Samba Share Configuration

Looking through Windows

The view a Windows user has of Samba is defined by various parameters for shares, file permissions and locking. And this is exactly what we will be investigating in this article: Who is allowed to access what directories, who can view or modify what? BY THOMAS DRILLING AND MICHAEL MIELEWCZIK

Linux and Windows use completely different filesystems and fundamentally diverse approaches to permission management. This makes mapping directories and files belonging to one filesystem on the other a complicated task, governed by a multitude of parameters.

You will not always need Samba to reflect SMB based functionality in all respects. Often it is simply a case of emulating the response of a Windows server to an incoming SMB client request without really doing anything spectacular on the Linux side of the equation.

Locking is a particularly delicate issue that affects access to shared files and directories and involves Samba mapping typical Windows behavior to the technical facets of the Linux filesystem. File and directory names, and how they are encrypted for transmission, is another equally complex topic that includes the major difference between the operating systems with respect to case sensitivity and the length of filenames.

Parameters and Shares

The ubiquitous Samba configuration file, "/etc/smb.conf", is also used for setting up shares. The names of any shares that Samba needs to manage are reflected by entries here.

The "[homes]" and "[printers]" sections of "/etc/smb.conf" are particularly significant when creating new shares. They must occur at a position between the pre-configured "[global]" section and any other sections in "smb.conf". "[homes]" is a special Samba share that allows users to access their own home directories. Any further sections used to defined shares start with the share name surrounded by square brackets.

Hannes Keller, visipix.com

Basic Access Control

The most important parameters are "path" and/or "directory" as they specify the directory to be shared by Samba:

```
[executive_photos]
  path = /home/samba/boss/images
```

The following parameter is used to display shares in the client's network neighborhood:

```
comment = pictures of the boss
```

This comment appears next to the share name. You can use the following parameter to display the name of a specific directory, but prevent access to it:

```
dont descent = /too_embarrassing
```

"veto files" defines a list of files and directories that clients will not be able to view or modify. The entries can contain meta tags such as "*" or "?". Multiple entries are separated by the "/" character:

```
veto files = /*.tmp/*.conf
```

Further attention is required to define directories as veto files. The "browseable = yes" parameter makes the share visible in the browse list requested by the client. The command line tool, "net view", or the network neighborhood can be used to view shares. The "browseable = no" setting removes the share from the browse list, but it will not prevent access to shares whose names are known.

DOS/Windows and Unix are completely different in their handling of access permissions and file attributes. Samba has to act as a proxy here to allow any access whatsoever. This does not mean that Samba introduces its own access mechanisms; instead it relies on the access control features provided by the Unix system: that is, DOS attributes need to be mapped to Unix file permissions.

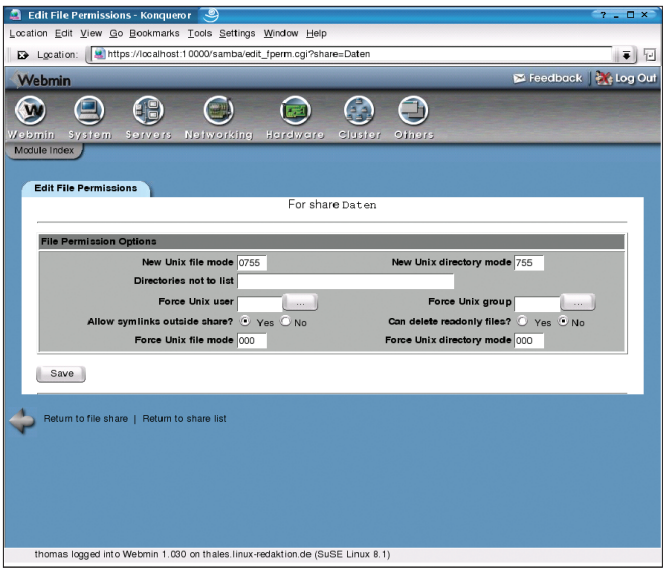

Figure 1: Access privileges for a share are the logical product of the "create mask", "directory mask" and "force directory mask" entries

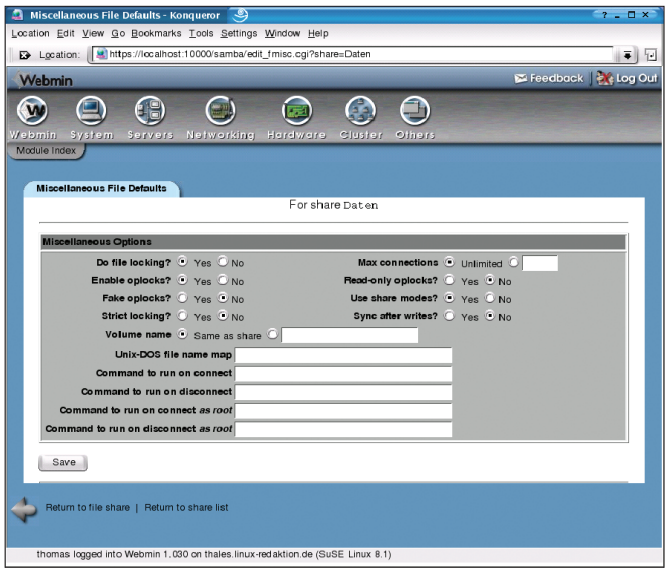

Figure 2: Webmin also allows you to define most locking parameters. In this case, the Webmin fieldnames actually correspond to the parameter names

Three statements are used for this purpose: "map", "create mask" (or "create mode") and "force create mode". "map" uses the executable bit "x" for Linux files, which does not have a counterpart under DOS/Windows, to store three DOS file attributes that are not available on Linux. For example, the following entries:

```
map archive = yes
map system = yes
map hidden = yes
```

allow Samba to assign the DOS archive bit to the owner's executable bit, the DOS system bit to the group executable bit, and the DOS hidden bit to the world executable bit. The "create mask" argument is a bit pattern of the type used by the Unix "chmod" command. The default value is "0744", which means that group members and all other users have read-only access to the owner's files. The following statement:

```
create mask = 0740
```

would prevent universal readability. However, the parameter only specifies the maximum privileges that a file can have; if the user or the Windows program specify reduced privileges, this request will be honored. The "force create mode" can be used to enforce mandatory attributes. The following statement:

```
force create 02
```

stipulates "rw-rw—-" for example, if the above mentioned example originally stipulated "rw-r—--". These parameters apply to simple files; "create directory" and "force directory" with corresponding parameters are used for directories. The default value for directories is "755", as access is impossible without the executable bit. "read only = yes" will make complete shares read only.

The "guest ok" parameter (or alternatively "public") allows guests to access this share via their clients. The permissions granted to the guest account are defined by the "guest account" parameter. The "ftp" user is often used to assign guest privileges for client PCs. The entry in the share section must appear as follows:

```
guest account = ftp
```

The default setting for guest access is the user "nobody".

Logon and Permissions

If the share configuration entry specifies:

```
allow trusted domains = yes
```

the share can only be accessed by clients belonging to the same domain. If the share configuration entry specifies:

```
available = no
```

client is not permitted to the share. The administrator can use this option to specify who will have access to the share.

The "deny hosts" option allows you to specify hosts that will be denied access to the share. This assumes that the host is not additionally listed in "hosts allow" as this entry has priority:

```
deny hosts = nutcase, spy
```

The IP address, network group and/or network group/netmask can also be specified as parameters. Group names must be preceded by a "@" character. The following entry:

```
deny hosts = @interns
```

would deny share access to any hosts in the NIS "interns" group.

```
deny hosts = 192.168.1.0/255.➥
255.255
```

would deny access to any clients with IP addresses in the range 192.168.1.0 through 192.168.1.255. The parameters for "allow hosts" follow the same pattern.

However, user based access control makes more sense than a host based equivalent. "valid users" is the best option for admins that only need to provide access to a few users. The following statement:

```
valid users = bigboss, pr, mom
```

would restrict access to the directory with the bosses pictures to the named users. Again, you can specify group access by prepending an "@" character. In a similar way, "invalid users" will deny users. The "read list" allows you to define a list of clients that will receive read-only access to a Samba share which is writable by default:

```
read list = @marketing, michael
```

A share which is read-only by default can be made writable for users by adding them to the "write list".

Possible insecurities in the Samba configuration can be removed by stipulating "follow symlinks = no". Samba will follow symbolic links by default, even if they point to sources outside the share. You can alternatively stipulate "wide links = no" in combination with "follow symlinks = yes". This will prevent Samba from following links outside the share.

Locking Parameters

The handling of concurrent access to a file is one of the most difficult tasks for any network operating system. The "blocking locks" is a critical, albeit very specific option that allows you to influence the Samba daemon's behavior. The status of the option becomes apparent when a client stipulates "byte range locking" on requesting an open file from a shared directory and when a time limit has additionally been set.

If "blocking locks" is enabled (default) the requested lock is queued and the request automatically repeated until it is successful or a timeout occurs. If the option is disabled, the request is denied if the lock range cannot be released:

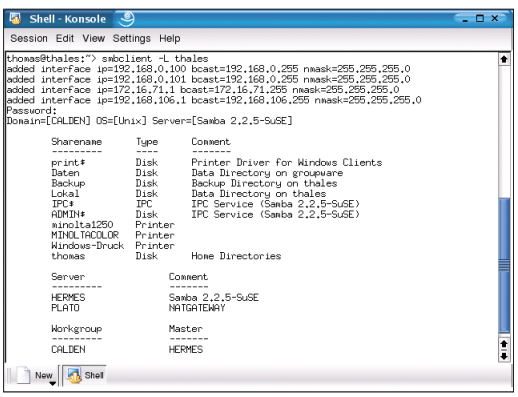

Figure 3: The Samba "smbclient" command can be used to quickly view existing shares

```
blocking locks = false
```

The "true" option would enable blocking locks for a share.

The "oplocks" option enhances the Samba servers performance and is enabled by default. If you are experiencing difficulty in your network environment, you might like to disable this option by specifying:

```
oplocks = false
```

The performance boost is achieved by client side caching of data and commands. Clients that do not respond to an "oplock break" request are a special case for the Samba server. You can set the "oplock break wait time" to tell Samba to wait for a number of milliseconds before sending an "oplock break" request to a client for this share. The corresponding entry would appear as follows:

```
oplock break wait time = 10
```

If there is a danger of too many oplock requests occurring for a specific file on a Samba share in close succession, you can use the "oplock contention limit" option to set a limit for the number of requests that will receive a positive response.

The "level2 oplocks" option became available in Samba version 2.2.1, and has been part of Samba ever since. This option allows an NT client that possesses a read-write oplock for a file to demote itself to a "read-only oplock" status, when a second user requests access to the file. This can improve performance for some file access operations as it avoids reissuing an oplock request.

The following statement is used to enable Level 2 oplocks for a share:

```
Level2 oplocks = true
```

Level 2 oplocks assume that "oplocks" are enabled and kernel oplocks disabled.

The "strict locking" option allows you to influence the file locking behavior of a Samba server for a Samba share. If the option is enabled, the "file locking" status is checked whenever read or write access is requested. If locks are in place, the request is denied automatically.

If the option is disabled, the client must explicitly request a file locking status check:

```
strict locking = yes
```

enables the option. If the general oplock functionality for a share is enabled, you can use "veto oplock files" to mark any files to be excluded from oplocking:

```
veto oplock files = /*.conf/
```

Running Command Line Tools

The "exec" option allows you to run a command on the server before opening a connection between the client and a share. The following "/etc/smb.conf" entry provides an example of a typical configuration:

```
preexec = echo
"Connecting %u to
%S by %m" >> /tmp/log
```

This example would display a message on the client to the effect that the client was connecting. The "exec" and "preexec" configuration options are synonymous. The "postexec" option can be used to run a command shortly before closing a Samba connection. This allows you to display a short message for example:

```
postexec = echo
"Closing Samba Connection"
```

The "root postexec" option allows you to run a command with root privileges shortly before closing the connection. The basic configuration is the same as the "postexec" option. The "root preexec" option can be used in a similar way to "root postexec" to run a command with root privileges before opening a connection between a client and a specific share.

Connection Configuration Parameters

The "max connections" option specifies the number of connections the server permits. This option is useful if your Samba server's performance drops due to a large number of simultaneous

sessions. The following settings allows an unlimited number of simultaneous connections:

```
max connections = 0
```

The "volume" option simply defines the volume name for a Samba share. This may not sound important, but some programs that are launched from CDs need a specific volume name if they are to run. The following statement sets the name of the Samba share volume to "Install" for example:

```
volume = Install
```

Granular Access Permissions

Samba relies on Unix/Linux access control for file permissions, thus allowing consistent mapping of permitted access between the worlds of Unix and Windows. Whereas Unix offers a clear-cut file mode with nine access bits, Windows relies on detailed Access Control Lists. An ACL can stipulate different rights for arbitrary groups and users, thus allowing combinations that the Unix mechanism cannot reflect.

The Posix ACLs provide an ACL system for Linux, although it is not compatible to the NT ACL system. The ACL support provided by the operating system depends on the kernel version and distribution. Posix ACLs provide both the familiar file mode bits and an arbitrary number of entries for additional users and groups. Users or groups explicitly listed here are assigned individual access permissions (read, write, and execute). The permissions that can be assigned by the ACL are defined in the ACL mask.

The ACL can be manipulated by the "setfacl" commando on the Linux side; "getfacl" shows the assigned permissions. Samba 2.2 and later allows Windows clients to view and modify ACLs – provided you have compiled Samba "--with-acl-support". Windows NT/2000 ACLs are far more complex that those provided by Posix, however, so Samba again needs to act as a proxy, although this should not cause any problems in most practical applications. ■

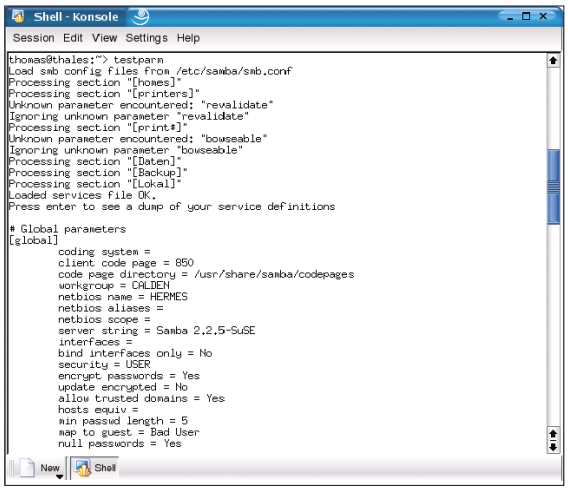

Figure 4: The "testparm" command provides a quick list of default "global" parameters, and of the share parameters

advertisement