Insert catchy name involving Samba here

Matthew Geddes
Tellurian Pty Ltd

AUC X World II, July 2004

Abstract

This paper aims to give a technical tour of Samba, how it's integrated with Apple's OS X operating system and a brief overview of how Samba might be used to ease network administration in a mixed-network environment.

Contents

1 Introduction **4**
1.1 What is Samba? . . . 4

2 Overview of Windows networking concepts **4**
2.1 SMB/CIFS . . . 4
2.1.1 Overview . . . 4
2.1.2 Support within Samba . . . 5
2.1.3 Related configuration parameters . . . 6
2.2 The Domain . . . 7
2.2.1 Overview . . . 7
2.2.2 Support within Samba . . . 8
2.2.3 Related configuration parameters . . . 9
2.3 Name resolution (WINS, bcast) . . . 9
2.3.1 Overview . . . 9
2.3.2 Support within Samba . . . 9
2.3.3 Related configuration parameters . . . 9
2.4 Browsing / Network Neighbourhood . . . 10
2.4.1 Overview . . . 10
2.4.2 Support within Samba . . . 11
2.4.3 Related configuration parameters . . . 11
2.5 Active Directory . . . 11
2.5.1 Overview . . . 11
2.5.2 Support within Samba . . . 12
2.5.3 Related configuration parameters . . . 12

3 Configuring Samba on OS X **12**
3.1 Primary Domain Controller . . . 12
3.2 Backup Domain Controller . . . 13
3.3 Member Server or Workstation . . . 14
3.4 Standalone host . . . 15
3.5 Active Directory member . . . 15

4 Samba security parameters **15**

5 Samba simple performance parameters **16**

6 Command-line tools **16**
6.1 **smbtree(1)** and **findsmb(1) nmblookup(1)** . . . 16
6.2 **mount(1)** . . . 17
6.3 **smbcquotas(1)** and **smbcacls(1)** and **net(1)** . . . 17
6.4 **testparm(1)** . . . 18
6.5 **smbstatus(1)** . . . 18
6.6 **smbclient(1)** . . . 18

7 Caveats **19**
7.1 Unsupported features on OS X 19
7.2 Password encryption 20

1 Introduction

1.1 What is Samba?

Samba is an Open Source project originally started by Andrew Tridgell in late 1991 to scratch an itch. His itch was to get Sun workstations communicating using parts of the DEC PathWorks network protocols. Luckily for many of us, his code also worked with Microsoft's SMB client.

Samba has moved along a great deal since those early days. At the time of writing, the current version of Samba from the samba.org website is 3.0.4. Apple's 10.3.4 patch bundle ships with a modified Samba version 3.0.2.

2 Overview of Windows networking concepts

2.1 SMB/CIFS

2.1.1 Overview

SMB (which stands for *Server Message Block*) is the name that was originally given to the Windows File and Print Services file system supported by earlier versions of the Windows operating system. With later versions of the protocol, it's name was changed to CIFS (or *Common Internet File System*).

It is important to understand that the permissions assigned to a share on a CIFS host do not override the permissions assigned by the operating system at a lower level. When a remote user attempts to access some resources on our local machine using the CIFS protocol, Samba will check it's configuration to make sure that we, the administrators, have allowed that user to access that resource. If that works, the user is allowed to establish a network connection to that resource, but whether the user can read or write to a particular file on that share is entirely up to the underlying operating system.

A feature of most modern network file systems is the concept of file locking. Consider the problems that could occur if two users were to write to the same file at the same time? In some cases, this is safe. In others, devastating. Initially, many network operating systems implemented a method of locking files that locked the whole file when it was being written to. This disallowed other programs or users access to that file while it was in use. Many implementations also allowed other users or programs access to that file as long as they only intended to read from the file (ie, making no changes).

These days, many systems allow *byte-range locking*. Byte-range locking allows a program to hold a lock over part of a file and not other parts, allowing another program to lock a different part of the file that may not be locked.

In addition to using locking to maintain data integrity, a feature was added to CIFS called *opportunistic locking* or *oplocks*. Opportunistic locking allows a CIFS client to cache a local copy of a file while it's reading from it and writing to it. When another client wishes to deal with that file, an oplock break signal is sent to the first client (that holds the oplock on that file). That first client can then write it's changes back to the server and the second client is allowed to start processing that file.

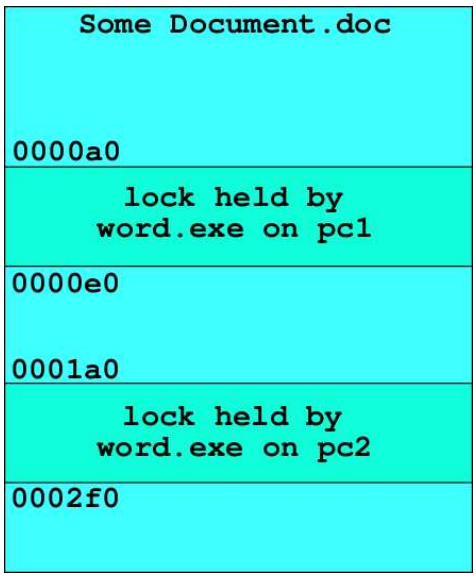

Figure 1: Byte-range locking

By allowing clients to maintain locally cached copies of files, performance is increased due to the network overhead that has been eliminated. Opportunistic locking is just a method to allow that to happen at the right time. Figure 2 briefly demonstrates opportunistic locking at a high level.

2.1.2 Support within Samba

Samba supports the SMB/CIFS protocols and has done for some time. Samba's file and print services have been considered stable for some time and depending on who's benchmarks (if any) you believe, Samba on Unix can actually outperform Windows NT and 2000 doing the same job!

The implementation of the SMB/CIFS protocols is even more clever in Samba than it seems. As part of the protocol, CIFS has support for some quite complex features. These features include:

- Opportunistic locking
- File and byte-range locking
- Access Control Lists (ACLs) / file system permissions

These are heavily intertwined with the way Windows NT and friends treat these internally.

Unix operating systems support file system permissions and various flavours of locking, but quite often they are different from other operating systems such as Windows.

Samba implements these sets of file system permissions and locking features exactly as another CIFS host (such as a Windows 2000 PC) would expect them to appear. Samba

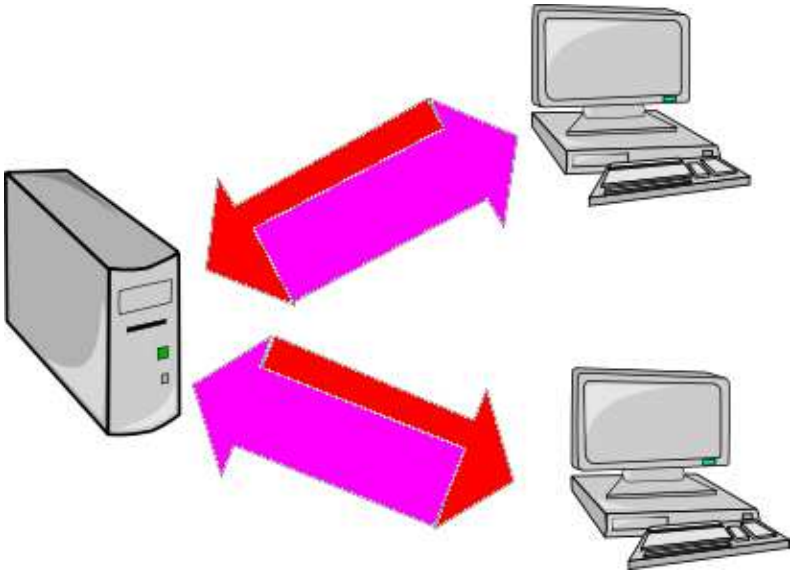

Figure 2: Opportunistic locking

also translates these CIFS network features to the way the underlying Unix operating system implements them. In some cases (such as those Unix operating systems that support ACLs and Kernel-level oplocks), it's a straightforward translation, but on many Unix systems (such as Apple's OS X), Samba needs to be quite clever to do this translation.

Figure 3 demonstrates the fact that Samba acts as a go-between between Windows and Unix network hosts.

It is for this reason that remote CIFS users (such as Windows PC users) need to have a local user account and may only access resources on that host that the Samba share-based permissions allow and that the underlying Unix permissions allow.

2.1.3 Related configuration parameters

Some of the related configuration options are not available on all platforms Samba can be found on.

- anything parameter with the word 'oplocks' in the name found in the smb.conf(5) manual page relates to opportunistic locking
- anything other parameter that contains the word 'lock' in the name relates to access locking
- likewise, any parameter with a name that contains 'acl' relates to access control lists (acls).

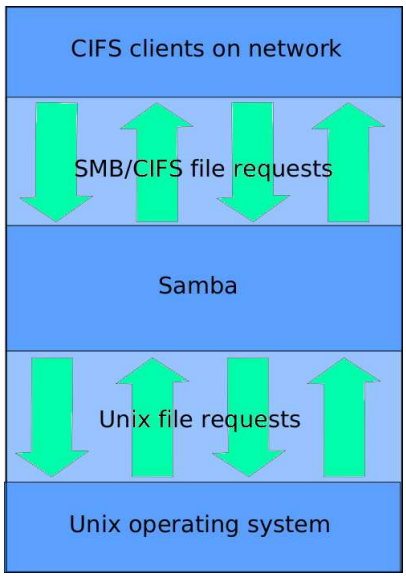

Figure 3: Samba acts as a go-between between Windows and Unix network hosts

2.2 The Domain

2.2.1 Overview

Central to most Microsoft Windows-based networks is the concept of a *domain*. A domain provides a method of centrally managing security and authentication through the use of *trust relationships*. These trusts are formed between the workstations and member servers on a network and the network's Domain Controllers. These trusts allow the domain member (the workstation or member server) to pass off any authentication or authorisation requests to the Domain Controllers.

A user who has a domain account on the domain controllers may use the same username/password pair (or other authentication token) to access resources on other hosts within that domain.

It's worth noting that a Windows (or NT) domain is very different from a DNS '.com'-style domain. The latter is primarily only a name resolution structure.

These so-called 'Domain Controllers' come in two flavours - *Primary* and *Backup* Domain Controllers. Each domain has a single Primary Domain Controller (or PDC) and may have any number of Backup Domain Controllers (BDCs). The PDC is the only host on the network that may write to the user accounts database, so this is where accounts are created and maintained. The PDC regularly replicates the contents of the user accounts database to the BDCs. The BDCs may provide authentication and authorisation services, but cannot change the contents of the user accounts database.

The BDCs provide a level of load-balancing by providing those authentication and authorisation services. They also provide a crude form of fault tolerance. If the PDC is unavailable, authentication can still happen (because the BDC has a read-only copy of the accounts database), but changes to the user accounts database (such as adding new accounts or chang-

ing passwords) cannot happen.

It is also possible for one domain to trust another (and vice versa). Allowing users from one domain to be able to access resources located in another domain without needing an account in that second domain.

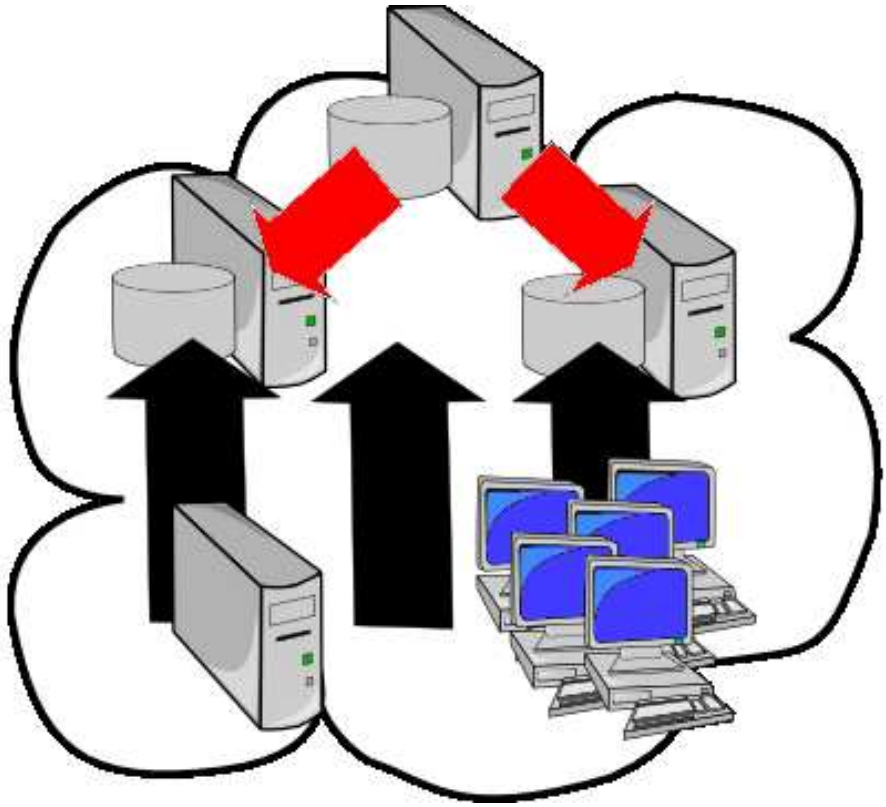

Figure 4: The Windows NT Domain network model

2.2.2 Support within Samba

As far as domains go, Samba officially supports the following in versions greater than 3.0.0:

- Acting as a Primary Domain Controller
- Acting as a Backup Domain Controller (even with a Windows PDC!)
- Forming a trust relationship with a domain; ie, acting as a Domain Member
- Forming a trust between two domains
- Avoiding domains altogether (often referred to as a Standalone server)

2.2.3 Related configuration parameters

- `domain logons` - Should we serve authentication requests?
- `password server` - tells Samba the name of the Primary Domain Controller to use for authentication
- `security` - tells Samba whether to use user-level or share-level security
- `encrypt passwords` - are the other hosts on the network expecting encrypted SMB/CIFS passwords?

2.3 Name resolution (WINS, bcast)

2.3.1 Overview

On a Windows network, WINS (or the Windows Internet Naming Service) is the primary method for resolving NetBIOS names to IP addresses. WINS was originally intended to replace DNS (the name resolution system used by the internet), but with Windows 2000 and Active Directory, Microsoft has started using DNS - providing WINS only for backwards compatibility.

Some of the major differences between WINS and DNS include:

- DNS has only recently had support for dynamic updates. Prior to that, it was necessary to manually maintain a static database of name-to-address mappings. WINS has always had the capability of both dynamic and static name-to-address mappings.
- NetBIOS names can come in many flavours. Where DNS primarily deals with host names, NetBIOS names can actually be names of computers, users, workgroups or even users and network services.

2.3.2 Support within Samba

Samba supports both the WINS client and WINS server functions. It is even capable of doing DNS lookups to answer WINS requests.

2.3.3 Related configuration parameters

- `wins support` - Act as a WINS server
- `wins server` - use a particular host(s) for WINS lookups
- `name resolve order` - set name resolution order ('node-type' in Windows-speak)
- `dns proxy` - perform DNS lookups to fulfil WINS requests

On platforms that support the name service switch (libnss), standard Unix resolution can be done against a WINS server. At this time, OS X doesn't support the name service switch.

2.4 Browsing / Network Neighbourhood

2.4.1 Overview

SMB/CIFS browsing is not the same thing as what you do with your web browser!

Supporting Microsoft Windows NT Server in the Enterprise (Microsoft Press) describes browsing to be the process of viewing all of the available network resources. SMB/CIFS network browsing can be thought of, from a user's point of view at least, as being similar to flicking through the Yellow Pages phone directory. It's much more efficient if you know what you're looking for, because you're able to look it up straight away and call them, whereas with browsing, we're not always sure what's out there, so we have a flick through the browse list and see if we can find something suitable. The front-end to this browsing process under Windows is called 'Network Neighbourhood'.

There is a certain amount of structure behind this browsing stuff. In fact, there are a number of different types of machines that participate in maintaining a list of machines and resources on our network. These are:

- Domain Master Browser - This is the big, bad machine that sits at the top of a browsing tree. It collects browse lists from other Master Browsers and propagates them down the chain
- Local Master Browser - The LMB maintains the browse list for it's own workgroup or subnet and propagates related information both back up to the DMB and further down the browse tree to the Backup Browsers
- Backup Browser - These machines accept browse requests and announcements from clients. They pass any new entries up the chain to the LMB and receive information about other parts of the network from the LMB.

Figure 5 shows these roles graphically.

These roles are defined mostly by elections. If a machine comes up on a network and is a PDC, a Preferred Master (the default for many Windows operating systems) or cannot find an LMB or DMB, it forces an election. Each operating system that is able to participate in one of these elections has a factory-coded number. This election is just a case of seeing who's number is the biggest.

This can happen quite frequently on a poorly configured network and it has been known for some operating systems (Windows 98 for one) to ignore the results of these elections when they lose - causing browsing mayhem.

Periodically these master browsers sort out amongst themselves which new entries have appeared in the browse list and clean out old entries (from machines that have rebooted, for example). This has it's problems (the cause of which I don't discuss here in much detail). The two most common problems with this browsing process are:

- Operating systems, such as Windows 98, ignoring the fact that they've lost a browse election and deciding to keep their own copy of the browse list - effectively splitting the browse list in two.

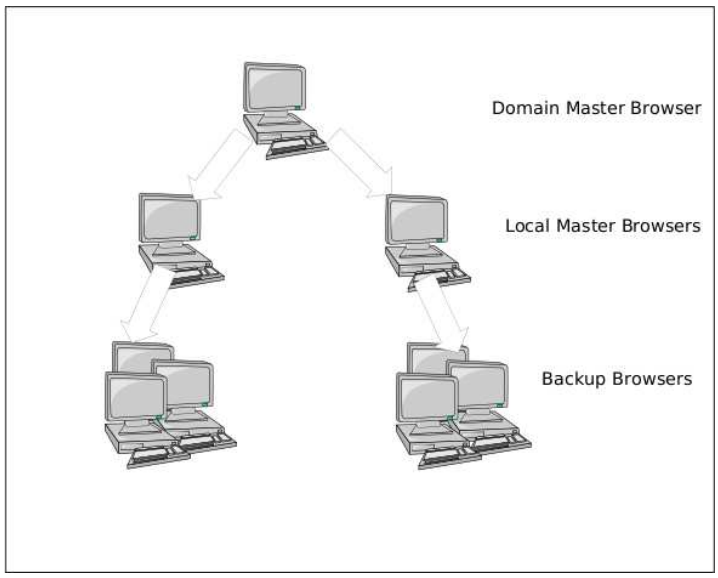

Figure 5: Windows network browsing roles

- Due to the timing behind the propagation of browse entries, it's not uncommon to have to wait up to 15 minutes for a machine or other resource appear to in the browse list and up to 51 minutes for it to be removed after a resource becomes unavailable.

It is for these reasons that browse lists (and tools such as 'Network Neighbourhood') should be treated as a convenience when they work, but should never be used as any kind of diagnostic

2.4.2 Support within Samba

Samba is able to take part in browse elections and act as any of the above browse masters. Samba can also query browse lists on a browse master.

2.4.3 Related configuration parameters

- `domain master` - When true, attempt to become the domain master
- `local master` - When true, attempt to become the workgroup master
- `preferred master` - When true, attempt to force a browse election
- `os level` - sets the amount of clout that this host carries in browse elections

2.5 Active Directory

2.5.1 Overview

Active Directory is intended to succeed the Windows NT Domain model. Active Directory allows more complex organisational structures than are easily maintainable under the NT

domain model.

Some of the key differences between Active Directory and the NT domain model are:

- Active Directory provides an LDAP interface to much of the data stored in Active Directory (such as user account and host information)
- Active Directory prefers DNS over WINS

2.5.2 Support within Samba

At the time of writing, the only way a Samba host can participate in an Active Directory network is as a member. Directory controller support is not yet supported. Samba is able to participate in a native Active Directory network, so the directory controller does not need to be configured in mixed mode.

2.5.3 Related configuration parameters

- `security` - May be set to 'ads' for Active Directory
- `password server` - a list of AD servers to authenticate against

3 Configuring Samba on OS X

We'll concentrate on configuration of Samba using the **smb.conf(5)** configuration file, rather than Apple's graphical adminstration tools. The graphical tools just manipulate this file and this way we'll gain a better understanding of what's actually happening.

The **smb.conf(5)** configuration file follows a basic format that is easily read by humans, as well as being relatively easy for a piece of software to parse. The file syntax is very similar to that of the typical Windows `.INI` file.

Global parameters are found under the `[global]` section, whereas specific shares and their attributes are set under their own sections.

3.1 Primary Domain Controller

Figure 10 contains an example of an smb.conf file as might be found on a Unix host configured as an NT 4.0-compatible Primary Domain Controller.

Here's a simple explanation of the parameters found in the example in Figure 10:

- `netbios name` - The computer's NetBIOS name. It default's to the computer's hostname
- `workgroup` - The name of the workgroup or domain we belong to. Samba doesn't need to differentiate between the two (it works it out by itself)
- `security` - Set to 'user' to tell Samba to use user-level security, which it does by authenticating users against the smbpasswd file (or optionally PAM, LDAP or TDB).

```
[global]
netbios name = OURPDC
workgroup = DOMAINNAME
security = user
encrypt passwords = yes

domain logons = yes
domain master = yes
preferred master = yes
local master = yes
os level = 65

wins support = yes

[netlogon]
comment = standard netlogon share
public = yes
writeable = no
path = /shares/netlogon
```

Figure 6: Simple example of an smb.conf file for a PDC

- `encrypt passwords` - Post-NT4SP2, over-the-network password encryption was turned on. It's possible to turn that off, but given that Samba supports it, it's more straightforward to leave it on. Turn it off for compatibility with older systems.
- `domain, local and preferred master` - Should we attempt to start and win a browse election?
- `os level` - In a browse election, the computer with the largest, errr, number wins. Windows 2000 is around 64.
- `wins support` - Should we act as a WINS server?

Windows NT 4.0 PDCs by default have a share called `netlogon`, which is also defined in our smb.conf example. The netlogon share only needs to be read-only in many cases, as it's usually just used to serve out login scripts and user and group policy files.

3.2 Backup Domain Controller

It's possible to use Samba as a BDC in an NT-domain-style network. Figure 7 shows an example **smb.conf(5)** file.

The main differences between this and the previous PDC example, are that we now have the `security` parameter set to domain, indicating that we're a member of the domain.

```
[global]
netbios name = OURBDC
workgroup = DOMAINNAME
security = domain
encrypt passwords = yes

domain logons = yes
domain master = yes
preferred master = yes
local master = yes
os level = 65
password server = OURPDC

wins support = yes

[netlogon]
comment = standard netlogon share
public = yes
writeable = no
path = /shares/netlogon
```

Figure 7: Simple example of an smb.conf file for a BDC

Because `domain logons` are still set to true, Samba works out that we're a BDC (instead of a PDC or standard member server). The `password server` directive allows us to tell Samba whereabouts to find the PDC.

3.3 Member Server or Workstation

To have a Samba machine make some of it's resources (file and printer shares) available to other hosts in the domain, we make it a member of that domain. Figure 8 shows us how.

In this example (Figure 8), we configure our member server to use the PDC and BDC machines as WINS servers. We also tell Samba to avoid browse elections and that we're a member of a domain and that we can authenticate against either OURPDC or OURBDC. Domain logons are turned off.

The `[homes]` share tells Samba to automagically share out each user's home directory. The `valid users = %S` grants access to each share to the user that owns it. For a share called fred (fred's home directory) fred will be the only user allowed to connect to that share.

```
[global]
netbios name = FILESERVER
workgroup = DOMAINNAME
security = domain
encrypt passwords = yes

domain logons = no
domain master = no
preferred master = no
local master = no
os level = 1
password server = OURPDC OURBDC

wins server = 10.0.0.1 10.0.0.2

[homes]
comment = Users' home directories
valid users = %S
writeable = yes
```

Figure 8: Simple example of an smb.conf file for a member server

3.4 Standalone host

For a standalone host, we don't need to configure much at all. Figure 9 shows an example configuration file for a standalone host, such as a web server.

In this example, we see a specific directory (/var/www/html) shared out with the name 'website'. The '@webadmin' value assigned to the `valid users` parameter tells Samba to allow connections from any user in the 'webadmin' group.

3.5 Active Directory member

Configuring Active Directory is very similar to configuring a Samba host as a member server. The only difference being that the `security` parameter is set to 'ads'.

4 Samba security parameters

- `interfaces` / `interfaces only` - tells Samba to only accept network requests on a particular set of network interfaces
- `security` - configures Samba to use different methods of access control to resources on our host

```
[global]
netbios name = WEBSERVER
workgroup = WORKGROUP
security = user
encrypt passwords = yes

domain logons = no
domain master = no
preferred master = no
local master = no
os level = 1

[website]
comment = website share
valid users = @webadmin
writeable = yes
path = /var/www
```

Figure 9: Simple example of an smb.conf file for a standalone server

5 Samba simple performance parameters

- `kernel oplocks / level2 oplocks / oplocks` - These parameters should be left on on most hosts, as they allow caching of files by clients. It may be necessary to turn them off if they cause problems
- `follow symlinks` - Setting this to a false value can give great performance benefits at the cost of some functionality. If you don't want Samba to follow Unix symbolic links (shortcuts), turn this off.

6 Command-line tools

We've seen Apple's shiny GUI, but there's a suite of useful command-line tools that may be used to manipulate Windows machine over the network.

6.1 smbtree(1) and findsmb(1) nmblookup(1)

The **smbtree(1)** command can map a network of workgroups, computers and shares into a simple tree diagram. Figure 11 shows an example using the **smbtree(1)** command.

findsmb(1) is a perl script included with Samba that can probe the network looking for SMB/CIFS hosts. Figure 6.1 gives some sample output of the findsmb command.

```
[global]
netbios name = FILESERVER
workgroup = DOMAINNAME
security = ads
encrypt passwords = yes

domain logons = no
domain master = no
preferred master = no
local master = no
os level = 1
password server = ADSPDC

wins server = 10.0.0.1 10.0.0.2

[homes]
comment = users' home directories
writeable = yes
valid users = %S
```

Figure 10: Simple example of an smb.conf file for an Active Directory member

The **nmblookup(1)** tool is used to NetBIOS name lookups on NetBIOS names. In the example shown in Figure 13, you can see that it's possible to perform a lookup on a computer name, workgroup name and even find out which host is the master browser.

6.2 mount(1)

Figure 14 demonstrates how to mount a remote Windows share under OS X. The **umount(8)** command is used to unmount the share. In the example, we are asking Samba and OS X to make the resources under the `mgeddes` share on the `laptop` server available under the local `/mnt/remote` directory as the user mgeddes. The 'mgeddes' before the '@' sign is the username, whereas the other is the share name.

6.3 smbcquotas(1) and smbcacls(1) and net(1)

smbcacls(1) allows users to remotely set access controls (ACLs) on remote filesystem objects (such as files and directories). Figure 16 demonstrates how to display and set access controls on a remote file.

```
diving:~ matthew$ smbtree -U testuser
Password:

TELLURIAN
        \\DIVING                        Mac OS X
                \\DIVING\matthew                User Home Directories
                \\DIVING\ADMIN$                 IPC Service (Mac OS X)
                \\DIVING\IPC$                   IPC Service (Mac OS X)
MGEDDES
\\LAPTOP Windows 2005 Server
```

Figure 11: Output of the **smbtree(1)** command

```
diving:~ matthew$ findsmb

IP ADDR         NETBIOS NAME   WORKGROUP/OS/VERSION
------------------------------------------------------------------------
10.0.0.225   LAPTOP        +[MGEDDES] [Unix] [Samba 3.0.0]
10.0.0.229   DIVING         [TELLURIAN] [Unix] [Samba 3.0.2]
10.0.0.20    PDC           *[TELLURIAN] [Unix] [Samba 3.0.4]
10.0.0.21    BDC            [TELLURIAN] [Windows 5.1] [Windows 2000 LAN Manager]
```

Figure 12: Output of the **findsmb(1)** command

6.4 testparm(1)

The **testparm(1)** command should be used when changes are made to the **smb.conf(1)** file to ensure that there are no syntax errors. Figure 17 demonstrates it's use and output.

6.5 smbstatus(1)

smbstatus(1) is a tool used to display the status of network sessions to our Samba service. It is also capable of showing open files and and file locks held. Figure **??** demonstrates this.

6.6 smbclient(1)

The **smbclient(1)** utility may be used in a similar fashion to the FTP command-line tool found on most operating systems. It can be used for transferring files over the SMB/CIFS protocols. It is also often used for viewing shares on a remote host (with the -L switch) and automating the transfer of files from a Windows host.

```
diving:~ matthew$ nmblookup diving
 querying diving on 10.0.0.255
 10.0.0.229 diving<00>
diving:~ matthew$ nmblookup tellurian
 querying diving on 10.0.0.255
 10.0.0.229 diving<00>
diving:~ matthew$ nmblookup -M -- -
 querying __MSBROWSE__ on 10.0.0.255
 10.0.0.225 __MSBROWSE__<01>
```

Figure 13: Sample use of the **nmblookup(1)** command

```
diving:~ root# mount -t smbfs //mgeddes@laptop/mgeddes /mnt/remote
 Password:
diving:~ root# mount
 /dev/disk0s3 on / (local, journaled)
 devfs on /dev (local)
 fdesc on /dev (union)
 <volfs> on /.vol
 /dev/disk0s5 on /Volumes/Untitled 2 (local, journaled)
 /dev/disk0s7 on /Volumes/Untitled 3 (local, journaled)
 automount -nsl [304] on /Network (automounted)
 automount -fstab [307] on /automount/Servers (automounted)
 automount -static [307] on /automount/static (automounted)
 //MGEDDES@LAPTOP/MGEDDES on /mnt/remote
diving:~ root# umount /mnt/remote
```

Figure 14: Example mounting an SMB share

7 Caveats

7.1 Unsupported features on OS X

The two main features that are not supported on OS X are Posix ACLs and kernel-level oplocks.

ACLs allow an administrator to grant some fairly granular access controls or security permissions on a file or directory. This is how Windows NT and Novell Netware treat file-system security, whereas many Unix operating systems still use the concept of users, groups and ownership.

Kernel oplocks, when supported, are used by Samba to handle caching of shared files by clients. This saves Samba having to do that itself and also provides better stability between Samba and other services such as NFS.

```
diving:~ matthew$ smbcquotas -U mgeddes //laptop/mgeddes
Password:
Quotas are not supported by the server.
```

Figure 15: Examples using smbcquotas

```
diving:~ matthew$ smbcacls -U mgeddes //SERVER/matt_test NTDomain2.png
 Password:
 REVISION:1
 OWNER:SERVER\mgeddes
 GROUP:SERVER\None
 ACL:SERVER\mgeddes:ALLOWED/16/READ
 ACL:BUILTIN\Administrators:ALLOWED/16/FULL
 ACL:NT AUTHORITY\SYSTEM:ALLOWED/16/FULL
 ACL:SERVER\mgeddes:ALLOWED/16/FULL
 ACL:BUILTIN\Users:ALLOWED/16/READ
diving:~ matthew$ smbcacls -U mgeddes -S ACL:mgeddes:ALLOWED/16/FULL //SERVER/matt_tes
 Password:
diving:~ matthew$ smbcacls -U mgeddes //SERVER/matt_test NTDomain2.png
 Password:
 REVISION:1
 OWNER:SERVER\mgeddes
 GROUP:SERVER\None
 ACL:SERVER\mgeddes:ALLOWED/16/FULL
```

Figure 16: Using **smbcacls(1)** to view and set remote file permissions

7.2 Password encryption

As of Windows NT version 4.0 Service pack 2, passwords were no longer sent across networks in plaintext, except when the password was being changed. The password was first sent through a one-way password-hashing algorithm and used in a challenge-response handshake with the server. The one-way hashing algorithm used was different from any of the hashing algorithms used on Unix systems. As a result, it's not possible to have password encryption turned on and authenticate using the standard Unix PAM system or the /etc/passwd file.

It is possible, however, to use the `unix passwd sync` and related options to configure a Samba PDC to update the Unix accounts database, or even an LDAP directory when a user changes their Windows password.

```
diving:~ matthew$ testparm
 Load smb config files from /etc/smb.conf
 Processing section "[homes]"
 Processing section "[printers]"
 Loaded services file OK.
 Invalid combination of parameters for service homes.
   Level II oplocks can only be set if oplocks are also set.
 Invalid combination of parameters for service printers.
   Level II oplocks can only be set if oplocks are also set.
 Server role: ROLE_STANDALONE
 Press enter to see a dump of your service definitions
 ^C
```

Figure 17: Using the **testparm(1)** command to check the **smb.conf(5)** file syntax

```
diving:~ matthew$ smbstatus -B

Samba version 3.0.2
PID     Username      Group         Machine
-------------------------------------------------------------------
  758   matthew       matthew       laptop       (10.0.0.225)

Service      pid     machine       Connected at
-------------------------------------------------------
matthew        758   laptop        Tue Jun 29 12:52:29 2004
```

Figure 18: Using the **smbstatus(1)** command to view open shares and locked files

```
matthew@laptop:Samba$ /usr/local/samba/bin/smbclient -U matthew //diving/matthew
Password:
smb: \> ls
  .                                   D        0  Tue Jun 29 09:53:05 2004
  ..                                  D        0  Tue Jun 15 10:24:58 2004
  .bash_history                       H     5151  Tue Jun 29 08:33:25 2004
  .CFUserTextEncoding                 H        3  Tue Jun 15 10:24:58 2004
  .DS_Store                           H    12292  Fri Jun 25 17:09:55 2004
  .profile                            H       19  Wed Jun 23 10:21:49 2004
  .ssh                               DH        0  Tue Jun 22 19:52:10 2004
  .Trash                             DH        0  Thu Jun 24 22:52:25 2004
  .viminfo                            H     6069  Wed Jun 23 13:09:06 2004
  cvs                                 D        0  Mon Jun 28 15:49:23 2004
  Desktop                             D        0  Thu Jun 24 22:52:25 2004
  Documents                           D        0  Fri Jun 25 15:03:49 2004
  Library                             D        0  Tue Jun 22 18:42:25 2004
  Movies                              D        0  Tue Jun 15 10:24:59 2004
  Music                               D        0  Mon Jun 28 20:43:44 2004
....
```

Figure 19: Using the **smbclient(1)** command to connect to a share