Trustix Secure Linux 1.2 Users Guide (v1.2.0)

November 27, 2000

Contents

Acknowledgements

- Linux is a trademark of Linus Torvalds
- UNIX is a trademark of the open group
- Some parts of this documentation are based on various man pages and documents from the Linux Documentation Project (http://www.linuxdoc.org)
- We thank RedHat and Mandrake without whom the creation of Trustix Secure Linux would have been a much bigger task.

1 Introduction

This is the users manual for Trustix Secure Linux — A Linux based operating system made by Trustix AS especially for servers. In this document, we will attempt to describe what we believe will be the most common tasks performed on this system.

We hope you will have as much fun using Trustix Secure Linux as we had creating it!

1.1 What is Linux?

Linux is an operating system (OS) kernel created by Linus Torvalds and a loosely knit team of contributors from all around the world. A complete operating system consists of several programs and libraries in addition to the OS kernel.

A Linux distribution is a collection of such programs and libraries, as well as the OS kernel and several applications and services. There are several Linux distributions available on the market, but Trustix felt that most of these were too much geared for running on desktop computers and thus cumbersome for server use. We therefore decided to create Trustix Secure Linux — a linux distribution for servers with a primary focus on security and simplicity.

Linux is a clone of Unix and therefore share a lot of characteristics with it. In this documentation the word Unix will be used in the meaning of Unix and free clones like *BSD and Linux.

1.2 The command line

To be able to use Unix efficiently, one must learn to use the command line. Command Line Interfaces are known for not being very friendly to new users, but potentially much more powerful than any GUI (Graphical User Interface). Since complexity and security area two things that do not go very well together, and GUI solutions are much more complex than CLI (Command Line Interface) from a design point of view, we have chosen to only support CLI in Trustix Secure Linux.

We also felt that at least we would not want to weigh down a server with loads of graphical bells and whistles that we wouldn't see anyway. Our servers are on the fourth floor, while the developers and administrators sit on the ground floor only seeing the servers through ssh login. We believe that similar configurations are more the rule than the exception around the world.

When giving examples of commands to enter on the command line, we will prepend a # if the command is supposed to be run as root, and a $ if the command should be run as a normal user. This preceding character should not actually be entered on the command line, as it is only put in the documentation to show what should be entered on the command line. Actually, you will find this character as the last letter of the command prompt in many Unix operating systems.

When describing how to use a command, the following form is often used:

```
$ command <argument> [argument]
```

The $ means that the command could be issued by a normal user. The arguments in triangular brackets ('<>') should always be given to the program, while arguments in square brackets ('[]') are optional. The fictional command 'command' from the above example should be called like this (always remembering not to actually type in the preceding $s and #s):

```
$ command arg
```

or

```
$ command arg nextarg
```

1.3 Basic commands

To navigate the directory tree of a Unix system, you use the command 'cd'. This command works just like in DOS/Windows — you use the directory (or "folder" in windows terms) you want to move to as the argument. To check what directory you are currently working in, you can use the command 'pwd' (Print Working Directory). Note that Unix systems use the slash (/) instead of the backslash (\) as directory separator.

Example: To make /home your working directory, use the following command:

```
$ cd /home
```

To copy a file, use 'cp <source> <destination>'. To move or rename a file, use 'mv', which has the same basic syntax as 'cp'.

Example: To copy the file /home/user/filename to the directory /home/user/files/, you could write the following:

```
$ cp /home/user/filename /home/user/files/
```

If /home/user/ is your working directory (cd /home/user/), it is enough to write:

```
$ cp filename files/
```

To make a command work on many files at the same time, use the wildcard '*'. This wildcard expands to mean "all files". Thus the command:

```
$ cp * /home/user/files/
```

would copy all files from the current working directory to /home/user/files. You could also use a command like this:

```
$ cp /home/user/files/* /home/user/morefiles
```

Similarly, to copy all the files from the current working directory to /home/user/files/, you would use this command:

```
$ cp * /home/user/files
```

Paths like /home/user/files can be specified relatively to your working directory or as "absolute paths" that can mean only one place in the file hierarchy. The difference between absolute and relative paths is the inclusion of a preceding '/'.

To clarify with an example: If your current working directory is /home/john/, specifying the path /home/john/files/ would mean the same as just saying files/ to the system. The difference is that if your working directory was /home/pete/, saying files/ would mean /home/pete/files, while saying /home/john/files still would mean the same.

If this seems unclear, try rereading the above examples of ‘cp’ usage keeping relative and absolute paths in mind.

To list the files and directories in your working directory, use the command ‘ls’. To get a more comprehensive listing with more information about each file, use ‘ls -l’.

As in DOS/Windows, it is also possible in Unix to have hidden files, but unlike those operating systems, ‘hidden’ is not a file attribute in Unix. Instead, all files or directories which have names starting with a dot (‘.’) are hidden. To list all files including “hidden” files with ‘ls’, use ‘ls -a’. You can also specify several arguments to ls at once. To get a comprehensive listing of all files—including hidden ones—in a directory, use ‘ls -la’.

To list the contents of a file, you can use ‘cat <filename>’. This command dumps to the screen the whole contents of the file specified as an argument. ‘cat’ can also take several filenames as arguments, and print their contents in the order specified at the command line.

If the output from ‘cat’ is more than one screenful of text, you would traditionally use ‘more’ instead. ‘more <filename>’ stops after one screenful of text, and allows you to scroll down one screenful by pressing the space bar, or one line by pressing the enter key.

A more convenient alternative to ‘more’, called ‘less’ is also offered. This command enables the user to also move upwards in the file that is displayed, and also has some more advanced features. The arrow keys on the keyboard are used to scroll up and down. To search for a string, type ‘/’ followed by the string you want to search for and hit the enter key. To get the next hit for your search string, press ‘n’. When you are done reading, press ‘q’ to quit.

A bigger listing of commands for ‘less’ can be accessed by pressing the ‘h’ key while viewing a file.

The commands ‘more’ and ‘less’ can also be used if the output from a program is too big to fit the screen. If your working directory is /bin (cd /bin), the output from ’ls -l’ would not usually fit the screen. Using ’ls -l | less’ would open the output of ’ls -l’ in ’less’. Redirecting the output from one program into another is called piping, and the ’|’ character is called a pipe.

For comprehensive help on any Unix command, use ‘man’ with the command name as an argument. As the command uses ‘less’ to display the content, maneuvering inside man is done the same way as in less.

1.4 Editing a file

The editor included with Trustix Secure Linux is called ‘vi’. This is the standard editor for Unix, so chances are that if you learn at least the basic use of ‘vi’, you will be able to edit files on any Unix system.

The primary issue to understand about ‘vi’ is its ‘modes’. There are two primary modes - Command mode and Insert mode. When in command mode, hitting the keys on the keyboard will issue various commands. Insert mode is used when writing text.

The editor is invoked with ‘vi <filename>’. To edit the file smb.conf in the /etc directory, you would use ‘vi /etc/smb.conf’.

The editor is started in command mode. Here you can move around using the arrow keys, or hjkl if your keyboard does not have any arrow keys. To start writing, hit ‘i’ to insert text before the character the cursor is over or ‘a’ to insert text after it. ‘A’ (shift-a) can be used to insert text at the end of the current line.

To get back into command mode, hit the escape key.

When in command mode, hitting ‘x’ deletes the character under the cursor (like the delete-key in windows). ‘X’ (shift-x) works like the backspace key in other editors.

To search for a string in ‘vi’, make sure you are in command mode (hit esc if you are not sure) then use the same method as for ‘less’. ‘/’, enter the string, press enter. Also like in ‘less’, ‘n’ can be used to cycle through the matches.

The ‘vi’ editor might seem cumbersome and impractical to use, and to a new user, this might be correct, but as it has many more functions than the ones described here, an experienced user can really make it fly. For example, this documentation was primarily written using ‘vi’.

For more information about this editor, we can recommend the book ‘Learning the vi editor’ from O’Reilly books.

(http://www.oreilly.com/catalog/vi6/noframes.html)

1.5 Using rpm

The software used by Trustix Secure Linux is typically distributed in rpm packages. The installation program for rpm packages is called ’rpm’ and is used like this:

```
# rpm -ivh <package_file_name>
```

To get a list of all packages installed, type the following:

```
$ rpm -qa
```

The whole list might be a little overwhelming, and will probably not fit on a terminal screen, so this is a good place to use pipes and pipe the output to ‘less’ like this:

```
$ rpm -qa | less
```

If you want to know if a particular package is installed, you can use the command from the above example and search for the name as documented earlier. You can also use the following syntax:

```
$ rpm -q <packagename>
```

For example:

```
$ rpm -q apache
```

would tell you if apache is installed, and which version of it that is installed if it is.

All packages that come with Trustix Secure Linux are digitally signed with gpg (GNU Privacy Guard). This means that before installing an rpm, you would be advised to check its integrity with the following command:

```
$ rpm -K <filename>
```

1.6 Service management

Server programs, like the apache web server or the proftpd ftp server, are often referred to as services. This is quite natural when you think about it, as these are programs that enables the server computer to give a certain service to other computers.

Many such server programs are included in Trustix Secure Linux, but for security reasons, and because it is our opinion that the computer should do just what the administrator tells it and no more, only the most important of these are started by default. However, it is quite trivial to configure additional services.

The probably easiest way of doing this is by using the program 'ntsysv'. Just type the command, and you will find yourself inside an easy to use interface for turning services on and off. Just make sure that you do not turn on unnecessary services. If unsure as to whether you need a service or not, the safest option is to leave it off, and test if what you want to do works with that service off. If it did, you did not need the service. It is never a good idea to enable a service just because you perhaps might need it later. Enable it later when you need it, instead.

If a service is enabled by default, though, it is probably for a good reason, so you should be absolutely sure you know the consequences of what you are doing before turning such a service off. Common services enabled by default are:

- keytable – Loads the selected keyboard map. Especially important if you do not use a US keyboard.
- netfs – Automatically mounts any network file systems specified in /etc/fstab.
- network – Scripts for starting up network access. If you do not enable this, the network will be unavailable on boot.
- random – Saves and restores system entropy pool for higher quality random number generation. A nice security feature.
- postfix – The mail server, initially configured to only accept local connections. Needed for services like cron to be able to send mail to root when encountering problems.
- syslog – A facility used by many deamons for logging purposes. If you do not run this, other services may not be able to generate logs.
- crond – The cron daemon is a standard Unix program that runs user-specified programs at periodic scheduled times. This program is useful for updating various system databases and so forth.

Notice that 'ntsysv' does not automatically start a configured service, so that if you for example turned on 'sshd', this program would not start until the next time you rebooted you computer. To start the secure shell service at once, you could use the following command:

```
# service sshd start
```

To stop a service, change 'start' to 'stop'. Many services will also provide some status information if the keyword 'status' is used. The 'service' command will start or stop a specified service, now, but will not affect what happens the next reboot.

A handy and often more effective alternative to 'ntsysv' is the command 'chkconfig'. To configure sshd to run on system startup, you would use one of the following:

```
# chkconfig sshd on
```

or

```
# chkconfig --add sshd
```

1.7 Generating a server certificate for use with OpenSSL

Many secured services use OpenSSL to achieve encryption. For this to work, you will need to generate a server certificate and encryption key. To create this, you can use the following steps:

1. Change directory to /etc/ssl/certs
2. Use the command:

```
RANDFILE=/dev/urandom openssl req -new -x509 -nodes \
   -out server.pem -keyout server.pem -days 365
```

This generates a RSA private key and a certificate and puts them both in a file called server.pem. You would probably want to change this if you want to have have separate certificates for each encrypted service.

Note that this certificate is not very useful for business purposes, as it is not issued by any well known third part that has paid money to the large browser makers. This means that users will have to click through all kinds of warnings to get into pages protected by such an SSL certificate. The connection will still be properly encrypted, but you might have a hard time convincing people to use your webshop if their computer has first informed them that you might be doing someone bad.

2 Managing user accounts

Trustix Secure Linux comes bundled with many different tools for managing user acconts. Programs are provided for adding, configuring and deleting accounts.

When installing the system, the all powerful superuser 'root' is created, and you set the password for this user. You also get the chance to create other user accounts, but it is of course also possible to do this later at any time. This part of the documentation deals with creating, managing and deleting user accounts and groups.

2.1 Introduction

Each user that is going to access the computer, should have his or her own user account. If they do, one can always see who is logged in and who is doing what. Since every user only has access to a limited set of files and resources, this will also decrease the amount of damage that can be done through a compromised account or by a malicious user.

If people want to share files on the server, a group can be created for this purpose, and those who should have access to the files can be added to that group. A user can be a member of several groups.

2.2 The ‘su’ command

The command ‘su’ allows a user to run a shell with substitute user and group IDs. This means the command can be used to “switch” between users without having to log out and in again. This is very pratcical for system administrators, who should use their normal accounts for most work, but also frequently want to issue one or two commands as root.

To change to root, use the following command:

```
$ su -
```

To change to another user, the username is added:

```
$ su - someuser
```

If issuing the ‘su’ command as a normal user, you will be prompted for the “target” users password (i.e. the user you want to change to). When using ‘su’ as root, you will not have to enter any password, as the all powerful root user can do anything he wants anyway.

NOTE: For security reasons, a user has to be a member of the group with GID 0 (typically the group root) to be able to ‘su’ to root. See the section on group management for instructions on how to do this.

2.3 Adding a user

Adding a user in Trustix Secure Linux is quite simple. In most circumstances, following the three steps described here would be sufficient:

1. First, log in as root. This is necessary for gaining access to the files where user information is stored.
2. Second, use the command ‘adduser’ and supply a username as an argument to this command like this:

   ```
   # adduser newuser
   ```

   The above command creates a new user called ‘newuser’. The username can be almost anything, but there are some simple criteria that must be matched:

   - A username should be created from the 26 letters of the latin alphabet and the numbers 0 – 9, and thus can not contain any “European” or “special” characters.
   - Only lowercase characters should be used.
   - While creating usernames longer than 8 letters is possible, many programs depend on 8 letters being the maximum length of usernames, so one should adhere to this standard.

   Creating a username based on the real name of the user is advisable for practical purposes, but by no means a necessity.
3. The new user has to have a password. To set a password, one uses the command ‘passwd’. When called with no arguments, passwd defaults to changing the password of the user invoking the command. The user root can supply the name of the user whose password should be changed as an argument like this:

```
# passwd newuser
```

The command produces the following output:

```
Changing password for user newuser
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

The passwd command will ask for the password twice, to make sure it was spelled correctly.

4. If you wish to enter additional information about the new user, you can use the 'chfn' command with the username supplied. This command is an abbreviation for 'CHange FiNger information', since the preferred way of collecting this information later would be through the command 'finger'. If you are using NIS or 'yellow pages', prepend 'yp' to the command, making it 'ypchfn'.

```
# chfn newuser
Changing finger information for newuser.
Name []: New User
Office []: Asterisk
Office Phone []: 555-risk
Home Phone []: 555-home

Finger information changed.
```

We can now use the finger command to view the new information.

```
# finger newuser
Login: newuser                          Name: New User
Directory: /home/newuser                Shell: /bin/bash
Office: Asterisk, 555-risk              Home Phone: 555-home
Never logged in.
No mail.
No Plan.
```

2.4 Deleting a user

If for some reason you want to remove a user totally from you Trustix Secure Linux computer, you could use the command 'userdel' as root with the username as the only argument. If you also want to remove the users home directory, add the switch '-r'.

Example:

```
# userdel -r newuser
# finger newuser
no such user
```

2.4.1 Choosing a good password

The password is the last and often only line of defense for a user account, and should be treated thereafter. One should take due care never to choose a password that is easily guessable. Bad passwords include, but are not limited to:

- Your own username
- Own name or different variants thereof.
- Name of spouse, children, pets, or favourite food.
- Anything based on the keys on the keyboard, like asdfgh, qwerty1, and 1qweasd. (All these are real life examples that broke in seconds!)
- Anything containing a dictionary word in any language, including reversed words and words where some letters are removed or changed with numbers.

The length of a password has traditionally been limited to 8 characters, but as Trustix Secure Linux uses the more powerful MD5 algorithm instead of the traditional crypt(), limits to a passwords length are virtually removed. Some programs might limit the password to 127 characters, so one should probably not exceed this. A good length minimum length for a password is 8 characters, but longer passwords are better as long as one does not use something easily guessable from the above list.

As opposed to usernames, passwords should contain both upper and lower case letters and also numbers. Always remember that Unix passwords are case sensitive.

Last, here's a tip for generating a fairly secure password: Think of a phrase that you can remember easily like "My second boss had pointy hair and said 'Um..' a lot". Now take the first letter of each word, changing words like second into '2'. The phrase then translates to 'M2bhphasUal'. This would be a password that is fairly hard to guess, and at the same time quite easy for you to remember. Never use phrases that you often use in your daily speak for constructing such passwords.

It must also be mentioned that M2bhphasUal is no longer usable as a password as it has been published in this documentation. Think of a phrase yourself.

2.4.2 Changing password

A user should change his or her password at least once a year. To do this, the user logs in and issues the 'passwd' command like this:

```
$ passwd
Changing password for user
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

Note that the user first has to enter the current (old) password. Before the new password is entered twice.

2.5 Account Expiration

It is possible to set the expiry of both user accounts and passwords. This can easily be done with the command ‘chage’. You can get all the expiry information you need about a user by typing ‘chage -l username’. For privacy reasons, users can only specify their own username when doing this. The user root can specify any user. The output of ‘chage -l username’ can look something like this:

```
Minimum:        0
Maximum:        365
Warning:        -1

Last Change:            Jul 07, 1999
Password Expires:       Jul 06, 2000
Account Expires:        Nov 02, 2000
```

The ‘Minimum’ field tells us that the user can change his password anytime he wants to. You can set the minimum number of days between each password change for the user with: ‘chage -m [mindays] username’. A sane example could be ‘chage -m 7 username’. Here the user would have to wait one week (7 days) before the password can be changed again.

The reason for the minimum setting is that users often like to keep the same passwords for exended periods of time. When the time comes that they have to change it, they change to a temporary password, and then back again to the old one. If one specifies a minimum of days between each password change, the use of this technique is discouraged.

It is usual to set the maximum number of days a user can keep the same password without changing it. Frequently changing the password is a matter of system security. Setting the maximum value is very similar to setting the minimum value. All you have to do, is change the -m to -M, so the command line would look something like ‘chage -M 365 username’. This forces the user to change his password within a year.

The warning field specifies how many days before the password expires, the user should get a warning. In this case, a warning is never given. This can be changed easily by using ‘chage -W <DaysBeforeWarning> username’. This warning is given when users log in, so if they seldom or never log out, they will not recieve this warning. Setting this to 14 days could be a good idea.

The three last fields are mostly given away by their names:

- ‘Last Change’ indicates when the user’s password was last changed.
- ‘Password Expires’ is the date when the password will expire. This is the value of ‘Last Change’ plus the number from the Maximum field.
- ‘Account Expires’ is the date when the account will expire. This can be changed with the -E switch followed by a date specified to chage. The date should be in the format MM/DD/YY.

  ```
  Example: ‘chage -E 09/13/01 username’ will set the account ‘username’
  to expire on the 13th of September 2001. One can also specify a number
  of days since January 1, 1970 instead of a date.
  ```

2.6 Managing groups

Groups can be added and removed with much the same commands as users. To add a group, you use the command ‘groupadd’ followed by the name of the group.

Example:

```
# groupadd newgroup
```

Running the above example would generate an extra line in the file /etc/group along these lines:

```
newgroup:x:105:
```

This line consists of several fields separated with a ‘:’ The first field is the name of the group name, the second is the group password, the third is the numerical group ID or ‘GID’. The fourth field is a list of the users who are members of this group. In the above example no users are in the group, making it virtually useless. We add some users to it by writing the usernames in the fourth field separated by commas. In this example newuser and olduser have been added in the manner described earlier:

```
newgroup:x:105:newuser,olduser
```

Any number of users can be members of any number of groups.

When a user has been added to, or removed from a group, he will have to log out for the change to take effect.

To delete a group, use the command ’groupdel’ followed by the group name:

```
# groupdel newgroup
```

The above example issued as the user root will delete the group ’newgroup’ if it exists. If the group does not exist, the program will tell you so. Files belonging to this group will still be owned by this GID which is no longer bound to a specific group. Group ownerships of such files should of course be changed.

2.7 The basics of file permissions

Most Linux systems are designed for multiuser purposes, Trustix Secure Linux included. For the file system this means that features like file permissions and ownership are necessities. Users can control the access rights on their files and directories, defining who should is allowed to read, edit, and execute their files.

There are three different basic permissions: read, write, and execute. These permissions are defined on three levels: the user in ownership of the file, the group in ownership, and everybody else regardless of group.

The permissions mean the following:

- Read permission leat a user read the contents of the file, or in the case of directories, list the contents of the directory.
- Write permission lets the user write to and modify the file. For directories, this permission lets a user create, rename, or delete files within the directory.

- Finally, execute permission lets the user run the file as a program. This is only useful if the file is a binary program or a script file. For directories, a user will need execute permission to be able to use the directory as the working directory (cd into it).

Let's look at an example that demonstrates file permissions. Using the 'ls' command with the '-l' option displays a "long" listing of the file, including file permissions.

```
/home/larry/foo]# ls -l stuff
-rw-r--r--    1   larry   users         505 Mar 13 19:05 stuff
```

The first field in this listing represents the file permissions. The third field is the owner of the file (larry) and the fourth field is the group to which the file belongs (users). The last field is the name of the file (stuff). The other fields are not important in this example.

This file is owned by 'larry' and belongs to the group 'users'. The string '`-rw-r--r--`' lists in order, the permissions granted to the file's owner, the file's group and everybody else.

The first character of the permissions string (-) represents the type of file. A '-' means a regular file (as opposed to a directory or device). The next three characters (`rw-`) represent the permissions held by the file's owner, 'larry'. The 'r' stands for 'read' and the 'w' stands for 'write'. Thus, 'larry' has read and write permissions to the file 'stuff'.

As mentioned, besides the read and write permission, there is also the 'execute' permission — represented by an 'x'. However, a '`-`' is listed here in place of an 'x', so larry doesn't have execute permission on this file. this is fine, as the file 'stuff' isn't a program of any kind. Since Larry is the owner of the file, he may grant himself, or anybody else, execute permission for the file if he so desires. (This will be covered shortly.)

The next three characters (`r--`) represents the group's permissions for the file. The group that owns this file is 'users'. Because only an 'r' appears here, any user who belongs to the group 'users' may read this file. If the user wants to write to or execute the file, he must ask 'larry' to give him the appropriate rights.

The last three characters (also `r--`) represent the permissions granted to every other user on the system (other than the owner of the file and those in the group users). Again, because only an 'r' is present, other users may read the file, but not write to it or execute it.

Some more examples:

`-rwxr-xr-x` — The owner of the file may read, write, and execute this file. Users in the file's group and all others may read and execute the file.

`-rw-------` — The owner of the file may read and write to the file. No other user can access the file.

`-rwxrwxrwx` — All users man read, write to, and execute the file.

2.7.1 Changing file permissions

The command chmod is used to set the permissions on a file. Only the owner of a file may change the permissions on that file. the syntax of chmod is:

```
chmod {a,u,g,o}{+,-}{r,w,x} filename
```

Briefly, you can supply one or more of All, User, Group, or Other. Then you specify wether you are adding rights (+) or taking them away (-). Finally, you specify one or more of read, write, and execute. Some examples of legal commands are:

`$ chmod a+r filename` — Gives all users read access to the file.

`$ chmod og-x filename` — Remove execute permission from users other than the owner. (g=group, o=other)

`$ chmod u+rwx filename` — Let the owner of the file read,write and execute the file.

`$ chmod o-rwx filename` — Remove read, write and execute permissions from users other than the owner and the users in the file's group.

2.7.2 Changing the owner and group of a file

It is possible to change a file's owner and group. To do this, you can use the 'chown' and 'chgrp' commands (CHange OWNer and CHange GRoUp).

To change the owner of a file, one would write:

```
# chown newowner file
```

This gives the file ownership to a new owner. Please notice that for security reasons, only root is able to change the owner of a file.

Changing the group is just as easy:

```
$ chgrp newgroup file
```

Notice the usage of the 'chgrp' command instead of chown. For the same security reasons, users are only able to change the group of a file to a groups they are members of. The user 'root' can change the group of any file to any group.

2.8 Disk quotas

To ensure that a user does not use more disk space than he/she should, one can specify a maximum amount of disk space that the user can use at any time. A user using far too much disk space, could effectively jam a badly configured system, as he could fill a whole partition thus stopping anybody else from writing to the disk. A nice alternative to quotas is of course educating your users not to use up all the disk space, but accidents will happen even to the best, so some high quota could prove a nice 'safety net' in all cases.

The easiest way to start using quotas, is to specify for which filesystems quotas should be used in /etc/fstab. Let us say that you want to use quotas for the first partition on the first SCSI disk which is mounted as /home. The line in /etc/fstab for that device would then look something like this:

```
/dev/sda1 /home ext2 defaults,usrquota 1 2
```

Next, use the command 'quotacheck' to initialize the disk quotas for the file system:

```
# quotacheck -auvg
```

Next, turn on quotas with 'quotaon'

```
# quotaon -a
```

To edit the quota for the user 'username' on this partition you would issue the command 'edquota username'. This will put you into the 'vi' editor and enable you to edit the quotas for the user. It will look something like the folloing:

```
Quotas for user username:
/dev/sdb1: blocks in use: 40, limits (soft = 0, hard = 0)
inodes in use: 20, limits (soft = 0, hard = 0)
```

A quota can be specified for both inode use and block use. The most important limit would be the maximum number of disk blocks the user is allowed to use at any time. A disk block is the smalles number of bytes that can be allocated on the disk. To find out how big your disk blocks are, you can do the following:

1. `$ mkdir test`
2. `$ ls -ld test`

The output from ls could look something like this:

```
drwxr-xr-x   2 newuser  users        1024 Feb 17 10:04 test/
```

The number written right in front of the file creation date is the file size for files, but for directories, the block size is reported. The block size in this example is 1024 bytes or 1Kbyte. If we want the user to be able to use 50MB, but recieve a warning when using more than 40MB, we would edit the lines in edquota to the following:

```
Quotas for user username:
/dev/sdb1: blocks in use: 40, limits (soft = 40000, hard = 50000)
        inodes in use: 20, limits (soft = 0, hard = 0)
```

3 Windows File and Print Services

The windows fileserver capability of Trustix Secure Linux is handled by a program called Samba. This package can take care of:

- File and print services.
- Authentication and authorization
- Name resolution
- Service announcement (Browsing)

In other words: Samba allows you to move your computer running Trustix Secure Linux into your windows "Network Neighbourhood" without causing a stir. It allows the windows machines, and other computers also running Samba, to access files and print documents neither knowing nor caring that they are being served by a Linux host.

This is managed by the protocol suite CIFS (Common Internet File System), a name introduced by Microsoft. At the heart of CIFS, you find SMB (Server Message Block), the cornerstone of file and printer sharing in a windows network.

Samba is an open source implementation of the CIFS protocol suite, and actually a quite fast implementation, too. When Samba 2.0 was released in January 1999, one of the most important improvements was speed. Ziff-Davis Publishing used their Netbench software to benchmark Samba 2.0 on Linux against Windows NT4. They ran all of their tests on the same PC hardware, and their results showed Samba's throughput under load to be at least twice that of Windows NT.

More information about Samba can be found at http://www.samba.org

3.1 Installation

If you didn't install the fileserver package during the installation of Trustix Secure Linux, you can easily install it now. Download or locate on your CD, a file named samba-2.0.2tr.i586.rpm (the version number might differ). Just install it as described earlier in this document.

That's it. Now on to some basic configuration.

3.2 Configuring Samba

The configuration file for Samba is called smb.conf and resides in the /etc directory. The default configuration on install is to have the following capabilities:

- Each user gets his own home directory mapped up.
- Encrypted passwords are default. This means that you will not be able to connect with Windows 95 without an SMB update, Windows NT 3.x, and Windows NT 4 with service pack 1 or 2 (SP 3 and up are OK).
- Printer configuration is read from /etc/printcap. Printers configured for the system are automatically shared.

3.2.1 Setting the workgroup

Before you start using Samba, you should set the server's workgroup. You will have to be root to do this. The configuration of the Samba fileserver is set in the file /etc/smb.conf. The workgroup is set in the [global] section of this file:

```
[global]
workgroup = workgroupname
```

'workgroupname' should be replaced with the name of the workgroup that the server should be in.

Now you have to restart the Samba daemon to make it reread the configuration file. Still as root, issue the following command:

```
# /etc/rc.d/init.d/smb restart
```

3.2.2 Adding users

Samba users are added with the smbadduser program. Do that like this:

```
# smbadduser newuser
```

The user must also have a Samba password set. This is set with the program smbpasswd like this:

```
# smbpasswd newuser
```

Follow the prompts.

Note that all Samba users must first have a valid user account on your system. This is discussed in the user administration chapter.

3.2.3 Usernames

Client usernames on a SMB network can have up to 255 characters. On a Unix system this will generally be limited to 8 characters. This means that a user can have one long username on the client and another short one on the Samba file server. To avoid this problem, you can use a username map file. Samba needs to be told where to find this in the [global] section of smb.conf:

```
[global]
username map = /etc/samba/usermap.txt
```

Be sure to restrict access to this file so that only root is able to edit it.

The file's syntax is like this:

```
johnd = JohnDoe
users = @account
nobody = *
```

- Wildcards (*) are used to match any free form client username in the username map file.
- @ is used to map a NT group to a Unix group.
- ; and # are comments

3.2.4 Username level

Windows can often send usernames entirely with capital letters. Windows usernames are not case sensitive, however Unix usernames are. USER and user would be two different accounts on a Unix system while on a windows system they would be the same. As a solution to this, Samba does the following:

1. Check for a user account with the exact name sent by the client.
2. Test the username with all lowercase letters.

3. Test the username with the first letter capitalized

If you want Samba to try more combinations of lower and uppercase letters, you can specify this with the global option username level.

```
[global]
username level = 3
```

With this value, Samba would try all permutations of the username haveing three capital letters.

3.3 Plain text passwords

To be able to use old windows versions as clients for Samba, you will have to enable plain text passwords. As you have chosen Trustix Secure Linux, we believe that you value the security of your system. We therefore recommend that you upgrade your clients instead of lessening security, but if you have a specific reason for using old clients, Samba also has support for unencrypted passwords. To enable this, you will have to make some small changes in the file /etc/smb.conf:

```
[global]
security = user
encrypt passwords = no
```

Restart Samba to reflect these changes:

```
# /etc/rc.d/init.d/smb restart
```

To make clients that use encrypted passwords use plain text passwords instead, locate and run the one of the two scripts NT4_PlainPassword.reg and Win95_PlainPassword.reg that fits your system. The one for Win95 also works with Win98. These files can be found on the Trustix Secure Linux system in the directory /usr/doc/samba-2.0.7/docs. The client computers need to be rebooted after running the scripts for the changes to take effect.

3.4 WINS

In the old days, before NetBIOS name servers (NBNS), name resolution worked entirely by broadcast. This approach is not possible if your domain spreads over different subnets, because the broadcast would be stopped by the routers. To overcome this problem, Microsoft provides WINS, Windows Internet Naming Service. Name registration and resolution requests can be directed to one computer in the network instead of using the awkward broadcast mechanism.

Your Trustix Samba server is by default set up as a WINS server and primary domain controller in your network.

3.4.1 Setting up Samba as a WINS server

To set up Samba as a WINS server one makes sure that the following is present in the file /etc/smb.conf:

```
[global]
wins support = yes
name resolve order = wins lmhosts hosts bcast
```

This is all you have to do. The name resolve order tells Samba how to resolve NetBIOS names. The values mean the following:

- `wins`: Use the wins server
- `lmhosts`: Use a LAN Manager LMHOSTFILE
- `hosts`: Use standard Unix name resolution methods. (/etc/hosts, NIS, DNS)
- `bcast`: Broadcast method.

3.4.2 Using another WINS server

This should be easy. You just tell Samba where the other WINS server is:

```
[global]
wins server = 192.168.44.55
```

This would make Samba redirect all WINS requests to the server at 192.168.44.55.

3.5 Logon scripts

Samba supports the execution of Windows logon scripts. These scripts are stored on the Samba server and transported to the client and executed there once the user logs on.

This sets up Samba to use logon scripts:

```
[global]
domain logons = yes
security = user
workgroup = SOME WORKGROUP
os level = 34
local master = yes
preferred master = yes
domain master = yes
logon script = %U.bat

[netlogon]
comment = The domain logon service
path = /export/samba/logon
public = no
writeable = no
browsable = no
```

The the user johnd logs in, the server will look for johnd.bat. These scripts must reside in the netlogon base directory, in this case /export/samba/logon.

4 Web Services

4.1 Introduction

The web server we have chosen to distribute with Trustix Secure Linux is called Apache. It was originally based on a program called NCSA httpd 1.3. It has since then evolved into a much superior product rivalling, and probably surpassing, any other Unix based web server in terms of functionality, efficiency, and speed. The program is running on an ever increasing number of internet servers, its user group showing no signs of diminution.

4.2 Security

As in all of our other services, we have focused on security when making the default configuration for the Apache server distributed with Trustix Secure Linux. We have included software suitable for providing secure http connections, and made some security improvements on the default configuration.

We have chosen to disable the possibility for running cgi scripts in the default configuration. If you need to run such scripts it is easily enabled, but beware of the security concerns it causes. As cgi scripts are programs run by the server every time somebody connects to them with their browser, they run the cgi program. As very few programs, if any at all, are perfectly programmed one risks the exploitation of security holes by crackers to gain access where no crackers should be.

If you find that you need cgi scripts, you should know that Trustix Secure Linux comes readily configured to use the SuEXEC feature provided by Apache. SuEXEC is a wrapper that makes cgi scripts run with the privileges of the owner of the script, as opposed to the privileges of the user running the server, most often root or the user nobody. Using SuEXEC means that an insecure script is more likely to only damage the data belonging to the user running it.

Although it provides some useful features for editing web pages, we have chosen not to include the Microsoft FrontPage Extensions in Trustix Secure Linux. This is because even though it can make some things easier, it has so many security holes that it is not something you would put on any server you care about. We believe most people would rather use five minutes more to edit those web pages than wait hours while the admin is restoring the system from backup because the system was cracked.

4.3 Installation

If you did not select the option to install the web server when you installed Trustix Secure Linux, you will have to do so manually. The package you need for this is the apache rpm package from the install CD or a Trustix mirror. Se the introductory chapter for how to install an rpm package.

The main apache files are installed in /home/httpd, its modules are installed in /usr/lib/apache. Some files are also put elsewhere on the disk.

4.4 Overview of /home/httpd

The directory /home/httpd contains several directories that are more or less integral to Apache. These are: cgi-bin, html, and icons. Their uses are in short terms as follows:

`cgi-bin` — This directory contains the cgi-scripts that are to be used by the system. As a security precaution, it can sometimes be wise to allow only the cgi script residing in this directory to be run. This would enable users only to run cgi scripts that have been checked and approved of by an administrator. This would provide adequate security while still allowing the functionality of cgi scripts. Approved cgi-scripts put in this directory could also be a valuable addition for users even if they are allowed to roll their own scripts.

`html` — The html directory is the topmost directory for web documents - The so called DocumentRoot. When someone writes the URL http://your_server.com in their browser, the web server responding to your_server.com would read /home/httpd/html/index.html and send that back over the link. The URL http://your_server.com/foo/bar.html would in this example correspond to the file /home/httpd/html/foo/bar.html.

This directory also contains a comprehensive manual for the Apache web server.

`icons` — The directory 'icons' is not really very important to the functionality of the apache server. It only contains a lot of small bitmap pictures that can, as the name implies, be used as icons on web pages. It uses little disk space, its presence poses no security risk, and having access to some icons to put on a page can often be quite practical. These icons are also used when listing the contents of a directory. In addition to this, removing the icons when upgrading could break web pages already dependent on them. For all these reasons, we have chosen to include these icons in Trustix Secure Linux.

Use rpm to get a full list of all files installed with the Apache package. The command would be:

```
$ rpm -ql apache
```

4.5 Configuration

To configure the Apache web server, you will typically have to edit its configuration files which are contained in /etc/httpd/conf. The package comes with a good default configuration, but there are certain things that you should be aware of, and some defaults you might want to change. The files are also heavily documented through comments, so you should have few or no problems understanding what the various options do.

Remember to restart the http daemon when you have made changes to the configuration files. This can be accomplished using the following command:

```
# /etc/rc.d/init.d/httpd restart
```

4.5.1 CGI scripts

For reasons already mentioned, the default configuration in Trustix Secure Linux is not to allow the usage of cgi scripts. If you are sure you want to allow this, you should edit the file access.conf and find the block starting with <Directory /home/httpd/cgi-bin>. To allow the execution of cgi scripts, the block should look like this:

```
<Directory /home/httpd/cgi-bin>
AllowOverride None
Options ExecCGI
</Directory>
```

The "Options" line is used when allowing types of access to directories. The option "ExecCGI" grants the right to execute cgi scripts in this directory.

You should also make sure that the cgi module is properly loaded. The file httpd.conf contains many lines starting with LoadModule and Addmodule. You must make sure that the to following lines are present in the appropriate places:

```
LoadModule cgi_module         modules/mod_cgi.so
AddModule mod_cgi.c
```

The last thing that must be done to allow execution of cgi-scripts is making sure the following line is present in the file srm.conf:

```
AddHandler cgi-script .cgi
```

If you need to allow execution of cgi scripts from other places than the cgi-bin directory, typically to allow users to create and use their own cgi scripts, you should only have to add the option "ExecCGI" to the "Options" line of the particular directory block in the config file.

To enhance the security of the web server while still allowing the usage of cgi scripts, Trustix Secure Linux comes readily configured with the SuEXEC wrapper. This wrapper will be used every time a cgi script is run.

4.5.2 Virtual servers

One computer running the Apache web server can be set up to host the web pages of several domains. It can for example be set up to show a different page for http://www.your_server.com, http://security.your_server.com, and http://www.something_else.com. Making several DNS names point to the same computer can be done in any number of ways, all beyond the scope of a short Apache manual. We refer instead to Apaches own documentation which can be found at http://www.apache.org/docs/vhosts/index.html, and Glenn Stevens' extensive DNS guide at http://eeunix.ee.usm.maine.edu/guides/dns/dns.html.

When you have set up the different DNS entries to point to your server, you will have to edit the file /etc/httpd/conf/httpd.conf. For each virtual server you should add an entry like the following with the appropriate names changed to fit your setup (lines starting with a # are comments and can be removed):

```
<VirtualHost host.your_server.com>
ServerAdmin webmaster@host.your_server.com
DocumentRoot /home/httpd/html/host.your_server.com
ServerName host.your_server.com
ErrorLog /var/log/httpd/host.your_server.com-error_log
TransferLog /var/log/httpd/host.your_server.com-access_log
</VirtualHost>
```

ServerAdmin should be set to the e-mail address of the person responsible for the web server. DocumentRoot defines where the server should search for the pages to display given the specific ServerName which is defined in the nest line. The ErrorLog and TransferLog files are where the web server will put its logs when serving pages for the ServerName. The placement of these log files should concur with the rest of your servers setup.

4.5.3 Other options

The Apache web server is extremely configurable. There are many options that deserve mention while still not needing their own chapter. We will attempt to describe the most important of them here.

DocumentRoot set in srm.conf tells Apache where to look for web documents on the disk. Should you for whatever reason want to put these files somewhere else than in the default /home/httpd/html, this is where you inform Apache of the change.

Users can make their own web pages available as http://your_server.com/ username by putting them in a specified directory inside their home directory. The name of this directory is specified in the variable UserDir set in srm.conf. The default here is public_html. Note that 'other' must be given execute rights to both the users home directory and public_html for this to work.

In the file httpd.conf, the variable ServerAdmin should be set to the e-mail address of the person in charge of the web server. The value of ServerAdmin set outside any VirtualHost block will also be the default value when no other value is specified inside the block.

Apache is a multithreaded server, where each thread communicates with one client. The values of MinSpareServers, MaxSpareServers, StartServers, and MaxClients are all used to define limits for the number of processes. The values MinSpareServers and MaxSpareServers define the minimum and maximum values of threads not currently serving a client. If this number goes outside the limits defined by these values, extra threads are spawned or killed until the number of spare servers again is inside these limits. StartServers defines how many spare servers should be spawned when Apache is started. A sane value should be somewhere between MinSpareServers and MaxSpareServers. MaxClients defines the maximum number of clients that can be served at one time. One should take care not to set this limit too low, as this would result in users not being able to view ones pages. Experimentation is required to find the ideal values for these options. The default values will be adequate in most cases.

4.6 Log files

For security, debugging and other various reasons, the Apache web server creates extensive log files in the directory /var/log/httpd. To simplify searching, the date and time are part of every log entry. If you are having problems with the web server, checking the logs from the appropriate time is a good place to start.

The access log contains an entry for every page that has been downloaded, and can be used for statistical purposes. There are several programs that can do this collecting of statistics automatically. Among the more popular is The Webalizer by Bradford L. Barrett. (http://www.mrunix.net/webalizer/)

5 E-Mail Server

Mail server software enables sending and receiving e-mail between users and programs. Servers that are not intended as mail servers should nevertheless have mail server software installed so that other running services are able to send messages to their administrator.

Trustix has selected the mail transfer agent (MTA) called Postfix as its mail server program. This is because Postfix is designed with security in mind. We also include Washington University's IMAP and POP daemons.

If you are setting up a server to handle users' e-mail, consider adding IMAP and/or POP software. Both IMAP and POP are protocols used for reading e-mail over a network or dialup connection. Using POP, the users may retrieve mail and store it locally on their computer, while IMAP lets users store their e-mail folders on the mail server, enhancing portability, and saving local disk space.

Secure IMAP (SIMAP) is normally preferred to standard IMAP, since the data (people's passwords and e-mail) is encrypted when sent over the network when using this protocol.

The rest of this chapter describes the installation and configuration of the e-mail services available on Trustix Secure Linux, and the administrative tasks typically performed.

5.1 Installation and first time configuration

If you didn't select Mail Server when you installed Trustix Secure Linux, you can install it later by mounting the CDROM or downloading the portfix rpm from FTP or HTTP install sites.

You should also make sure you run postfix by using the command 'chkconfig postfix on'.

Through this section, the referenced files will be located in /etc/postfix unless otherwise specified.

5.1.1 Forwarding root's mail

Mail sent to the root user should be forwarded to the system's administrator. Edit the aliases file and find the following line:

```
#root: you
```

Remove the '#' and insert your e-mail address instead of 'you'. Whenever the aliases file is changed, run the command 'newaliases' as root. This is actually quite important, as postfix does not like to deliver mail to root and will instead just let the mails lie in the spool.

5.1.2 Setting up mail domain names

Usually, you will want to specify that mail sent from your server appears to be coming from the domain instead of that particular computer, i.e. your from-address should be user@mycorp.com instead of user@mailserver.mycorp.com. Edit the file main.cf and uncomment the following line:

```
#myorigin = $mydomain
```

If your server is to handle mail for your domain, you need to add the following line to main.cf:

```
mydestination = $myhostname,localhost.$mydomain,$mydomain
```

If you want your server to accept mail for other domains, add those domains to the same line separated by commas, e.g:

```
mydestination = $myhostname,localhost.$mydomain,myotherdomain.com,mycorp.org
```

5.1.3 Setting up a mail server for send-only

If your mail server is not supposed to receive mail for a domain, the typical configuration is to set it up as a "null client". A null client does not receive mail, and delivers no mail locally.

To set up a null client, add the following lines to main.cf:

```
myorigin = $mydomain
relayhost = $mydomain
```

In addition, to prevent mail from being delivered locally, edit the file master.cf, and comment out (prefix with '#') the following lines (Note: they are not necessarily in this order!):

```
smtp      inet  n       -       n       -       -       smtpd
smtp      unix  -       -       n       -       -       smtp
local     unix  -       n       n       -       -       local
```

5.2 Forwarding mail

The Postfix mail transfer agent can forward mail from one account to another e-mail address, or receive mail for an adress that has no matching username on the system. The most common way to attain this is to use the 'aliases' file we used earlier to forward roots mail.

To make the system receive mail for a username that does not exist on the system and forward it to some valid e-mail address, you would add the following line to the aliases file:

```
nonexist: someuser
```

This makes the system receive mail addressed to the non existent user 'nonexist' and forward it to the user 'someuser'. The user someuser must exist or can also be an other alias created in the file. Remember to always run 'newaliases' after changing the aliases file, as postfix needs this to know about any changes you have made.

It is also possible to forward the mail to more users with a line like this:

```
nonexist: someuser, otherusr
```

You can add as many comma separated usernames or valid e-mail addresses to this line as you like.

Users can also forward their own e-mail some other account. They can do this by putting a file called '.forward' in their home directory and merely writing the other adress they want the mail to be forwarded to in this file. It is also possible to specify a file that mail should be delivered to.

Example .forward:

```
/home/user/Mailbox
user@work.com
777@mail2sms.com
```

This example file would make all mail be saved in the file /home/user/Mailbox, and would also forward the mail to both user@work.com and also the users mobile phone through a (fictional) mail2sms gateway.

5.3 Reading mail

To read mail over the network using other computers as clients, you will need to run one or both of the pop and imap daemons. To do this, you need to alter one or two lines in the file /etc/inetd.conf. Open this file as root in an editor and find the following block:

```
# Pop and imap mail services et al
#
#pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd ipop2d
#pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd ipop3d
#imap    stream  tcp     nowait  root    /usr/sbin/tcpd imapd
```

You should remove the prepended hashes from the appropriate lines for the services you want to offer. If you want users to be able to use the imap-protocol, which is supported by most major mail programs like Netscape Messenger, Microsoft Outlook and Emacs/Gnus, you would change the block to look like the following:

```
# Pop and imap mail services et al
#
#pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd ipop2d
#pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd imapd
```

You will have to restart inetd with '/etc/rc.d/init.d/inet restart' for the changes to take effect. If inetd is not already automatically started every time you reboot your computer, you should run 'chkconfig inet on'.

5.4 Encrypted mail exchange

The program 'stunnel' which comes with Trustix Secure Linux is able to create SSL encrypted tunnels for services that are normally run through inetd. This is a recommended substitute for running certain services like imap and pop3 unencrypted.

Before using stunnel, you have to create a site certificate as described in the introductory chapter.

Stunnel can be started manually or from inetd. The command for starting imap tunneling program manually is:

```
stunnel -p /etc/ssl/certs/server.pem -d simap -l /usr/sbin/imapd
```

The '-p' switch tells the program where it should look for its certificate, the '-d' switch makes it run as a daemon on the 'simap' port, and the '-l' switch says what program should be tunneled.

This command can be put in a startup script like /etc/rc.d/rc.local to make it start every time you boot the computer.

6 The DNS name server

If you want to use your server as a name server, you will be happy to know that Trustix Secure Linux comes with the program called the Berkeley Internet Name Daemon, or bind for short.

6.1 Setting up a caching only name server

A chaching only name server will find the answer to name queries and remember the answer for the next time you need it. This will shorten the waiting time the next time, especially with a slow connection.

Bind comes already configured to act as a caching name server. All you have to do is start it with '/etc/rc.d/init.d/named start', and add it to your default configuration with 'chkconfig named on'.

To make your system actually use the name server, you will also have to make a few changes to the file /etc/resolv.conf. This file should have been created during the installation, and should look something like this:

```
search your-domain.com other-domain.com
nameserver 192.168.1.10
```

The hostnames and IP-address should match your setup. The system should be told to use 127.0.0.1 as nameserver, making the file look like this:

```
search your-domain.com other-domain.com
nameserver 127.0.0.1
nameserver 192.168.1.10
```

6.2 Setting up a name server for a domain

While bind can be nice for just caching the output from other name servers, it is even better as a name server for serving your domain. In this example, we will set up the computer at a master server for 'your-domain.com'. This name and all IP-addresses should be changed into whatever is appropriate for your setup.

First, insert the following block into the file /etc/named.conf:

```
zone "your-domain.com" {
notify no;
type master;
file "your-domain";
};
```

The line 'notify no;' tells bind not to notify slave servers when zone files are updated. If you have slave servers, you would probably want to change the 'no' to 'yes'.

Then, put the following into the file /var/named/master/your-domain:

```
;
; Zone file for your-domain.com
;
; The full zone file
;
@       IN      SOA     ns.your-domain.com. hostmaster.your-domain.com. (
                        200002151       ; serial, todays date + todays serial
#
```

```
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
;
                NS      ns              ; Inet Address of name server
                MX      10 mail.your-domain.com.  ; Primary Mail Exchanger
                MX      20 mail2.your-domain.com. ; Secondary Mail Exchanger
;
localhost       A       127.0.0.1
ns              A       192.168.196.2
mail            A       192.168.196.4
mail2           A       192.168.196.7
```

All hostnames and IP addresses should be modified to fit your desired net setup.

Two things must be noted about the SOA record. ns.your-domain.com must be an actual machine with a A record. It is not legal to have a CNAME record for the machine mentioned in the SOA record. It's name need not be 'ns', it could be any legal host name. Next, hostmaster.your-domain.com should be read as hostmaster@your-domain.com. This must be a mail alias or mailbox where the person in charge of DNS should read mail frequently (typically, you would set up 'hostmaster' as an alias for the person in charge in the file /etc/postfix/aliases).

The MX lines in the file tells where mail addressed to someone@your-domain.com should be delivered. First the number before the hostname tells the priority of the servers. When mail is sent in this example, delivery would first be attempted to mail.your-domain.com, as this name has the lowest number, and thus the highest priority.

If you want to have several names for the same machine, you would use the CNAME record. The following line would be appropriate if your name server is also your web server:

```
www CNAME ns
```

Note that a name *not* ending in a dot ('.') will be expanded with the default hostname (in this case your-comain.com). A name that ends with a dot, will not be expanded. Thus the above example could also have been written as:

```
www.your-domain.com. CNAME ns.your-domain.com.
```

Next, you should add the reverse zone. This allows the conversion from a IP address to a DNS name. Put the following into /etc/named.conf:

```
zone "196.168.192.in-addr.arpa" {
notify no;
type master;
file "your-domain.rev";
};
```

Note that in the IP address area before the .in-addr.arpa the numbers are supposed to be in the opposite direction from the "real" IP addresses (compare with the addresses in /var/named/master/your-domain). Put the following into the file /var/named/master/your-domain.rev:

```
@       IN      SOA     ns.your-domain.com. hostmaster.your-domain.com. (
                        200002151 ; Serial, todays date + todays serial
                        8H      ; Refresh
                        2H      ; Retry
                        1W      ; Expire
                        1D)     ; Minimum TTL
                NS      ns.your-domain.com.

2               PTR     ns.your-domain.com.
4               PTR     mail.your-domain.com.
7               PTR     mail2.your-domain.com.
```

If this is all done, you can restart bind to make it reread the configuration. The preferred way of doing this is running 'ndc restart'.

7 Print services

The print services of Trustix Secure Linux are taken care of by a set of tools called LPRng. Using these tools, your server can be configured as a remote print server. LPRng is also used by Samba for its Windows print services, and it is therefore a necessity to configure this to act as any kind of print server.

To be able to use any configured printer, one must be sure to run the printer daemon, lpd. To make this be started every time the computer is restarted, use chkconfig ('chkconfig lpd on'). Every time any of the configuration files are changed, lpd should be restarted with '/etc/rc.d/init.d/lpd restart'.

7.1 Configuring a printer

The printer configuration is set in the file /etc/printcap. The format of this file was originally based on the termcap file format, which can make it a bit hard to understand without some work. In this document, we will only provide some examples and try to explain them. The examples and explanations can also be found on the LPRng home page.

We start out by defining a printer connected to the parallel port, as this printcap entry is very simple:

```
# parallel printer
     lp:
     :lp=/dev/lp0
     :sh:sf
```

- The first line is a comment describing the printer.
- Second is the printer name, 'lp'
- Third, we tell the program to use the device /dev/lp0
- The 'sh' and 'sf' will prevent lpd from trying to create banner pages or form feeds between jobs.

If you discover that Unix jobs result in a 'staircase' appearance, then you need to force the printer to do LF (linefeed) to CR/LF (carriage return/line feed) conversion.

```
# simple parallel printer
     lp:
     :lp=/dev/lp0
     :if=/usr/libexec/filters/lpf
     :sh:sf
```

The lpf filter specified in the 'if=' line will do LF to CF/LF conversion.

If you want to print to a remote printer or print server which supports the RFC1179 protocol, you can use the following printcap:

```
# simple remote printer
remote:
     :lp=raw@serverhostname
```

If this printer or remote print server does not do its own formatting, you might consider adding the 'if=' line from the last example to the above example.

For more information about creating a printcap entry, and also quite a bit of other information, see the LPRng home page http://www.astart.com/lprng/LPRng.html or the Linux Documentation Project at http://www.linuxdoc.org

7.2 Running the print server

The LPRng daemon supports running as a print server for remote systems, but is per default set to deny any connections from other hosts. To allow connections from the outside, this has to be defined in the file /etc/lpd.perms.

Example lpd.perms file:

```
# allow root on server to control jobs
ACCEPT SERVICE=C SERVER REMOTEUSER=root
REJECT SERVICE=C
#
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M
# all other operations allowed
DEFAULT ACCEPT
```

This config file would allow all remote hosts to send print jobs to your host, which would probably be a bad idea. You should add a line like this:

```
REJECT NOT REMOTEHOST=*.ourname.com,*.friendly.com
```

This way connections from computers that do not have DNS entries ending in ourname.com or friendly.com are rejected.