Learn Linux, 302 (Mixed environments): Configure Samba

Set up Samba for a variety of purposes

Sean A. Walberg
Senior Network Engineer

27 April 2011

Samba uses a human-readable file to manage and store its configuration parameters, so the most sophisticated tool you'll need to configure Samba is a text editor. Learn how the configuration file is structured, how Samba interacts with the network, how to configure logging, and how to debug problems with Samba.

View more content in this series

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for the Linux Professional Institute Certification level 3 (LPIC-3) exams.

See our developerWorks roadmap for LPIC-3 for a description of and link to each article in this series. The roadmap is in progress and reflects the *current objectives (November 2010)* for the LPIC-3 exams. As each article is completed, it is added to the roadmap.

In this article, learn to:

- Navigate the Samba server configuration file structure
- Use Samba variables and configuration parameters
- Identify key TCP/User Datagram Protocol (UDP) ports used with Server Message Block (SMB)/Common Internet File System (CIFS)
- Configure Samba logging
- Troubleshoot and debug problems with Samba

This article helps you prepare for Objective 312.1 in Topic 312 of the Linux Professional Institute's Mixed Environment Specialty exam (302). The objective has a weight of 6.

Prerequisites

To get the most from the articles in this series, you should have an advanced knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. In

addition, you should have access to a Windows environment that you can use to test file and print access.

The Samba configuration file

About the elective LPI-302 exam

Linux Professional Institute Certification (LPIC) is like many other certifications in that different levels are offered, with each level requiring more knowledge and experience than the previous one. The LPI-302 exam is an elective specialty exam in the third level of the LPIC hierarchy and requires an advanced level of Linux system administration knowledge.

To get your LPIC level 3 (LPIC-3) certification, you must pass the two first-level exams (101 and 102), the two second-level exams (201 and 202), and the LPIC-3 core exam (301). After you have achieved this level, you can take the elective specialty exams, such as LPI-302.

Samba, like most UNIX daemons, is configured through text files that are human readable, rather than through a graphical tool for binary file editing. The most important configuration file is called *smb.conf,* which contains all the parameters that Samba needs to run in your environment.

Note: Although smb.conf was designed to be edited with a text editor, the Samba team has come up with a web-based tool called the *Samba Web Administration Tool*. Alternatives, such as webmin, also exist. It is important to remember that you can still edit smb.conf before or after these tools have been run, because you're operating on a text file.

Samba's configuration file has a fairly simple format that uses three different constructs:

- **Sections.** Sections group the configuration into independent areas. For example, a file share has its own section.
- **Parameters.** Parameters are key-value pairs. The keys are well-known attributes such as "read only."
- **Comments.** Comments let you make annotations to the configuration file that don't affect the configuration, such as to indicate the help desk ticket that documents the share.

Sections

Build your own feed

You can build a custom RSS, Atom, or HTML feed so you will be notified as we add new articles or update content. Go to developerWorks RSS feeds. Select **Linux** for the zone and **Articles** for the type, and type `Linux Professional Institute` for the keywords. Then, choose your preferred feed type.

Sections group the configuration file into different areas. You start a section by enclosing the name of the section in square brackets (`[]`). The section continues until the next section is defined or the end of the file is reached.

Three section names have special meaning:

- **`global`.** Anything in this section applies to the whole server. Configuration items in the `global` section can be overridden at the share level, if needed.

- **homes.** The `homes` section serves as a template for all user shares, and Samba takes care of mapping a user name to the configuration in this section, virtually eliminating the need to configure a separate share each time you want to let a user into his or her home directory.
- **printers.** This section is similar to `homes`, except that it's used for printers.

If the section name used is not one of the above, it is considered a file or printer share.

When a connection request comes in to Samba for a particular share name, the daemon looks for a section with that name that would define the properties of that share. If the section is not found, Samba looks through the list of users on the system to see whether the connection refers to a user. If not, Samba consults the list of system printers to see if a printer by that name exists. If the connection matched a user, the configuration for the `homes` sections is used. If the printer was matched, the `printers` section is used. In all cases, the section-level configuration overrides the `global` configuration section.

If none of the above cases is matched, there is one final check. If a default service is configured, that service is used. If not, an error is passed back to the client. By default, the default service is not configured, so incorrect share names will result in an error.

Parameters

Parameters take the form `key = value`, which assigns the *value* to the *key.* The keys are all documented in the smb.conf man page. Samba configuration is largely an exercise in understanding which keys are needed to achieve the behavior you want and determining the appropriate value to use.

Parameters generally take strings as values. Samba does support macros, which allow you to vary the value of the parameter according to items like the share name or input from the user. For example, the `homes` section defaults to the user's UNIX home directory, but you can use macros to make this parameter use any location and substitute the user name into the file path at the time of connection. Macros start with the `%` character and will be discussed as they are required.

If a parameter's value must extend over two or more lines, all lines except the last must end in a backslash (`\`), just like a UNIX shell.

Comments

Comments begin with a semicolon (`;`) or octothorpe (hash, or `#`). Comments can be used to explain the reason for items, to track changes, or to indicate section boundaries.

An example configuration

Listing 1 shows an example smb.conf file, illustrating the different pieces of the file.

Listing 1. An example configuration file

```
# This is a comment
; So is this
# Remember that all shares need to be entered in the Wiki! -Opsteam
[global]
  workgroup = BIGCO
```

```
  # %v gets expanded to the version of Samba
  server string = Samba Server Version %v
  # By default any file starting with . will have the hidden attribute set
  hide dot files = yes

# Home directories come from the UNIX password file
# anyone matching a user will use this section
[homes]
  comment = Home directories
  # dot files will be hidden because it's a global

[printers]
  comment = System printers
  printable = yes

# A share that everyone can see
[projecta]
  path = /var/spool/projects/projecta
  # Override the global version of hiding dot files
  hide dot files = no
```

Some special things to note about this example configuration are:

- Two different styles of comments are used. One begins with a hash, the other begins with a semicolon.
- This file defines one share called *projecta.* Any other shares will be automatically created from the users and printers defined on the system.
- The `server string` parameter uses the `%v` macro for part of its value. At run time, `%v` will be replaced with the version of Samba.
- `hide dot files` is set to yes at the global level but set to no inside the projecta share. Home directories use the configuration from the `homes` section and so will have their dot files (such as .profile) hidden. Files in projecta will have their dot files visible.

Samba network interaction

Samba is a network service that runs over IP, which allows it to communicate with other hosts on the network that are also using IP. As a Samba administrator, you need to understand how the Samba services behave on the network to troubleshoot connectivity problems.

At a high level, you can look at Samba as offering three different kinds of network services:

- **File and print sharing.** Offering files and printers to other network devices and using those services on other machines
- **Name services.** Name resolution services necessary to participate in a Microsoft network
- **Domain services.** Samba can replace various Microsoft server roles such as a legacy domain controller and integrates with newer Active Directory Domain Services (AD DS) servers

File and print sharing

File and print sharing are implemented within `smbd`, which is one of the Samba daemons. Microsoft file sharing used network basic input/output system (NetBIOS) over TCP when it first moved into the IP world. This method encapsulated NetBIOS content inside a TCP session using TCP port 139.

The NetBIOS protocol encompasses several features. TCP port 139 is used only for the session services, which are file transfer and message passing. The name lookup services are not handled on this port.

NetBIOS over TCP works, but there is overlap between the session and reliability features provided by NetBIOS and those provided by TCP. With some subtle changes, it became possible to run SMB/CIFS right on top of TCP. This method is known as *direct hosting* and is used to simplify the protocol. Direct hosting occurs on TCP port 445.

When NetBIOS was eliminated from the protocol suite, Microsoft needed another way to handle name lookups. Domain Name System (DNS) was a natural choice, which is why DNS forms the basis of AD DS.

By default, Samba listens on ports 139 and 445. You can change this behavior with the `smb ports` global parameter. For example, `smb ports = 445` tells Samba to listen only on port 445. You can have Samba listen on any port you want, although any client wanting to connect would have to be told to use the nonstandard port.

If you are unsure which ports Samba is listening on, you can use the `netstat` command to find out. Listing 2 shows this command in action.

Listing 2. Using netstat to find which ports SMB is listening on

```
# netstat -antp | grep smbd
# netstat -antp | grep smb
tcp   0  0 :::445                    :::*                        LISTEN      2830/smbd
tcp   0  0 ::ffff:192.168.1.143:445  ::ffff:192.168.1.147:4724  ESTABLISHED 2877/smbd
```

Listing 2 shows the `netstat` command being run, with the output being filtered through `grep` for the string `smb`. The `netstat` options used show all (`-a`) TCP (`-t`) connections in a numeric (`-n`) format along with the name of the process (`-p`) responsible. This output shows two lines: The first contains the string `LISTEN`, which means that the daemon is listening for incoming connections. Here, the daemon is listening on port 445. The second line shows an `ESTABLISHED` connection, where 192.168.1.147 is connected to port 445 on the local host (192.168.1.143). Thus, from the output in Listing 2, you can conclude that `smbd` is only listening to port 445 and that one client is connected.

Name services

NetBIOS provides a name services layer responsible for network browsing and name lookups. For example, the host SERVER1 is resolved to an IP address by using NetBIOS name service requests on UDP port 137. Support for browsing and election of support roles such as the master browser happens on UDP port 138, otherwise known as the *datagram services port.* Name services are implemented within the `nmbd` daemon.

It is important to note that the name services use UDP instead of TCP. UDP packets are connectionless and can be broadcast to all hosts instead of a single unicast stream. With UDP's broadcast functionality, NetBIOS name services can be made easier to process on the network.

Samba version 3 does not have any parameters to control which ports `nmbd` listens on, but Samba version 4 implements the `nbt port` and `dgram port` global parameters, which control the name service and datagram services ports, respectively.

You can use something similar to what you saw in Listing 2 to show which ports `nmbd` opens. This is shown in Listing 3.

Listing 3. Showing the ports that nmbd is listening on

```
# netstat -anup | grep nmbd
udp        0      0 192.168.1.255:137       0.0.0.0:*        2975/nmbd
udp        0      0 192.168.1.143:137       0.0.0.0:*        2975/nmbd
udp        0      0 0.0.0.0:137             0.0.0.0:*        2975/nmbd
udp        0      0 192.168.1.255:138       0.0.0.0:*        2975/nmbd
udp        0      0 192.168.1.143:138       0.0.0.0:*        2975/nmbd
udp        0      0 0.0.0.0:138             0.0.0.0:*        2975/nmbd
```

Besides looking for `nmbd` instead of `smbd`, the command in Listing 3 looks for UDP ports instead of TCP ports with `netstat`'s `-u` option. The results are a series of lines showing that `nmbd` is listening on ports 137 and 138 on a variety of interfaces along with listening on the broadcast address of 192.168.1.255. Both of the name services ports rely on host-to-host communication and broadcast communication.

Domain services

The Samba team is constantly updating the software to make it integrate more closely with Microsoft networks and to replace Microsoft infrastructure. To do so, Samba must emulate these infrastructure services on the network.

Most of these services involve Kerberos and the Lightweight Directory Access Protocol (LDAP) in some way. These are advanced topics and will be covered in more depth in later articles. For the moment, just be aware that Samba can do more than just file sharing.

Summary of the ports that Samba uses

Table 1 provides a summary of the file sharing-related ports on which Samba daemons listen.

Table 1. Summary of ports used in Samba

Port	Protocol	Service	Daemon	Description
137	UDP	`netbios-ns`	`nmbd`	NetBIOS name services
138	UDP	`netbios-dgm`	`nmbd`	NetBIOS datagram services
139	TCP	`netbios-ssn`	`smbd`	NetBIOS over TCP (session services)
445	TCP	`microsoft-ds`	`smbd`	Direct-hosted SMB

The tag in the **Service** column is the well-known service name, which comes from a file called */etc/services*. The services file helps applications resolve service names to port numbers. The file also helps humans relate port numbers to service names. Even though most services reserve

both TCP and UDP ports, applications do not have to use both UDP and TCP. Reserving both eliminates possible confusion when two different services try to use the same port number on different protocols.

It is also worth noting that the numbers and names in /etc/services are not enforced. You can run daemons on different ports other than what is expected as long as you can communicate this change to the clients. For example, you could run Samba on ports 5137 through 5139 and 5445 as long as your clients weren't expecting to talk on the standard ports.

Troubleshooting Samba problems

Samba is not immune to problems. Sometimes, these problems are caused by the system administrator; sometimes they're caused by the user. Your job as the system administrator is to figure out where the problem lies, then how to solve it.

Testing the configuration file

If Samba won't start or you want to check your configuration file for correctness, then the `testparm` utility will help. This utility checks smb.conf for correctness. Listing 4 shows the result of `testparm` if there is an error.

Listing 4. Using testparm on an incorrect smb.conf file

```
# testparm
Load smb config files from /etc/samba/smb.conf
Unknown parameter encountered: "hide dto files"
Ignoring unknown parameter "hide dto files"
Processing section "[homes]"
Processing section "[printers]"
Processing section "[public]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

[global]
        workgroup = MYGROUP
        server string = Samba Server Version %v
        passdb backend = tdbsam
        log file = /var/log/samba/log.%m
        max log size = 50
        cups options = raw

<< rest of the output omitted >>
```

The output of `testparm` starts with the location of the files. If you want to specify a different file, pass the name of the file on the command line, as in this example:

```
testparm /home/me/smb.conf
```

Next, `testparm` is complaining about an invalid parameter called `hide dto files`. This parameter should really be *hide **dot** files.*

After processing the configuration file, you are given some information about the server's role and a condensed version of the configuration file. This version has the comments stripped out and is

consistently formatted, so you will sometimes catch errors here that you missed while browsing smb.conf in a text editor.

You should run `testparm` on your configuration file after making changes. Samba ignores most typographical errors in the configuration files and doesn't always write messages to the console when starting up. It's likely that you won't catch these types of mistakes until something isn't working properly. `Testparm` alerts you to any typos in smb.conf.

By default, `testparm` only shows you the configuration as entered in smb.conf. If you suspect that you are using a default value somewhere, you can use the `-v` option to force `testparm` also to show default values.

Another use of `testparm` is to limit the output to a single section or parameter. Listing 5 shows how to use `testparm` to see the value of the `security mask` option.

Listing 5. Limiting testparm to a single parameter

```
# testparm -s --parameter-name "security mask"
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[public]"
Loaded services file OK.
0777
```

In Listing 5, the `-s` parameter stops `testparm` from waiting for user input in between parsing smb.conf and displaying it to the screen. Using `--parameter name "security mask"` asks for the value of `security mask`. The result is `0777`, which is the default value. In this mode, it is not necessary to specify `-v` to show default values.

Connecting as a client

In lieu of going to a user's desktop and trying things out yourself, you can do a great deal of testing from the command line at your own desk. The first and easiest test is to make sure that you can connect to the Samba port. The easiest way to do this is with the `telnet` command, which is shown in Listing 6.

Listing 6. Testing connectivity with telnet

```
# telnet bob 139
Trying 192.168.1.134...
telnet: connect to address 192.168.1.134: Connection refused
```

In Listing 6, the root user is connecting to the server *bob* on port 139. You could also use port 445 to test for the direct-hosted SMB port. The result is `Connection refused`, which indicates either that the daemon isn't listening on that address or that a firewall is blocking the connection. Other results, such as `No route to host` or `Connection timed out`, can mean the same thing.

Clients generally connect to a server with a name, not an IP address. If you use `telnet` to connect to the server by name rather than IP address, pay special attention to the IP address returned. In

the example above, the server (bob) was resolved to 192.168.1.134. Sometimes, you can have errors in your DNS records that result in the clients connecting to the wrong address.

If you are not using DNS for Windows name resolution, you can use the `nmblookup` command to perform a NetBIOS name lookup. Listing 7 shows a query for the bob server.

Listing 7. Performing a NetBIOS name query for bob

```
# nmblookup bob
querying bob on 192.168.1.255
192.168.1.138 bob<00>
```

According to Listing 7, the server bob is at 192.168.1.138, not 192.168.1.134, as you saw in Listing 6. This result points to a problem with DNS, especially if ports 139 and 445 are responsive on 192.168.1.138.

Another test is to see whether the configuration file denies access to a particular host. `Testparm` is used again in Listing 8.

Listing 8. Checking access with testparm

```
# testparm /etc/samba/smb.conf  seanspc 192.168.1.147
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[public]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Deny connection from seanspc (192.168.1.147) to homes
Deny connection from seanspc (192.168.1.147) to printers
Deny connection from seanspc (192.168.1.147) to public
```

Firewalls or application?

There are many ways to block a connection to a host, but all the options can be grouped into two categories: network and application. By blocking at the network layer, either on a firewall or using a host-based firewall like iptables, you will see that the `telnet` connection shown in Listing 6 is refused or times out. This is because the packet never makes it to the Samba application.

If Samba is configured not to allow connections from a particular host, you will see that the `telnet` connection succeeds, but any client access is greeted with an error. This is because the packet is read by the application but doesn't like the IP address or hostname and sends an application-level error. Without accepting the packet at the application layer, Samba can't know whether the IP address is acceptable.

In Listing 8, three items are passed to `testparm`:

- The path to the Samba configuration file
- The NetBIOS name of the machine to test
- The IP address of the machine to test

The output from Listing 8 shows that the machine in question is denied access to all shares. When using `testparm` in this mode, the utility does not actually connect as that machine. Instead, `testparm` processes the configuration file to see whether the access would be allowed.

If all the tests up to now succeed, you can attempt to make a client connection using the `smbclient` utility. The first test is to attempt to browse the share list, which is shown in Listing 9.

Listing 9. Showing a machine's shares

```
[sean@bob source3]$ smbclient -L '\\bob'
Enter sean's password:
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-69.fc13]

        Sharename       Type      Comment
        ---------       ----      -------
        extdrive        Disk
        Sean Walberg's iMac Disk
        timemachine     Disk
        IPC$            IPC       IPC Service (Samba Server Version 3.5.6-69.fc13)
        test            Printer   test
        Downstairs_Laser Printer   HP 6L
        Cups-PDF        Printer   Cups-PDF
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-69.fc13]

        Server               Comment
        ---------            -------
        BOB                  Samba Server Version 3.5.6-69.fc13

        Workgroup            Master
        ---------            -------
        MYGROUP              BOB
        WORK                 SWALBERG-XPLT
        WORKGROUP            IMAC-1FC525
```

In Listing 9, the user is requesting a list of shares with the `-L` parameter on the server called *bob.* The server name is prefixed with two backslashes (`\\`), because it is a Universal Naming Convention (UNC) path. Be careful about your choice of single versus double quotation marks, as well. Single quotation marks interpolate and see the backslashes as escape characters.

If your server has more security set up, you may need to pass the name of the user or the domain with the `-W` and `-U` parameters, respectively.

Finally, you can try to connect to a share by omitting the `-L` parameter and specifying a full UNC path to the share. Listing 10 shows the client connecting to a server using a different workgroup and user name.

Listing 10. Connecting to a share with a different user name and domain

```
[sean@bob source3]$ smbclient '\\swalberg-xplt\photos' -U swalberg -W WORK
Enter swalberg's password:
Domain=[WORK] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \> dir
  .                                   D        0  Thu Jan  6 11:39:50 2011
  ..                                  D        0  Thu Jan  6 11:39:50 2011
<< files omitted >>
                38156 blocks of size 4194304. 2938 blocks available
smb: \>>
```

If these tests pass, you can be reasonably certain that the problem is not with the Samba configuration but somewhere between the client and the server or something on the client itself. You look at the logs in the next section, which offer some clues as to where the problem lies.

Logging and debugging

Logging and troubleshooting go hand in hand. Logs let you look back in time to see whether errors occurred and get more details about problems as they occur. If you're trying to figure out why something isn't working, you can increase the amount of logging until you get the necessary level of detail. Samba offers a handful of parameters in smb.conf to handle logging; you can use these parameters together to tailor the types of logs generated.

Samba acts like a traditional UNIX daemon in that it can log to the `syslog` facility as well as generate its own log files. Furthermore, the Microsoft Event Viewer tool can connect to a Samba server to pull logs. The catch with this latter feature is that Samba cannot log directly to the event log format: You must post-process your log files using Samba tools.

Log levels

Each log message that Samba generates has a *level,* which is an integer from 0 to 10. Higher-level messages, such as new connections and important errors, have lower levels. Debugging messages have higher numbers. You control the amount of logging by specifying the maximum level you want logged. A log level of 1 only logs messages with a priority of 0 or 1. If you want extensive logging, use a higher number.

Anything above a log level of 3 is meant for a developer's eyes and won't provide much help to you as a system administrator. Using a log level of 0 turns off all but a few startup messages and critical errors.

To configure the log level, use the `log level` parameter in the `global` section; for example, use `log level = 2` to set the logging level to 2. This setting logs any messages with a priority of two or less.

Exam note

The LPIC exam outline specifically mentions the `debuglevel` parameter as something important to know. `debuglevel` is a synonym for *log level,* and the two can be used interchangeably.

You can change the log level at run time by sending a `SIGUSR1` signal to the Samba process to increase the log level or `SIGUSR2` to decrease the log level.

You can also be more granular with your log levels, increasing verbosity only in certain features by specifying which class you want to log. These classes are:

- **`all`.** This parameter is optional: If you don't specify extra keywords, `all` is assumed.
- **`tdb`.** Log messages relating to trivial databases, which are key-value stores that Samba uses.
- **`printdrivers`.** Printer driver management routines.

- **`lanman`.** NT LAN Manager debugging.
- **`smb`.** SMB protocol debugging.
- **`rpc_parse`.** Parsing of remote procedure calls (RPCs).
- **`rpc_srv`.** The server-side RPCs.
- **`rpc_cli`.** The client-side RPCs.
- **`passdb`.** The older way of storing passwords on a Samba host.
- **`sam`.** The local Samba account database.
- **`auth`.** Various modules inside Samba related to authenticating users.
- **`winbind`.** The component used to allow Microsoft users to transparently log in to UNIX systems.
- **`vfs`.** Debugging messages for the Virtual File System modules, which let you add functionality to Samba through pluggable modules.
- **`idmap`.** Mapping of identities between UNIX user IDs and Microsoft security identifiers.
- **`quota`.** Messages relating to quota processing, both by Microsoft Windows NT policy and the UNIX file system.
- **`acls`.** Access control list processing.
- **`locking`.** File locking status and errors.
- **`msdfs`.** Log messages relating to Samba's distributed file system support.
- **`dmapi`.** Data management application programming interface (API) functionality. Samba must be compiled with a third-party DMAPI implementation to use this feature.
- **`registry`.** Emulation of the Windows registry.

To use this additional logging, append the keyword and the value to the log level parameter, separated by a colon (`:`). For example, `log level = 1 winbind:3` sets the system default log level to 1 and increases winbind logging to 3. This change helps you debug problems with single sign-on without drowning yourself in unrelated log files.

Log file location

To change the name of the log file, use the `log file` parameter. You can also use macros in the value. One often-used setting is to have one log file per client. To do this, specify:

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

This command separates the individual log files into one per client, with the rest of the messages still going to log.smbd.

If you want to log to `syslog`, you can specify `syslog = 1` to send all logs at level 1 or 0 to the local syslog server. Samba uses the `LOG_DAEMON` facility and maps Samba log levels to `syslog` priorities as follows:

- **`LOG_ERR`.** Log level 0
- **`LOG_WARNING`.** Log level 1
- **`LOG_NOTICE`.** Log level 2
- **`LOG_INFO`.** Log level 3
- **`LOG_DEBUG`.** Log level 4 or greater

If you are syslogging to a more advanced `syslog` daemon that can filter incoming messages and alert the system administrator, this is an excellent way to keep an eye on your Samba server.

Log metadata

You can add or remove some pieces of information that show up in all the log entries with more global parameters:

- **`debug timestamp`.** Adds a timestamp to the log message and is enabled by default
- **`debug uid`.** Logs the user and group IDs of the Samba process generating the logs
- **`debug prefix timestamp`.** Keeps timestamps on but removes the information about the place in the Samba source code that generated the log
- **`debug pid`.** Logs the process identifier of the Samba process that generated the log
- **`debug hires timestamp`.** Changes the timestamp resolution to microseconds instead of seconds
- **`debug class`.** Logs the class of the log message, which is helpful if you want to change the verbosity of a certain class (This option helps you determine which class you want.)

Logging can help you find problems, or it can drown you in noise. Samba offers a variety of logging options; use them sparingly.

System call tracing

If all else fails, you can use UNIX system tools to look at what's going on inside the process. The Linux `strace` program lets you trace all the system calls that an application makes. An application uses system calls to open and read from files, create and destroy processes, and interact with the rest of the operating system.

Listing 11 shows the root user tracing a Samba process that is throwing an error to the client.

Listing 11. Stracing a process

```
# ps -ef | grep smb
sean     13375 28812  0 21:54 ?        00:00:00 smbd -D
root     14294 13593  0 21:55 pts/2    00:00:00 grep smb
root     16132 28812  0 Feb27 ?        00:00:36 smbd -D
root     28812     1  0 Feb14 ?        00:00:28 smbd -D
root     28814 28812  0 Feb14 ?        00:00:00 smbd -D
[root@bob /]# strace -e trace=file -p 13375
Process 13375 attached - interrupt to quit
<< Output omitted >>
chdir("/home/sean")                      = 0
stat64("somedir", {st_mode=S_IFDIR|0700, st_size=4096, ...}) = 0
stat64("somedir/*", 0xbfcb5f60)          = -1 EACCES (Permission denied)
getcwd("/home/sean", 4096)               = 11
lstat64("/home/sean/somedir", {st_mode=S_IFDIR|0700, st_size=4096, ...}) = 0
lstat64("/home/sean/somedir/*", 0xbfcb5ffc) = -1 EACCES (Permission denied)
```

The first command looks for a list of the Samba processes. As Samba assumes the identity of the connected user, process 13375 can easily be identified as the one belonging to the user. Next, the `strace` command is run with two parameters: `-e trace=file` limits the output to system calls

having to do with files. For the types of problems that you will encounter with Samba, this is a good initial guess. The second parameter, `-p 13375`, tells `strace` to attach to the running process with that process ID.

If you watch the output of this command, you will see that `smb` is constantly scanning directories for changes. When the user tries the action that has the problem, you might see some output like that in Listing 11. The final few commands try to get information about files inside the directory with the `stat64` call. The result is `permission denied`, which means that the user is being rejected at the file system level, not by Samba. This command can give you more information with which to solve the problem, such as changing the attributes of the directory or telling the user he or she isn't allowed to access the directory.

Moving forward

This is the end of the Samba configuration topic. The next exam objective, 312.2, will have you creating and configuring file shares and learning how to access these shares from other systems.

Resources

Learn

- Logging user activity can be done through Samba's Virtual File System layer.
- The smb.conf man page web version is more convenient than the command-line version.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth of other resources for Linux developers and administrators.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.

Get products and technologies

- In the Samba Git repository, find out what's going on with Samba.
- Get the latest Samba version so that you're up to date with the latest features.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, or use a product in a cloud environment.

Discuss

- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Sean A. Walberg

Sean Walberg is a network engineer and the author of two books on networking. He has worked in several industries, including health care and media.