, and are using the add `user script` option to automatically create machine accounts, double check the option. If nothing looks amiss, disable it, and manually create the machine account. Now try again.
- If all the above fails, go back through the tutorial and double check everything. Methodically. Again, the material presented here has been tripled checked. The `smb.conf` file was moved to a clean install of Redhat, the directories and permissions were created/set as shown, and the controller was tested with a Windows XP client. Everything worked first time without error or incident.
- Finally, failing everything else, send a message to the Samba mailing list asking for assistance. Don't forget to detail your problem clearly, what you've tried so far, and enclose your configuration file. The list is populated by a lot of very fine people; someone will no doubt come to your rescue.

Sidebar I: SWAT

A lot of administrators accustomed to GUI configuration tools find working with a command line editor like vi or emacs both intimidating and frustrating. Situations like this are precisely what SWAT was designed to address. SWAT stands for Samba Web Administration Tool, and is bundled with the Samba package. In short, SWAT puts an easy to navigate interface on `smb.conf` using any web browser. It also provides context-sensitive help for all options, and a link to the vast array of documentation shipped with Samba.

Unfortunately, convenience always comes at a cost. First and foremost, SWAT requires the root password to accomplish much of anything, and that password is transmitted in plain text. This dangerous security breach is offset somewhat by the fact that SSL/HTTPS can be used for remote connections (there's a HOWTO located at www.samba.org/samba/docs/swat_ssl.html). Second, SWAT has a habit of rearranging the order of entries in smb.conf when changes are saved. If you've got a carefully ordered configuration file complete with insightful comments, I do not recommend using SWAT.

Another difficulty with SWAT is that it's turned off by default and users unfamiliar with the landscape of Redhat seem to have a lot of trouble turning it on. This last problem is easy to fix.

Sidebar II: SWAT configuration

The are two ways to enable SWAT, depending on whether your system is configured to use *xinetd* or *inetd*.

For systems running xinetd (RH 7.2 and, I believe, 7.1), the script

/etc/xinetd.d/swat must be edited (as root). Change the line that reads disable = yes to disable = no. If you want to access SWAT from a remote machine (a very bad choice on anything but a firewalled intranet), place a pound sign ('#') in front of the line that reads only_from = localhost. Now re-start the xinetd daemon by typing service xinetd reload (again, as root). You should now be able to access the SWAT service directly from the local machine by typing httpd://localhost:901, or from a remote host by substituting localhost for the host name.

On systems running inetd, two files must be edited. Ensure that /etc/services contains the line:

```
swat 901/tcp
```

Next, open /etc/inetd.conf and locate the line that reads:

```

swat stream tcp nowait.400 root /path/to/the/swat/binary swat
```

Replace /path/to/the/swat/binary with the correct path. For example, /usr/sbin/swat. Now restart the inetd service: service inetd reload. Follow the procedures in the paragraph above to connect to your server.

Section 6. Wrap-up

Summary

This tutorial stepped through the process of configuring Samba to assume the role of primary domain controller on a local network. The following topics were discussed:

- A brief glance into the history and importance of the Samba project, what Samba can do, what it can't do, what hardware components are pivitol to good performance, and how to best utilize the material presented.
- How to install Samba from RPM or source, how to configure the build process to ensure directory and file placement, and how to configure smb.conf to be a PDC, support roaming profiles, and support netlogons.
- How to create the required administrative directories on the server, set the correct permissions on those directories, and how to create the two sets (UNIX and Samba) of user and machine accounts for authentication.
- How to configure Windows 95/98/ME/2000/XP clients to joint the domain.
- How to troubleshoot an installation that doesn't work as advertised.

- And finally, where to find further Samba resources.

I hope in the course of working through this tutorial you found what you were looking for, and --ideally -- some things you weren't. I know I sure did. And that, in the end, is what life's all about. Learning new things and pushing the boundries.

Resources

Learn

- Learn how Samba 2.2 improves on the already-excellent Samba 2.0.8 to create an incredibly powerful enterprise-ready Unix/Windows integration solution.
- Visit the Samba site for all online documentation, including man pages, FAQs, HOWTOs, and downloadable PDFs. There's also an announcements page, information on joining one of Samba's many mailing lists, information on getting involved with the development team, and usual file download facilities (CVS, FTP, and HTTP).
- The IBM Technology Center hosts a page containing many of the current patches that have been incorporated into the current Samba CVS/HEAD developer tree.
- Scott Merrill have an excellent HOWTO on simulating a PDC/BDC environment using Samba based servers.
- If you're interested in more details on Microsoft's SMB specs, try this link.
- IBM also has an excellent resource available in the form of their Redbook Series which can be read online, downloaded in PDF format, or purchased in book form. Highly recommended. As noted in section one of this tutorial, there is a Redbook titled Samba Installation, Configuration, and Sizing Guide (SG24-6004-00, published July 14, 2002) and another titled Implementing Linux in your Network using Samba. Both are slighted dated regarding Samba version and interoperability with Windows newer operatiing systems, but each contain several useful sections that do not change with time (for example, sizing recommenations and general network considerations).
- "Dead Tree" Resources:
  - As noted several times through this tutorial, the seminal title on Samba is Using Samba (O'Reilly & Associates) by Eckstein, Collier-Brown, and Kelly. It's getting a bit dated, but the core material is still as current as the day the book was written. If you can have only one Samba book on your shelf, make it this one. Note: While some distributions do not include it, all packages downloaded from Samba.org come with the full text of this title in HTML; look under /usr/share/swat/using_samba/.
  - Another good title to have around is Special Edition: Using Samba (QUE) by Sharpe, Turner, and Potter. In particular, check out some of the chapters in the "Advanced Topics" section of the book.
- Want more? The developerWorks eServer™ zone hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials on the eServer brand.
- The IBM developerWorks team hosts hundreds of technical briefings around the world which you can attend at no charge.

Get products and technologies

- Download the latest Samba source code (as of this writing, 2.2.3a).

About the author

Tom Syroid

Tom Syroid is a contract writer for Studio B Productions, a literary agency based in Indianapolis, IN specializing in computer-oriented publications. Topics of interest/specialty include *NIX system security, Samba, Apache, and Web database applications based on PHP and MySQL. He has experience administering and maintaining a diverse range of operating systems including Linux (Red Hat, OpenLinux, Mandrake, Slackware, Gentoo), Windows (95, 98, NT, 2000, and XP), and AIX (4.3.3 and 5.1). He is also the co-author of *Outlook 2000 in a Nutshell* (O'Reilly & Associates) and *OpenLinux Secrets* (Hungry Minds). Tom lives in Saskatoon, Saskatchewan with his wife and two children. Hobbies include breaking perfectly good computer installations and then figuring out how to fix them, gardening, reading, and building complex structures out of Lego with his kids. Questions, comments, and errata submissions are welcome; you can either e-mail the author directly (dwcomments@syroidmanor.com.