User Management and Authentication for Samba

Access Control

Samba offers a highly granular access control system. In this month's Linux Magazine we will be discussing the really important aspects of Windows user access and account management. BY BERNHARD RÖHRIG

Nobody likes to show their cards before playing them. Applying this to data management on a server means that every user should only be allowed to read and modify the information belonging to them.

This information should not be available for other users to read or access. The Samba server provides a whole range of features for this purpose, and the */etc/smb.conf* file is where these options can be modified. The central configuration directive is:

```
security = securitylevel
```

This options sets the security level, which is the basic method for securing resources on the server and the type of authentication required from users wanting to access these resources. Box 1: Security Levels provides an overview of the four security levels implemented to date.

Share or User Level Security

The following setting is the simplest way to share resources to users:

```
security = SHARE
```

This level provides minimal protection against unauthorized read and write access to shared directories and is simple to set up.

On the other hand, every user with a password for the share will have at least read access. This is why the security level can only be recommended for access to CD ROM or print servers, or in absolutely friendly environments, particularly if used in combination with guest access – set by the the *smb.conf* directives *guest account*, *guest ok* and *guest only*. To provide a more sophisticated kind of user management, you might like to try:

```
security = USER
```

This level requires the users to identify themselves only once by means of a formal username and password based login. You can specify encrypted passwords for this level. After logging on to a server users can access any shares available to their user account within the bounds of the read and write permissions defined by their Linux user privileges.

It often makes sense to utilize the usernames and passwords defined in the Linux system's */etc/passwd* and */etc/shadow* for Samba. Authenticating with these credentials means sending the password in the clear to the server, which will then launch the *crypt()* function to compare the credentials with */etc/shadow*. The challenge response mechanism used by Samba is not available at this point as the mechanism is not capable of transferring the password.

Password Encryption

In addition to the "security = user" directive two additional configuration options must be set to use encrypted passwords:

```
encrypt passwords = yes
smb passwd file = /etc/samba/↩
smbpasswd
```

You do not necessarily need to supply the password file if you are using the default paths and names as specified when compiling the Samba server. Ensure that only root has read and write access to the file, as it contains the encrypted Windows passwords in a version for Windows 9x and NT (or later).

The file also contains the user ID of the UNIX account whose access privileges the Samba user gains after successfully logging on.

You can run the *smbpasswd* command to manage the password file. The following syntax:

```
# smbpasswd -a dwarf
```

creates a Samba account for the user "dwarf", provided a Linux account with this name already exists. For security reasons, you should not use the same password as for the Linux account, that is, if the user is even allowed to log on interactively.

Otherwise, you can type the following command:

```
#  passwd -l dwarf
```

to disable the Linux account. The admin user can type the following command to change the Samba password:

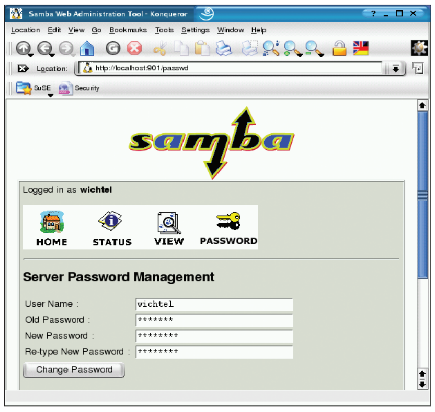
```
#  smbpasswd dwarf
```

The user "dwarf" only needs to type:

```
>  smbpasswd
```

The following command deletes the account:

```
#  smbpasswd -x dwarf
```


Samba Web Administration Tool - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://localhost:901/passwd
SuSE Security
samba
HOME
STATUS
VIEW
PASSWORD
Change Password

Logged in as **wichtel**

Server Password Management

User Name : wichtel
Old Password : *******
New Password : ********
Re-type New Password : ********

Figure 1: Swat interface

If you cannot get used to the *smbpasswd* command, it is possible to maintain the *smbpasswd* file via the Web based Swat interface (Figure 1). To allow your users to change their own passwords via the Swat interface, you must ensure that the Linux account has not previously been disabled (*passwd -l*).

You can still prevent interactive logins by assigning */bin/false* as the login shell in */etc/passwd*. The alternative is to use */usr/bin/smbpasswd* as the shell and to use Windows Telnet or Putty when changing passwords. Putty offers the advantage of sending passwords through an SSH tunnel, whereas Telnet will transmit passwords in the clear across the wire. Of course, transmitting clear text passwords should be avoided at all costs. If the passwords for the Samba and Linux accounts are identical, both accounts can be compromised with a single password – and this can have far reaching consequences for the security of your server.

In environments where many, or even the majority of users, require an interactive login in addition to file access, it might still be preferable to use the same password for both accounts. When changing passwords – and this is a task most users perform themselves – it makes sense to change the Linux password at the same time, without requiring additional user interaction. Samba uses the following *smb.conf* macro to automate this task:

```
unix password sync = yes
passwd program = /usr/bin/➥
passwd %u
passwd chat = *New*password* %n➥
\n *new*password* %n\n *changed*
```

The first line is self-explanatory. The second line contains the complete path to the Linux password management program ("%u" is the parameter for the username). The dialog between the administrator and the program, or password chat, must be emulated by the SMB daemon to allow password management. Our example was created for SuSE Linux 8.1 and will need some attention for other Linux installations.

"%n" is the parameter for the new user password (this is provided by *smbd* at runtime), "\n" represents the terminating "[Enter]" character when entering the password. Asterisks act as wildcards for any string. Make sure that you retain the correct case, as in "new".

Whenever a user launches *smbpasswd* to change their password, the password is first set to the new value in */etc/passwd*. This ensures that the old password is never required, as this process runs with "root" privileges. If anything goes wrong along the way, the Samba password is simply left unchanged.

Password Management

Password synchronization is tricky. On many Linux systems it will work for an individual user but not when the system administrator runs *smbpasswd* interactively. Additionally, password modifications often fail due to the strict controls that many Linux systems impose for new passwords, as many users will be unaware of these restrictions – and the non-descriptive error message that the *smbpasswd* program issues in this case is not much help.

If you have more than three Samba servers on your network, individual user account management on each machine can be troublesome at the best of times. The following *smb.conf* syntax provides a useful alternative:

```
security = SERVER
```

User authorization is no longer checked by an individual Linux machine; instead credentials are passed to a central server that either okays or refuses access to shared resources on the requesting machine. The advantage is that Samba accounts can be centrally managed thus reducing the workload on the admin.

If a Windows NT Domain Controller with the required user accounts is available on the network, the admin can simply add the following configuration file entry to the "security" setting we just discussed:

```
password server = DWARFKINGDOM
```

This assumes that the NetBIOS name of the domain controller is "DWARFKINGDOM". This line can also contain multiple entries, separated by space characters.

To access the resources on the Samba servers users will still need Linux accounts for the individual Linux machines, but accounts of this type do not require permission to logon interactively, and can even be disabled by invalid password entries. Solutions such as the following:

```
add user script = ➥
scriptlaunch %u
delete user script = ➥
scriptlaunch %u
```

Box 1: Security Levels in Samba 2

SHARE refers to share management introduced with WfW 3.11 (where a share is a common directory or print queue on a server). Linux users who access Samba resources receive this type of access provided they are included in the user list for the resource and know the correct password.

USER is the default setting as of Samba 2.0. The user authenticates once only on connecting and can access all the shares on the SMB server within the bounds of her Linux permissions.

SERVER is the same as **USER** with respect to access to resources. However, identities are not verified against */etc/passwd* on the Samba server itself but against a dedicated password server, which can be (but does not have to be) an NT Domain Controller. The main advantage is centralized administration of the user database.

DOMAIN has a similar effect to **SERVER**, with the exception that a Primary or Backup Backup Domain Controller in the Windows NT Domain is used as a password server. Samba version 2.2.3 or later can provide this functionality. After successfully logging on, users have access to any resources within the domain for which their accounts have been assigned permissions.

and the additional "winbindd" daemon (*man smb.conf*, *man winbindd*) allow the system administrator to completely neglect account management for this account type.

Instead of a Windows Domain Controller you can also use any Samba server for password management, provided the server has a user database ("security = user" in *smb.conf*, and its own *smbpasswd*). The server does not need to be a PDC. Centralized user management is useful even without a domain structure. If you want to stick to Windows conventions, you can stipulate the following *smb.conf* setting for your Samba servers:

```
security = DOMAIN
```

with the possible exception of one server that will assume the role of the Primary Domain Controller.

Protecting and Sharing

The resources of the machine running the Samba server are made available to authenticated users by means of a highly granular system of permissions. At the lowest level this will mean access privileges to those files and directories for which users are authorized due to their ID or group memberships.

Samba provides additional options that can be applied by means of directives in the configuration file. Access permissions of this type are assigned on by directory and are thus located in a "[share]" section of *smb.conf* where "share" represents the share name of a directory made available to Windows clients.

For example: The administrator has created a directory of sharable programs in */home/samba/shared _progs* and now wants to publish the directory as "programs" on the Windows network. The Listing „User Access Control" shows one possible approach to configuring this scenario in *smb.conf*. The "writeable = no" entry makes the directory read-only, which is the default setting. The *write list* specifies the users and groups who will be assigned write privileges, in our example these are the users "snowy" and "sleepy".

Similarly, the *read list* comprises users who will receive read-only access to the share. This setting still applies if the directory is set to "writeable = yes". The argument in our example is a group name, as is highlighted by the prepending @ character.

Note that the *write list* directive takes priority, which means that the user "sleepy" will have read and write access to the protected directory, although he is a member of the "dwarves" group. "admin users" are a potential security hazard. If a Samba user logs on with one of these IDs ("prince" in our example), the user will have unrestricted root access to all the files and directories in the share despite their Linux permissions.

Although you can add a share definition to *smb.conf* manually, Swat provides a far more convenient approach. You might like to enable the *Advanced View* which shows and allows you to edit all the *Security Options* you will require.

Box 2: Additional *smb.conf* Access Control Directives lists other directives that may be crucial to your daily administrative tasks.

What Else?

The techniques discussed so far cover most of the daily user management tasks a Samba admin may need to perform. Let's close with a look at one or two less common features and new developments.

User passwords can not only be verified against a local *smbpasswd* or a central password server, but (due to a recent development) also by reference to PAM (Pluggable Authentication Modules) or optionally LDAP (the Lightweight Directory Access Protocol). These two approaches to reducing administrative overheads are covered by two howtos in the Samba documentation [1], [2].

The following directive is used to map Windows usernames to Linux user accounts via the *smbusers* file:

Listing: User Access Control

```
[programme]
path = /home/samba/shared_progs
comment = shared program directory
writeable = no
write list = snowy, sleepy
read list = @dwarves
admin users = prince
```

```
username map = /etc/samba➘
/smbusers
```

This approach is commonly adopted to provide the user "Administrator" or "Admin" with superuser privileges. The entry in */etc/samba/smbusers* is:

```
root = administrator admin
```

Similar mappings are also common for guest access as provided by the "guest" user on Windows. ■

Box 2: Additional *smb.conf* Access Control Directives

guest ok: No password is required to access this share. The privileges assigned to the **guest account** are used for access.
guest account: A Samba user who accesses a share where GUEST OK has been stipulated without supplying a password does so with the permissions granted to the Linux user specified by this directive. "nobody" or "ftp" are typically used.
guest only: Only guest access is available to this (**guest ok** must also be stipulated).
valid users: A list of users and/or groups allowed to access this share (the READ LIST and **write list** provide a more granular approach). The "%S" parameter is normally used to provide access only to the owner of the "[home]" share.
force user: After logging on with their Samba account, users are assigned the ID and group memberships assigned to this Linux user. This provides for shared access by mapping multiple Samba accounts to a single Linux account. Samba 2.0.5 and later additionally modify the group memberships of the user; a bug in earlier versions meant that group memberships were left unchanged.
force group: Similar to FORCE USER; groups stipulated here will overwrite the group memberships assigned by **force user**.
hosts allow, hosts deny: These additional security features restrict or deny share access by reference to hostnames and IP addresses, and follow a similar approach to TCP wrappers (the *man hosts_access* manpages provide more details).

INFO

[1] "Configuring PAM for distributed but centrally managed authentication", part of the Samba documentation ("PAM-Authentication-And-Samba.html")

[2] "Storing Sambas User/Machine Account information in an LDAP Directory", part of the Samba documentation ("Samba-LDAP-HOWTO.html")