47.3. Securing Samba

The topic of security under Samba falls mainly into two categories: how to make the SMB server secure and how clients authenticate with the SMB server. Since the authentication issue is the thorniest, let's talk about it first.

In the `[global]` section of the *smb.conf* file, there is a directive called `security` that can take one of four values:*share*, *user*, *server*, or *domain*. Choosing *share* means that each shared resource has a set of passwords associated with it. Users must present one of those passwords to use the resource. *User* security requires users to provide a username and password to gain access to any of the shares. Samba can ask another SMB server to authenticate user credentials, instead of using local files, by selecting the *server* security setting. If you choose this security option, you will need to provide the `password server` directive a space-separated list of NETBIOS machine names that will do the authentication. The last security option is *domain*. In this model, your machine joins an existing NT domain that does all the user credential authentication.

If you are new to Samba, your best bet is to use *user* security. The ugliest problem of Samba now rears its head: to use encrypted passwords or not to. The issue here is that older Windows clients (early Windows 95 and pre-SP3 NT 4.0) send user passwords over the network in clear text. The good news about clear text passwords is that Samba can use your system's */etc/passwd* to authenticate users. All real accounts on your system will use their Unix username and password to connect to your SMB shares. The problems with this approach are:

- Passwords can be easily snooped from the network.
- Every SMB user requires a real account on your system.
- Newer SMB clients will need to be patched to connect to your shares.

If the first two reasons don't scare you off using clear text passwords, the last reason is pretty daunting if you need to patch a lot of workstations. However, if you still want to go this route, you need to add the elements listed inTable 47-1 to each client's registry (using *REGEDIT.EXE*).

Table 47-1. Registry settings for clear text SMB passwords

Operating system	Registry hack
Windows 95, Windows 98, Windows Me	Create a new field called `EnablePlainTextPassword` with the *dword* value 1 in the registry key:`\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP\`
Windows NT	Create a new field called `EnablePlainTextPassword` with a *dword* value of 1 in the registry key:`HKEY_LOCAL_MACHINE\system\CurrentControlSet\Services\Rdr\Parameters\`
Windows 2000	Create a new field `EnablePlainTextPassword` with a *dword* value of 1 in the registry key:`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkStation\Parameters\`

If you're not sold on clear text passwords, you will need to create a separate password file for SMB users. Luckily, there's a utility called *smbpasswd* that can manage this file for you. Adding a new SMB user who already has a Unix account on your system is as simple as:

```
# smbpasswd username
```

You will then be prompted for a password for this account. The drawback to this approach is the added maintenance of keeping the SMB passwords in sync with the Unix passwords. See *Using Samba* for some guidance here. The hope of the near future is to use an LDAP server (either Microsoft's Active Directory or a Unix LDAP server) for all system passwords. This is the dream of single-source logins and something the Samba team is working towards supporting.

After authentication issues, the big security concerns about Samba involve access control. Some of the ways to

handle access control have been shown in the configuration section of this article. Additionally, each share can use the `valid users` directive to limit the set of users to a space-separated list. You might also consider making the share `read only` and then put only a few users on the `write list`.