Underneath the hood of the imminent major release

Samba 3.0

Samba 3.0 is just around the corner, and there is some justification for high spirits: the free implementation of the SMB protocol provides Linux servers with a similar functionality to NT 4.0 and is making inroads on Windows XP.

BY VOLKER LENDECKE

The second beta release of Samba 3.0 [1] appeared just before close of press for this month's issue of Linux Magazine. A quick inspection of the number of CVS check-ins reveals that the development team has been working hard on the final release, which can be expected only a few weeks after this issue hits the shops. But beta – or even alpha – versions of Samba 3.0 are already in use at various sites, Germany's Federal Monopolies Commission being one of the the more noteworthy. It is hardly surprising that some admins are so eager to introduce the new major release, as it offers a number of attractive features in comparison to the established 2.2 version.

Two enhancements are more or less invisible from the outside: support for NT error codes and Unicode format filenames. NT error codes are transparent to the user, but important for Windows compatible error handling. Many Windows applications rely on an SMB fileserver acting just like its original Windows counterpart; thus, Samba has gone a long way towards totally emulating NT.

Unicode filename support adds the capability to display non-ASCII characters depending on your client version. In practical terms this means that clients with different country code support working on the same fileserver will actually see the same filenames. It also means that the Samba 2 *client code page* and *character set* options are things of the past.

Instead Samba 3 uses the *unix charset* option to specify the UNIX encoding of a filename, defaulting to *UTF8*, the 8-bit variant of Unicode, and thus allowing the UNIX filesystem to represent the full range of Windows characters. Administrators should typically look to retain this setting for new installations. However, special characters in filenames may not survive the jump from Samba 2.2 to 3.0 without some attention. This involves modifying typical 2.2 settings, such as

```
client code page = 850
character set = 8859-1
```

to *unix charset = CP850* in Samba 3, in order to retain current filenames.

The *dos charset* option is reserved for DOS and Windows 3.11 clients only, as Windows 95 or later systems use Unicode for filenames, and thus do not require special country coding. *display charset* specifies the character set that Samba will use to output messages on screen, however, the option will not influence the functionality of the server itself.

Another enhancement to fileserver operations affects VFS modules. Samba 3 allows you to load multiple modules simultaneously; a virus scanner and the trashcan, for example. The Samba developers renamed *vfs object* to *vfs objects*, to reflect this. Unfortunately, this means modifying and re-compiling any modules not specifically designed for Samba 3.0.

Passdb Backends

If you use SuSE's Samba packages – and thanks to some work by Lars Müller – you can tell Samba to either to use the *smbpasswd* file, or bind to the LDAP service on the fly. SuSE got this to work by basically storing two Samba versions on disk – one of them compiled with and the other without LDAP support. Users of other distributions will have to opt for *smbpasswd* or LDAP when compiling Samba. The internal name of the *smbpasswd* file interface is *passdb*.

Passdb Modules

In contrast to Samba 2.2., Samba 3 can configure *passdb* modules dynamically or even load them as shared objects at runtime. The *passdb backend* parameter takes care of this; it defaults to:

```
passdb backend = smbpasswd
```

and provides exactly the same functionality as Samba 2.2 without LDAP access. The alternative modules are *tdbsam*, *ldapsam_compat*, and *ldapsam*. The variant that uses the *smbpasswd* file as its passdb backend can store passwords for Samba, but does very little else, although this is perfectly acceptable for a normal file and print server. However, if you intend to use Samba as a domain controller, you will need to store more data about the users, such as the path to the roaming profile and the name of the login script for the user. Although Samba uses global parameters, such as *logon path*, which can be customized on a per user basis with macros such as *%U*, this approach is not particularly flexible.

Of course you could add some extra fields to *smbpasswd*, but it would be an extremely ugly hack. Modifying clear text fields of various lengths programmatically would imply re-writing the file on each modification. Password fields are of fixed length, and are simple to overwrite, even in a clear text file.

Samba 3 provides an extremely simple approach to resolving PDC limitations. The *passdb backend = tdbsam* parameter stores the information in the *smbpasswd* file in *passdb.tdb*. This allows you to individually specify fields for users, either using the *pdbedit* program, or the NT User Manager, which will be more familiar to Windows administrators. Incidentally, the Windows 2000 MMC cannot be used to edit NT users as it is designed for use with Active Directory only.

The *tdb* binary files face an issue that is common to all databases – it is impossible to create a consistent backup at runtime – this could be fatal for the central *passdb.tdb*. The *tdbbackup* program provides a workaround by parsing each *tdb* and writing its contents to another *tdb* file, which is not in use, and thus can be backed up.

Samba with LDAP

Samba has been capable of implementing an LDAP user database since 2.2, although the LDAP server interface was fairly rudimentary. It worked fine, but did very little apart from extracting the *smbpasswd* file from LDAP. Additionally, Samba 2.2 was challenged performance-wise, particularly where Samba and an LDAP server were not running side by side on a single machine: Samba 2.2 opened a new LDAP connection for even the most trivial of queries, and performed a complete TLS handshake between smbd and the LDAP server in scenarios where TLS was in use. This is expensive time-wise and definitely not on, especially in the case of servers with a heavy load.

Samba 3 will open an LDAP connection for each smbd, and keep the connection alive to provide far quicker handling of individual queries. Additionally, Samba 3 makes far more extensive use of LDAP.

In contrast to Samba 2.2, the new Samba version customizes the LDAP scheme in two respects; for one thing version 3 modifies group and ID mappings (see below) in LDAP, and for another the *sambaAccount* object used by 2.2 has been replaced by a *sambaSamAccount* object. The latter modification was designed to resolve attribute name conflicts with other applications. *sambaSamAccount* attributes all start with the *samba* prefix: thus *ntPassword* has now be renamed to *sambaNTPassword* and so on. The new schema is applied when you specify:

```
passdb backend = ldapsam
```

But the:

```
passdb backend = ldapsam_compat
```

is still provided to facilitate migration, providing for continued use of the *sambaAccount* object class.

The configuration that tells Samba which LDAP server to use has also changed. Where Samba 2.2 used the *ldap server*, *ldap port*, and *ldap tls* options, all three options are now expected as parameters to the *ldapsam* or *ldapsam_compat* module:

```
passdb backend = ldapsam:ldap:➘
//Server:Port/
```

or as follows using TLS:

```
passdb backend = ldapsam:ldaps:➘
//Server:Port/
```

If you have become accustomed to the *configure* option *--with-ldapsam* by now, you will be happy to hear that you can still use the three older options, *ldap server* et cetera.

IDMAP

IDMAP is one of the major internal changes introduce to Samba 3 – an important step on the road to total NT compatibility. Every user or group that Samba makes visible to clients at any point is represented by a SID (Security Identifier). Samba needs SIDs at various points, such as logging on to a domain, but also when displaying a file owner, or an Access Control List entry. Windows computers will expect a Windows compatible SID representation at this point, and cannot handle Linux users or groups.

Issuing Authority and Relative Identifier

A SID comprises two elements, the so-called issuing authority and a Relative Identifier (RID). The issuing authority is a Windows machine with NT or Samba Version 2.0, or later. The format of the issuing authority can differ, but the most important format is S-1-5-21-x-y-z. X, where the y and z placeholders are random 32-bit numbers which guarantee the uniqueness of the authority. A SID component is generated for each installation. A 32-bit RID is added to complete the SID; thus a SID can be represented as S-1-5-21-x-y-z-RID.

However, differing approaches lead to an issue when Samba needs to store SIDs on Linux: files can have only one owner, and there is no such thing as group ownership on UNIX style operating systems, but users log on with their Linux UIDs and not with their SIDs. Linux uses different number spaces to handle users and groups, in other words, user 1000 has nothing to do with group 1000.

Versatile SIDs

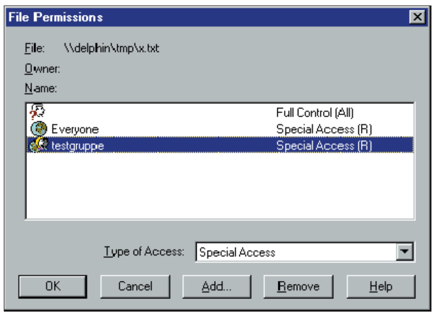
SIDs use a completely different approach: a SID can represent a user, or a group (or a local group, or …). In other words, you cannot immediately discern what a SID represents, and have to ask the issuing authority. Samba 2.2 uses algorithmic mapping to resolve this issue and requires unique mappings in both directions. Samba 2.2 converts between Linux IDs and SIDs as follows:

```
User-RID = UID * 2 + 1000
Group-RID = GID * 2 + 1001
```

This ensures that users will always have even RIDs, and groups will always have uneven RIDs. SIDs issued by Samba can be identified as users or groups without querying the issuing authority. An offset of 1000 is imperative, as Windows uses RIDs below this number for special pur-

Active Directory in Samba 3.0

The Active Directory support implemented in Samba 3 applies only to using Samba on member servers. In fact, there is very little difference between membership in an NT 4 domain and membership in an AD domain. Use *security=ads* instead of *security=domain* and add the *realm=kerberos-realm* parameter, then use *net ads join* instead of *net rpc join*. This allows clients to send Samba Kerberos tickets for authentication purposes and removes the need to query the domain controller on the fly. There are no changes apart from these.


File Permissions

File: \\delphin\tmp\x.txt
Owner:
Name:

	Full Control (All)
Everyone	Special Access (R)
testgruppe	Special Access (R)

Type of Access: Special Access

Figure 1: Windows NT 4 displaying an unknown SID

poses. For example, the Domain Administrators group is always assigned a RID of 512, and domain users have RID 513.

In the case of Samba 2.2 this scheme prevents you from simply grabbing a Windows user database from a domain controller and handing it to a Samba PDC. An NT PDC will not do you the favor of assigning even RIDs to users and uneven RIDs to groups, and Linux IDs are integers, rather than floating point numbers.

If you have used Samba 2.2 as a PDC with LDAP, you will probably have noted the RID attribute in the *sambaAccount* object. The attribute suggests that Samba 2.2 could use LDAP to assign arbitrary RIDs, but unfortunately this is not the case. Some parts of Samba actually do use this attribute, but others rely on algorithmic mappings. Samba fixes this by always following what the passdb says.

Winbind Daemon

Samba 2.2 does provide a service designed to convert SIDs to Linux-IDs and vice-versa, Winbindd. This daemon allows you to hand the user management of a Linux computer to a Windows domain. To do so, the Samba admin has to join the domain in the usual way. The following lines in *smb.conf* will do the trick:

```
[global]
workgroup = domainname
security = domain
```

Samba 3 does not use the *smbpasswd* command to join a domain, but instead:

```
net rpc join -U administrator
```

where *net* command not only looks up the domain to be joined, but also locates the PDC.

The domain controller passes a list of users and groups in SID format to winbindd. The daemon then has to create Linux style user and group IDs to reflect the list, and needs a range of numbers for this purpose. Samba 2.2 uses the *winbind uid* and *winbind gid* parameters for this purpose; Samba 3 renames these parameters to *idmap uid* and *idmap gid*, although again, the older format will still work.

Two entries are required in */etc/nsswitch.conf* to hand Linux user management to winbindd:

```
passwd: files winbind
group: files winbind
```

This means that when Glibc accesses */etc/passwd*, using a call to *getpwnam(3)*, for example, it will additionally reference the */lib/libnss_winbind.so* shared object, which in turn prompts winbindd to query a domain controller. To allow this to happen, you will of course need the correct version of *libnss_winbind.so* in your */lib* directory. Samba 3 binary packages will take care of this automatically, but if you intend to compile the sources yourself, you can locate the file below *samba/source/nsswitch/*.

Shares on Workstations

The first and second beta releases of Samba 3.0 work on the principle of winbindd providing unrestricted access to a list of Linux IDs, designed to extend the normal Samba server and resolve an issue. But what should Samba do, if a file in a Samba share is passed to a user whose account resides on a workstation on the network? This may sound slightly far-fetched, but in fact it is not only possible but quite a common occurrence. When setting the owner, the Samba server is handed a SID, which represents the future owner. This means that Samba will have to specify a suitable Linux UID for the SID.

In this situation Windows simply stores the SID as an object and responds with the SID on request. Future versions of Samba will emulate NT in this respect and also respond with the SID. Samba 2 and 3 can only accept a SID if it has been issued by an authority that the server can contact – in other words by the Samba server itself, or by any domain controller in domains where the Samba server is a member or where a trust relationship exists.

This restriction applies because Samba needs to decide whether the SID belongs to a user or a group. When ownership is transferred to a SID, you might assume that the owner would be a user, although NT does support group ownership of files: Samba would need to create both a UID for a user, and a GID for a group in this case, and postpone the decision until a login occurs. Beta 3 currently rejects unknown SIDs.

But one problem is here to stay: Samba has no chance of mapping selected Linux IDs to names, unless the

Practical Update Tips

Admins interested in updating to Samba 3.0 from a Samba 2.2 environment should take the following points into consideration, and plan and implement the configuration changes described in the following section.

- *encrypt passwords = yes* is the default. You will need to set *encrypt passwords = no* explicitly for users working without encrypted passwords on Samba 3.
- The *vfs object* parameter is now called *vfs objects*. External modules will need to be modified and re-compiled to reflect this change.
- The *character set* and *client code page* options no longer exist (see first page of article).
- Users with LDAP connections will either need to specify the *--with-ldapsam* option when compiling Samba 3, or specify the *passdb backend = ldapsam[_compat]:ldap[s]://Server:Port/* parameter to enable LDAP support dynamically.
- It makes sense to apply the *ldapsam* schema modification, even though this may be time-consuming.
- If you are using the virtual server with an smbd instance using *%L*, you should additionally specify *smb ports = 139*. Windows 2000 and later try port 445 first, but the *%L* no longer works on that port.

If you are thinking of migrating a production environment, think again, unless you have previous experience gained by migrating a test network of a similar scale, for example. If not – steer clear!

SID happens to be in a trusted domain. But this is an issue that Windows itself does not resolve: NT 4.0 simply tags SIDs of this type as *unknown accounts* or simply does not display them. The Windows 2000 or later GUIs display the SID in numerical format, if the current dialog supports this. Figures 1 and 2 show examples on Windows NT4 and 2000.

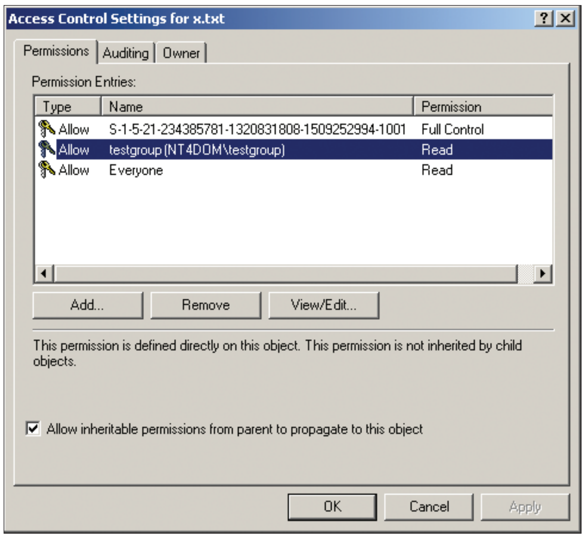
Samba 3 uses the *winbindd_ idmap.tdb* file to store these objects. The file is binary to improve access speed and Linux administrators are typically wary of binaries – for good reasons. Samba 3 helpfully provides the *net idmap dump* and *net idmap restore* tools to output *winbindd_ idmap.tdb* in a readable format, or to restore the file from a readable format. Incidentally, the file format is compatible to Samba 2.2.4 or higher, allowing you to use the *net* command to back up later 2.2 systems.

The use of Winbind on multiple machines in a network is an issue that Samba 2.2 fails to resolved. Each machine chooses an individual SID to Linux ID mapping. The development of IDMAP introduces an interface that allows for central assignment of Linux IDs. There is already an implementation designed to store IDMAP in LDAP – but this is still a 'work in progress'.

IDMAP provides the basis for three interesting Samba extensions. These are migrating Windows domains, group mappings and support for trust relationships.

Windows Domain

Thanks to the IDMAP database, SIDs that exist in a Windows user database can be retained when migrated. Samba needs to emulate an NT 4 backup domain controller installation to receive the required data including the password hashes from a Windows domain controller. That means you can transfer NT 4 and mixed mode Windows 2000 domains to Samba; Windows 2000 native mode will prevent the installation of an NT style BDC, however.


Access Control Settings for x.txt

Permission Entries:

Type	Name	Permission
Allow	S-1-5-21-234385781-1320831808-1509252994-1001	Full Control
Allow	testgroup (NT4DOM\testgroup)	Read
Allow	Everyone	Read

This permission is defined directly on this object. This permission is not inherited by child objects.

☑ Allow inheritable permissions from parent to propagate to this object

Figure 2: Windows 2000 displaying an unknown SID in numeric format

The following settings make Samba a BDC in the *WINDOWS* domain:

```
[global]
workgroup = WINDOWS
domain master = no
domain logons = yes
```

If the admin uses these *smb.conf* settings with *net rpc join -U Administrator* to join the domain, Server Manager displays Samba as a Backup Domain Controller rather than a member server or workstation. Figure 3 shows a computer called *delphin* (the author's laptop) running as a BDC in a Windows domain.

User Database from the PDC

Administrator privileges are required to join the domain as a BDC, an account capable of adding computers to the domain is not sufficient. The next step is to convince the PDC to output the user database and password hashes and it makes sense to require administrative privileges for the process.

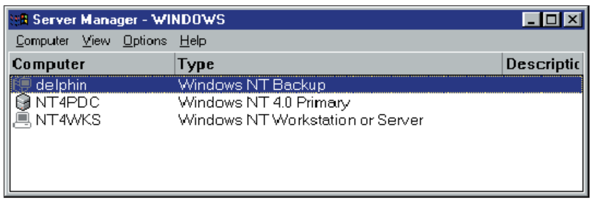
net rpc samdump allows you to investigate the user database and *net rpc vampire* overwrites the Samba user database with values retrieved from Windows. If this is to work, Samba will need to create the Windows users and groups locally on Linux. Samba uses a collection of shell scripts for this task, the "Scripts" insert provides a few sample parameters.

Provided the scripts are pointing at the right targets, the sysadmin can even use the NT User Manager to manage the Linux user database. The Linux *groupadd* command is quite strict as regards group names; thus, it makes more sense to use the *creategroup* script from *samba/ source/scripts*. The script generates new, random group names for groups that Linux refuses to create, and re-tries.

There may be no need for script based user database management on Linux and Solaris with Samba 3.0 in the future. As already mentioned, both systems possess the capability to use *nss _winbind* and winbindd to retrieve */etc/passwd* and */etc/group* dynamically from a domain controller.

This mechanism is – at least in theory – capable of dynamically creating Linux users from the information retrieved from a Windows user database. To do so, winbindd must be allowed direct access to the *passdb* database, and this is not supported in Beta 2. However, at the beginning of June Gerald Carter announced that he would be implementing this functionality.

For Samba 3.0 the fact that Samba joins the Windows domain as a BDC unfortunately does not mean that it can completely replace a BDC. Although Samba can download the whole user database en bloc, it remains blissfully unaware of deltas. Similarly, it will be impossible to use a Windows BDC in a Samba domain, as Samba implements far too little of what a Windows BDC expects from the user database.


Server Manager - WINDOWS

Computer	Type	Descriptio
delphin	Windows NT Backup	
NT4PDC	Windows NT 4.0 Primary	
NT4WKS	Windows NT Workstation or Server	

Figure 3: Samba as the BDC *delphin* in the Server Manager

User privileges are just one example: Samba does not support them at all (at present). A Samba BDC can just ignore anything it is not interested in, but a Windows PDC would need more information from the Samba machine. However, if you run Samba both on the PDC and your BDC, it is quite easy to achieve what Windows aims to achieve with this setup, that is redundancy and load balancing. To do so, admins will need to ensure that the user databases on the PDC and BDC are identical, and this includes the SID for the domain.

The easiest way to achieve this is to use the *ldapsam* backend – OpenLDAP provides the required replication mechanisms, and the only difference between the BDC and the PDC is the *domain master = no* option. Getting OpenLDAP replication set up is quite complex – even experienced admins might take two hours – and beyond the scope of this article.

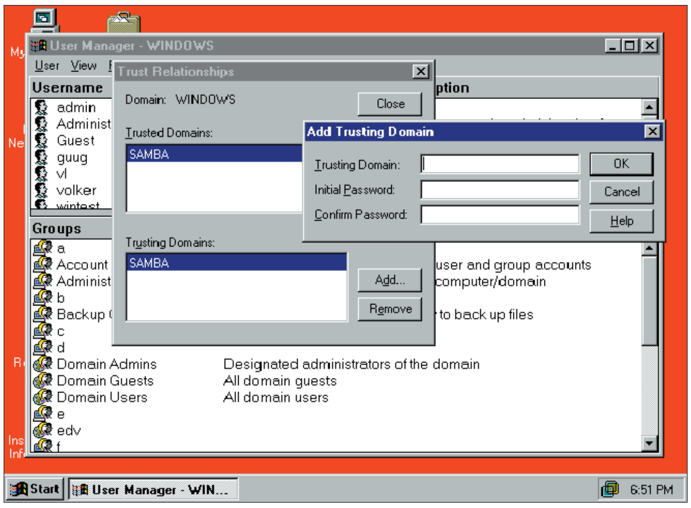


Figure 4: Setting up trust relationships on NT when the PDC is Samba with winbindd

Group Mappings

Samba 2.2 lacks one major feature required of a PDC: the ability to pass domain user group memberships to member machines. This is important as access privileges to computers in a domain are typically managed via domain groups rather than individual user accounts.

For example: Imagine a domain group called *it* that needs exclusive write privileges to a share called *\\SERVER\Prog*. Samba uses the following option:

```
valid users = @edv
```

to achieve this; a Windows member server uses the ACL editor for share privileges. To apply access control for domain groups to member servers, the domain controller needs to evaluate a user's group memberships when the user logs on and pass this information on to the member server. This is exactly what Samba 2.2 cannot do, as this functionality was first implemented in Samba 3.0.

To pass a Linux Samba 3 group membership to Windows, the admin needs to map the Linux and Windows groups using the *net groupmap* and its subtools. *net groupmap list* lists the existing mappings. Admins executing this tool will note that some mappings already exist, although they have not yet define any mappings of their own. There are technical reasons for this: Windows clients simply assume the existence of certain groups and group attributes for some tasks, for example, there always has to be a domain administrators group.

The Domain Administrators provide a typical example of mapping Linux groups to NT. Let's assume that the members of the *adm* group on Linux are to be assigned administrative privileges for their local workstations. The following command takes care of this:

```
net groupmap update rid=512 ➥
unixgroup=adm
```

The *rid = 512* parameter specifies the NT group that the Linux *adm* group will map to. *net groupmap update* is required to map this group, as a mapping already exists. However, if you want to create an NT group, you must first add the corresponding Linux group (*it* in our example) to the system:

```
net groupmap add ➥
rid=1000 unixgroup=it
```

At the current stage of development, Samba 3.0 Beta 2 requires you to specify a free RID. It can be assumed that the developers will automate this at a later stage, providing a function that locates a free RID for the new mapping.

Scripts

```
[global]
add user script = /usr/sbin/useradd "%u"
add group script = /usr/local/bin/creategroup "%g"
add user to group script = /usr/sbin/gpasswd -a "%u" "%g"
delete user from group script = /usr/sbin/gpasswd -d "%u" "%g"
set primary group script = /usr/sbin/usermod -g "%g" "%u"
delete user script = /usr/sbin/userdel "%u"
delete group script = /usr/sbin/groupdel "%g"
```

Windows Trusts

On Windows a trust relationship is a mechanism that, in domain B, allows you to manage regular users of domain A just like they were members of domain B. Windows uses trusts for various reasons:

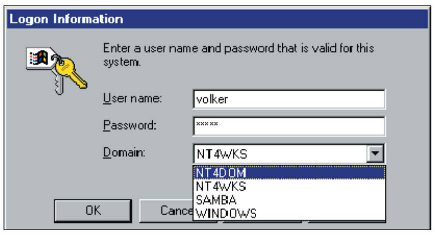
- To delegate administrative responsibility: a Windows domain administrator can modifying all aspects of a domain. An existing divisional structure within an enterprise may proscribe administrative rights for the whole hierarchy, but allow these rights for a specific branch of the hierarchy. A trust assignment allows administrators to retain responsibility for their own domains only, but provides users with freedom of movement across domain boundaries, allowing them to use any privileges assigned to them.
- Reducing the number of users on domain controllers: the User Manager for Domains becomes very clumsy if required to handle large numbers of users. Microsoft announced that NT 4 domains with more than 40,000 cannot be supported, and although not many enterprises will be affected by this limit, a user database can place a heavy load on a domain controller. Splitting a single domain up into multiple domains with trust relationships is a good workaround faced with this scenario.

Looked at from a technical point of view, a trust relationship between two domains is very similar to membership in a domain. To allow domain A to trust domain B, the domain controller for domain A needs to create a trust account in domain B (see below). The trust account is similar to a workstation account.

An example: Given two domains *WINDOWS* and *SAMBA*, where *WINDOWS* contains all the user accounts, and the *SAMBA* domain contains the machine accounts for the workstations. A workstation called *USERWKS* is a member of the *SAMBA* domain. What happens if a user called *volker* attempts to log on to the *WINDOWS* domain via the *USERWKS* workstation? The workstation will first ask its PDC, *SAMBAPDC*, if the user *volker* has entered the right password. But *SAMBAPDC* is unaware of *WINDOWS\volker*, and instead has to ask *WINDOWSPDC*.

The request that *SAMBAPDC* sends to *WINDOWSPDC* uses a secure connection to the netlogon service, the same method as the request from *USERWKS* to *SAMBAPDC*. The connection between *USERWKS* -> *SAMBAPDC* uses a shared key, generated by reference to the workstation account for *USERWKS* on *SAMBAPDC*. A workstation account for *SAMBAPDC* on *WINDOWSPDC* is required for the dialog between *SAMBAPDC* and *WINDOWSPDC*, and both systems must agree on a password.

Now there may be more than one domain controller for the *SAMBA* domain. Of course no-one will want to create accounts in the *WINDOWS* domain for each of the domain controllers, so the *SAMBA* domain controllers share a so-called trust account. The trust account is set up under Windows on the *WINDOWSPDC* using the *Profiles* menu of the User Manager. Allowing a domain to trust your current domain creates a trust account on the PDC.


Logon Information

Enter a user name and password that is valid for this system.

User name: volker
Password: *****
Domain: NT4WKS

NT4DOM
NT4WKS
SAMBA
WINDOWS

Figure 5: NT Logon dialog for the workstation *NT4WKS*, a member of the *SAMBA* domain with trust relationships

The second step involves telling the trusting domain the trust account password. The same menu item is used for this on *SAMBAPDC*. When you add a trusted domain, its password is checked and will be modified after a certain interval has elapsed.

Samba Configuration

A PDC running on Samba understands both of these functions: Windows can trust a Samba domain, and vice-versa – although this is more complicated. The first trust assignment where Samba is a server that another domain needs to trust, only entails the admin creating a trust account between the domains on the PDC in the usual way:

```
sambapdc:~# useradd windows$
sambapdc:~# smbpasswd -a -i ➋
windows$
New SMB password:
Retype new SMB password:
Added user windows$.
sambapdc:~#
```

After completing this you can add the *SAMBA* domain to *WINDOWSPDC* as a *trusted domain*. You are prompted for a password while doing so: this password will be the one you typed with the above commands.

The other direction where a Samba PDC trusts another domain is far more complex, and assumes that *nss_winbind* is working properly. Trusting another domain means that the users and groups of the external domain will be treated just like local groups, and this is what winbindd is used for. Originally designed to support single member servers, winbind now runs on a PDC and makes the users in the trusted domain visible to the local machine.

Samba Trusts Windows

If Samba is running on a PDC with winbindd, you first need to create a trust account between the two domains. Figure 4 shows the corresponding dialog box. Issue the following command on a Samba PDC that needs to trust another domain, to set the password for the domain:

```
sambapdc:~# net rpc trustdom ➋
establish windows
Password:
sambapdc:~#
```

This makes the users in the *WINDOWS* domain visible to machines in the *SAMBA* domain. Just ignore any error messages that may occur when setting up the trust; you can assume that they will have disappeared by the time the final release of 3.0 hits the streets.

Figure 5 shows the logon dialog for an NT 4.0 workstation called *NT4WKS*, which is a member of the *SAMBA* domain. The domain trusts both the *WINDOWS* and the *NT4DOM* domain. Thus, you can choose from a total of four user databases.

Conclusion and Outlook

Samba 2 falls short of domain functionality provided by Windows NT 4 in many respects. With Version 3 Samba has taken some giant leaps forward. The development team has succeeded in implementing most of the functionality of NT 4, with a few important attributes of Windows 2000 as an additional extra. Apart from joining Active Directory, you can also deposit Windows 2000 and XP printer drivers on a Samba server, and this is not supported by NT 4.

One characteristic is still lacking in Samba 3.0: It cannot be used as an Active Directory domain controller. This would not have involved too many code changes, but Samba cannot provide this functionality for structural reasons. The OpenLDAP-Team [2] would need to provide major developments to achieve this aim, including Windows style ACL support for LDAP objects. If you are interested in joining in development activities with this aim in mind, apply to the OpenLDAP development team. ■

INFO

[1] Samba: *http://www.samba.org*
[2] OpenLDAP: *http://www.openldap.org*

THE AUTHOR

Volker Lendecke is a member of the Samba team and a founder member of Service Network in Göttingen, Germany, where he is responsible for Samba, training and network security.