Chapter 1

Managing SMB File Sharing

1.1 Accessing SMB Shares

On Linux, a samba client is available that lets us connect to any Samba Server or CIFS export that is offered on our network. This includes samba exports from both Linux and Widows! To see which samba exports are available on a host, we use:

```
# smbclient -L //localhost
Enter SAMBA\somu's password:
Anonymous login successful
OS=[Windows 6.1] Server=[Samba 4.6.2]

Sharename       Type      Comment
---------       ----      -------
print$          Disk      Printer Drivers
sambashare      Disk      my share
IPC$            IPC       IPC Service (Samba 4.6.2)
Anonymous login successful
OS=[Windows 6.1] Server=[Samba 4.6.2]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

While it did ask for the root password, since we're just using the `-L` option to view the available shares, it's not needed. While `smbclient` does offer many more options, it's not very convenient to use. Instead, it's possible to directly mount the samba share on to the file system. This can be done with the command:

```
# mount -o username=somu //prime.vm.somuvmnet.local/sambashare /mnt
mount: wrong fs type, bad option, bad superblock on
 ↪  //prime.vm.somuvmnet.local/sambashare,
missing codepage or helper program, or other error
(for several filesystems (e.g. nfs, cifs) you might
need a /sbin/mount.<type> helper program)

In some cases useful info is found in syslog - try
dmesg | tail or so.
```

The utility complains that it doesn't know how to mount a CIFS file system. For this reason, we need a mount helper program, that usually starts with the prefix `mount.` and we can check if one is installed by typing it and pressing *double tab* to see if one appears in the command suggestions. In our case, the list doesn't contain `mount.cifs`, so we need to see which package might provide it:

```
# yum provides */mount.cifs
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
rhel-7-server-rpms/7Server/x86_64/filelists_db
 ↪  |  31 MB  00:00:15
rhel-7-server-rt-rpms/7Server/x86_64/filelists_db
 ↪  |  16 MB  00:00:10
cifs-utils-6.2-6.el7.x86_64 : Utilities for mounting and managing CIFS mounts
Repo        : rhel-7-server-rpms
Matched from:
Filename    : /usr/sbin/mount.cifs
```

So, we have to install it using `yum -y install cifs-utils` to be able to mount CIFS file systems. Now, the previous mount command will work:

```
# mount -o username=lisa //prime.vm.somuvmnet.local/sambashare /mnt
Password for lisa@//prime.vm.somuvmnet.local/sambashare:  ********
# mount | grep mnt
//prime.vm.somuvmnet.local/sambashare on /mnt type cifs
 ↪  (rw,relatime,vers=1.0,cache=strict,username=lisa,domain=PRIME,uid=0,noforceuid,gid=0,
 ↪  noforcegid,addr=10.0.99.11,unix,posixpaths,serverino,mapposix,acl,rsize=1048576,
 ↪  wsize=65536,echo_interval=60,actimeo=1)
```

Once the password has been entered, if we go to the mount location, i.e., in our case, `/mnt`, we can see the contents of the samba share. Using this method, we can mount samba shares directly onto our local file system, and it even allows us to copy files from Windows shares to our Linux computers!

1.2 Samba Server Configuration Overview

The purpose of a samba share, just like a NFS share, is to share something on the file system. As such, the following are the steps involved in the creation of a samba share:

- Create the share (i.e., a directory) on the Linux File system.
- Grant access permissions on the Linux File System - without this no matter what is shared gets an *access denied* error.
- Define the share in `smb.conf` - the main configuration file for the samba server, and every share needs to be defined in it.
- Configure security (in `smb.conf` as well).
- Consider Access Restrictions through `smb.conf` and/or Linux File System permissions.
- Start the Samba Server with `systemctl start smb nmb`. While `smb` is the file sharing service, `nmb` is for *naming*. So, if we want that naming is available as it is on a Windows system, the `nmb` service must also be started.
- Both services must be enabled as well with `systemctl enable smb nmb`.

1.3 Creating the Samba Share: Linux Tasks

This particular section deals with preparing Linux for use with Samba. Before work on the samba share can begin, a directory (which will eventually become the samba share) is available, the appropriate permissions are set and the required Linux users are configured as well.

So, first we create a directory called `/sambashare`. Next comes permissions - which can be anything, but if we want we could do it the *Windows way*:

```
# mkdir /sambashare
# chmod 777 /sambashare
```

While the above isn't secure, it ensures that whatever functionality we're trying to provide won't be hindered by file system permission restrictions.

If we want to grant access to specific users, we must ensure that the users exist at the Linux level. If they don't yet exist, we create said users using `useradd <userName>`. While we're going to create *samba users* later as well, samba users can only be created if a corresponding Linux user exists already!

1.4 Creating the Samba Share: smb.conf Tasks

The two primary packages required for a Samba server are called `samba` and `samba-client`. While there are packages that add functionality, they're optional. The primary configuration file for the samba server is called `/etc/samba/smb.conf`.

Within the file, there's a **global** section that deals with the configuration of the server itself. The first important parameter is the `workgroup` which is set to a value of *MYGROUP* by default.

It is also possible to set the interface(es) to be used by the samba server as well as allow connections only from certain hosts by setting the following lines (with appropriately modified values):

```
interfaces = lo eth0 192.168.12.2/24 192.168.13.2/24
hosts allow = 127. 192.168.12. 192.168.13.
```

Down below, there are share definitions as well, that define everything that will be shared from the server. There are a lot of examples available for us to configure our own shares. An especially illuminating one is the *[public]* share. We can define our own share with:

```
[sambashare]
comment = my stuff
path = /sambashare
public = yes
writable = yes
write list = +users
```

The `public = yes` statement makes it available to everyone in read-only mode, but the option `writable = yes` makes it writeable by everyone as well. The mere inclusion of *writable = yes* demands that a *write list* also be added. This indicates that anyone can access the contents of the share, but only those who're a part of the group *users* can write to the share.

1.4.1 Starting the samba server and verifying shares

Now we can start the service with:

```
# systemctl start smb
```

At this point we can verify that it works by using:

```
# smbclient -L //localhost
Enter SAMBA\somu's password:
Anonymous login successful
OS=[Windows 6.1] Server=[Samba 4.6.2]

Sharename       Type      Comment
---------       ----      -------
print$          Disk      Printer Drivers
sambashare      Disk      my share
IPC$            IPC       IPC Service (Samba 4.6.2)
Anonymous login successful
OS=[Windows 6.1] Server=[Samba 4.6.2]

Server               Comment
---------            -------

Workgroup            Master
---------            -------
```

To use this share, the one thing that's lacking is *samba users*.

1.4.2 Creating Samba users

The command to create new samba users is `smbpasswd -a <userName>` (the `-a` option adds a new user):

```
# smbpasswd -a lisa
New SMB password:
Retype new SMB password:
Added user lisa.
# smbpasswd -a lori
New SMB password:
Retype new SMB password:
Added user lori.
```

Again, the above user-creation process will only work if there is a user with the same username on the Linux system as well. If not, we'll get an error message like:

```
# smbpasswd -a tesla
New SMB password:
Retype new SMB password:
Failed to add entry for user tesla.
```

Now we're free to mount it on the file system, using:

```
# mount -o username=lisa //localhost/sambashare /mnt
Password for lisa@//localhost/sambashare:  ********
# mount | grep mnt
//localhost/sambashare on /mnt type cifs
 ↪  (rw,relatime,vers=1.0,cache=strict,username=lisa,domain=PRIME,uid=0,noforceuid,gid=0,
 ↪  noforcegid,addr=0000:0000:0000:0000:0000:0000:0000:0001,unix,posixpaths,serverino,
 ↪  mapposix,acl,rsize=1048576,wsize=65536,echo_interval=60,actimeo=1)
```

Now if user *lisa* were to go to /mnt and type the command `ls`, she'd get:

```
# su - lisa
$ cd /mnt
$ ls -l
ls: reading directory .: Permission denied
total 0
```

1.5 Tuning the Share for Access Restrictions

So, even when a user on samba with a corresponding Linux user tries to `ls` in the *samba share*, they get a *Permission denied* error. This is a SELinux issue. For example, when we see the security context of the share, the context is:

```
# ls -Zd /sambashare
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /sambashare
```

To ensure that the problem is caused by SELinux, one quick and easy way to test it is using `setenforce 0` and then checking if that worked. Now if we used the `ls` command in that directory, we'd get no errors. While turning SELinux to permissive mode is okay for testing purposes, it's only a temporary solution at best. Before we can enable SELinux to work with samba however, we need to take care of certain other options.

1.5.1 hosts allow

The `hosts allow` defines the networks that are allowed to use the samba share. The parameter values consist of only the relevant bits of the network ID. Thus, instead of writing `192.168.12.0/24`, we need only write `192.168.12.` and even host names are also allowed here in the format `.example.com`.

There's another way to specify which hosts can access the shares and which can't using **firewalld**. It might be better to allows hosts through only in firewalld, but for the end result it doesn't matter. However, setting up both `hosts allow` and firewalld may cause conflicts. Thus, one should allow all hosts to pass through, while the other restricts the passage to only certain hosts.

1.5.2 Read/Write permissions

Within the smb.conf file, the `writable = yes` parameter makes the share writable. However, we also need a `write list` parameter defined which is set to the name of a group. For example, if the members of the group *users* are to be allowed, then the parameter can be either: `write list = +users` or `write list = @users`. Both are accepted.

The use of `public = yes` makes the share accessible with *read* access to all users. However, if only certain users should have read access, then instead of the public parameter, we use the `valid users` parameter. To ensure only the users of the group *users* have read-write access, we use:

```
valid users = @users
writable = yes
write list = @users
```

Finally, we can remove any permission settings from the smb.conf file using `read only = no`, which makes the share readable and writeable to anyone, subject to the Linux file system permissions. If, however, `read only = yes` has been set, no one can write to the share no matter what Linux file system permissions have been applied.

1.6 Verifying the Configuration

To verify that the configuration itself contains no errors, we can use the command:

```
# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
Processing section "[homes]"
Processing section "[printers]"
Processing section "[print$]"
Processing section "[sambashare]"
Loaded services file OK.
Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions
```

The fact that it isn't complaining about any errors, as well as the `Loaded services file OK` statement prove that there's no errors.

1.7 Using Samba-Related SELinux Settings

To find out all the man pages related to SELinux and Samba, we have to first ensure that all the SELinux manpages are installed (using `sepolicy -a -p /usr/share/man/man8; mandb` commands) and then use:

```
# man -k _selinux | grep samba
samba_net_selinux (8) - Security Enhanced Linux Policy for the samba_net processes
samba_selinux (8)     - Security Enhanced Linux Policy for the smbd processes
samba_unconfined_net_selinux (8) - Security Enhanced Linux Policy for the
↪  samba_unconfined_net processes
samba_unconfined_script_selinux (8) - Security Enhanced Linux Policy for the
↪  samba_unconfined_script processes
sambagui_selinux (8) - Security Enhanced Linux Policy for the sambagui processes
```

The manpage we're looking for is called `samba_selinux` which contains information about the correct file and port contexts required. We can use two different types of SELinux contexts: `samba_share_t` and `public_content_rw_t`.

1.7.1 Choosing file security context

samba_share_t

Allows the samba share to only be used by samba.

public_content_rw_t

This security context enables access to the directory from multiple services such as samba, apache, etc.

1.7.2 Setting the appropriate file context

If for example, we choose `samba_share_t` to be set as the new file context, we use:

```
# cd /
# ls -dZ /sambashare
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /sambashare
```

We have to change the `default_t` file context to `samba_share_t` file context using:

```
# semanage fcontext -a -t samba_share_t "/sambashare(/.*)?"
[root@prime /]# restorecon -Rv /sambashare
restorecon reset /sambashare context
 ↪  unconfined_u:object_r:default_t:s0->unconfined_u:object_r:samba_share_t:s0
```

At this point the directory should be accessible while SELinux is turned on. Finally, we do:

```
# setenforce 1; getenforce
Enforcing
# mount | grep sambashare
# mount -o username=lisa //localhost/sambashare /mnt
Password for lisa@//localhost/sambashare:  ********
# mount | grep samba
//localhost/sambashare on /mnt type cifs
 ↪  (rw,relatime,vers=1.0,cache=strict,username=lisa,domain=PRIME,uid=0,noforceuid,gid=0,
 ↪  noforcegid,addr=0000:0000:0000:0000:0000:0000:0000:0001,unix,posixpaths,serverino,
 ↪  mapposix,acl,rsize=1048576,wsize=65536,echo_interval=60,actimeo=1)
# cd /mnt
# ls -l
total 0
```

Now the samba share mounts and is accessible without errors!

1.7.3 SELinux Booleans for Samba

The list of SELinux booleans for samba can be seen with:

```
# getsebool -a | grep samba
samba_create_home_dirs --> off
samba_domain_controller --> off
samba_enable_home_dirs --> off
samba_export_all_ro --> off
samba_export_all_rw --> off
samba_load_libgfapi --> off
```

```
samba_portmapper --> off
samba_run_unconfined --> off
samba_share_fusefs --> off
samba_share_nfs --> off
sanlock_use_samba --> off
tmpreaper_use_samba --> off
use_samba_home_dirs --> off
virt_use_samba --> off
```

The `samba_enable_home_dirs` and `use_samba_home_dirs` are two of the most important and useful ones. The former allows the users to access the Linux home directories shared by the present (this) samba server. However, the `use_samba_home_dirs` allows users to access *remote* home directories. So, on the samba server, we turn on home dirs by:

```
# setsebool -P samba_enable_home_dirs on
```

Now samba home directories are enabled on this server.

1.8 Opening the Firewall for SMB Traffic

To get a list of all the possible services on the firewall, we use:

```
# firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc
↪ bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb
↪ custom dhcp dhcpv6 dhcpv6-client dns docker-registry dropbox-lansync elasticsearch
↪ freeipa-ldap freeipa-ldaps freeipa-replication freeipa-trust ftp ganglia-client
↪ ganglia-master high-availability http https imap imaps ipp ipp-client ipsec
↪ iscsi-target kadmin kerberos kibana klogin kpasswd kshell ldap ldaps libvirt
↪ libvirt-tls managesieve mdns mosh mountd ms-wbt mssql mysql nfs nrpe ntp openvpn
↪ ovirt-imageio ovirt-storageconsole ovirt-vmconsole pmcd pmproxy pmwebapi pmwebapis
↪ pop3 pop3s postgresql privoxy proxy-dhcp ptp pulseaudio puppetmaster quassel radius
↪ rpc-bind rsh rsyncd samba samba-client sane sip sips smtp smtp-submission smtps snmp
↪ snmptrap spideroak-lansync squid ssh synergy syslog syslog-tls telnet tftp
↪ tftp-client tinc tor-socks transmission-client vdsm vnc-server wbem-https xmpp-bosh
↪ xmpp-client xmpp-local xmpp-server
```

Now, since we're using the samba server, we need the `samba` service to be allowed on the firewall. We enable it using:

```
# firewall-cmd --permanent --add-service=samba
success
# firewall-cmd --reload
success
```

On another machine (samba client), we can use `nmap` to verify the availability of service using:

```
# nmap prime.vm.somuvmnet.local

Starting Nmap 6.40 ( http://nmap.org ) at 2018-04-02 01:05 IST
Nmap scan report for prime.vm.somuvmnet.local (10.0.99.11)
Host is up (0.00070s latency).
Not shown: 991 filtered ports
```

```
PORT     STATE  SERVICE
22/tcp   open   ssh
25/tcp   open   smtp
53/tcp   open   domain
80/tcp   open   http
139/tcp  open   netbios-ssn
443/tcp  open   https
445/tcp  open   microsoft-ds
2022/tcp closed down
3306/tcp open   mysql
MAC Address: 00:0C:29:3B:B9:1C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 4.95 seconds
```

The `netbios-ssn` and `microsoft-ds` ports are the ones corresponding to the samba server, and thus it is accessible through the firewall.