SuSE Linux 7.1 Professional Installation Checklist

written by Felix Schallock

This guide is intended as an extended administrative checklist for a secure installation of SuSE Linux 7.1 Professional (www.suse.com) running on a Dell PowerEdge 2400. The computer should serve as web server and should provide content service, upload capabilities for updating web pages and remote administration capabilities via openssh [1]. This server will function as a DNS client to be able to resolve IP- Addresses and host names. Other services are not envisaged but may be added if required. General explanations about how to secure additional services will be provided. Anyone with the need to provide information using a web server might use this setup. However a basic understanding of Linux/Unix and networking principles is required.

The server used for this setup consists of one 866 MHz Pentium III CPU, 256 MB of RAM, two hot swap capable 9,1 GB Ultra/160 hard disks mirrored (hardware RAID -1), a SCSI CDROM and a HP DDS2 SCSI tape drive for backup purposes and a floppy disk drive.

[1] Openssh, a tool that provides an encrypted connection. URL: www.openssh.org

Table of contents

1. Initial BIOS Settings 4

2. Operating System Installation 4

Partitioning 4

Choosing Software packages 5

Configuring system components 7

3. Updating the system 7

General Information 7

Disabling Services 7

Updates 9

Additional security Programs 10

4. Configuring the System 10

Hardening 10

Configuring Apache 14

Web server protection on file system level 27

Configuring Openssh 29

Tcp wrapper 33

Global configuration 34

SuSEfirewall2 configuration 36

Configuring secumod 40

Configuring sudo 41

Configuring password policy 42

Configuring Syslogd 42

Configuring the backup 43

Tripwire configuration 44

Configuring lilo 48

Last configuration step 48

5. Testing the configuration 49

External test 49

Functionality Test 50

6. Periodical review / updates 51

Seccheck 51

7. Comments 54

8. Appendix 56

Examples 56

References:__________________________________ ____________________________ 64

1.Initial BIOS Settings

First of all you have to setup a BIOS password to protect the settings from b eing altered. This password should be different from any password you might want to use for user accounts. Depending on your version and manufacturer you will be prompted to press a key (e.g. "DELETE", "F1", "F2" or "F10") at boot time to gain access to th e BIOS. You should find an entry that is labeled BIOS password or similar where you can set your BIOS password. Don't use the boot password option, unless you are sure you will never need to reboot the machine remotely!

Make sure that the boot sequence se tting in the SCSI controller BIOS allows booting from CDROM, since it is necessary for the installation to boot from a CDROM.

Section	Task to do	Status
1.1	Set BIOS password	
1.2	Assure booting from CDROM is enabled	

2.Operating System Installation

Prior to starting with the installation process be certain that your server is not physically connected to a network.

Start the installation by booting from CDROM No. 1 and have a look at the SuSE installation guide which is either sent to you with the SuSE Linu x package you ordered or is included on the CDROM No.1 (directory /doc/).

If you have difficulties with your hardware, check the hardware knowledge database at the SuSE website (http://cdb.suse.de/cdb_eng lish.html), which can provide information about the supported hardware and special instructions that might be needed. If your hardware is not listed, try a manual setup procedure by typing *manual* at the boot prompt and try to autoprobe the modules.

Section	Task to do	Status/Value
2.1 2.2	Physically disconnect the computer from the network Boot from CDROM No.1	
2.3	Note the OS version you are about to install	

Partitioning

The number of partitions you create is up to you, however I recommend using seven par titions /, /boot, /var, /tmp, /home, /usr/local/httpd, /usr/local and one swap partition which should be a little bit larger than your physical RAM. The /usr/local/httpd partition is used for the apache web server, since it is the default directory for apa che. The space you need to allocate for these partitions depends upon the software you need to install for your purposes.

Note: There are many different opinions about the size of the swap partition, but be sure that you use one at least.

The file system you use is up to you (reiserfs, ext2), except for the /boot partition that should have an ext2 file system. This is an example of how your partitioning could look like:

Partition	Mount Point	File system	Used for	Size in % of overall hard disk capacity (at least 5GB)
swap		swap	Swapping of Memory	0,5
/dev/sda1	/boot	Ext2	Storage of bootkernel	0,1
/dev/sda3	/usr/local/httpd	Ext2	Apache web server related files	20
/dev/sda4	/var	Ext2	Log, spool and configuration files	20
/dev/sda5	/tmp	Ext2	Storage of temporary files	5
/dev/sda6	/home	Ext2	User home directories	5
/dev/sda7	/usr/local	Ext2	Source packages, compiled programs	5
/dev/sda8	/	Ext2	The rest	44,4

Section	Task to do	Status/Value
2.4	Create a separate partition for /usr/local/httpd	
2.5	Document the partitions you created	
2.6	Indicate the file systems for your partitions	

Choosing Software packages

Before you can choose your software packages you have to decide whether you want to use the capabilities of the kernel 2.4 series or the kernel 2. 2 series. The kernel 2.4.x allows you to use iptables, which is state full [2], for a local firewall on your system (SuSEfirewall2), but you can't use secumod, which provides additional security features for hardening your OS, only while using the kernel 2.2. x. With the kernel 2.2.x you can use a local firewall (SuSEfirewall) too but this one is only based on ipchains (lacks state full inspection).

Summary:
Kernel 2.2.x: ipchains (SuSEfirewall, not state full), secumod (extra OS hardening)
Kernel 2.4.x: iptables (SuSEfirewall2, extended firewalling capabilities, state full)

You can choose from several options to install software packages. Install at least the following packages (depending upon the usage you intend) [3]:

- Choose the "minimum System" as the basis

[2] state full means that the firewall keeps a list of connections and in which state they are e.g. connection request(SYN) was sent from host x to host y. Following this list the firewall is able to determine what the next packet should look like (SYN,ACK) and can therefore deny strange packets easily. For further information have look at Rusty Russell's "Linux Packet Filtering HOWTO", v.1.21, at http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc.html

[3] This selection is partially derived from those made by Marc Heuse in his work „Installation of a Secure Webserver", differences do exist.

- Add the following separate packages
  - From Basis (a): compat
  - From Applications (ap): sudo
  - From Network (n): apache, bindutil
  - From Security (sec): iptables (or ipchains), mod_ssl, scanlogd, seccheck, secumod (2.2.x kernels only), ippl and tripwire
  - From Commercial (pay): arkeia
  - Anything else you might want, but you don't need to select further packages.

Please remind that the more software you install, the more possibilities exist to compromise your system. Choose your packages wisely. Any dependent packages will be selected automatically.
The "compat" package provides libraries for compatibility with older packages. The package "sudo" includes a program for providing the ability to grant rights to normal users to execute specific programs as root. The "apach e" package contains the apache web server and related scripts. The "bindutil" package contains utilities for queering domain name servers. The package "iptables" ("ipchains") contains user space programs to manipulate the firewall rules for 2.4.x kernels (2.2.x kernels). The "mod_ssl" package provides SSL (Secure Socket Layer) functionality to the apache web server, enabling it to use an encrypted connection to exchange information between the server and the web browser. The "scanlogd" package contains a daemon that is able to detect port scans and it informs you about occurrences of those. The "ippl" package provides an IP protocol logger that can log incoming ICMP, UDP datagrams and TCP connections. The "seccheck" package contains scripts that will get ex ecuted automatically by cron daily/weekly/monthly and do security checks concerning passwords, users, devices and sockets and will send you an e -mail containing changes that occurred since the last run of the scripts. This enables you to have a summary abo ut changes that have occurred on your system periodically. The "tripwire" package contains a program that enables you to track changes made to the file system and therefore being able to detect unauthorized manipulations in a timely manner. The "arkeia" pa ckage includes backup software that is easy to set up and free for use on Linux systems.

Section	Task to do	Status/Value
2.7	Choose "minimal System" as Software basis	
2.8	Add the separate packages: (a) compat (ap) sudo (n) apache, bindutil (sec) mod_ssl, sc anlogd, seccheck, secumod (2.2.x kernels only), ippl, iptables (or ipchains) and tripwire (pay) arkeia	
2.9	The setup will automatically select dependent packages, be sure to not change the selection.	
2.10	Install the 2.4 series kernel or the 2.2 series kernel depending on your preference (iptables/ipchains, -/secumod)	
2.11	Install lilo[4] in the Master Boot Record (MBR)	
2.12	Provide a strong password for the root user (it should consist of numbers, letters and special characters with a length of 8 characters)	

[4] Lilo is a boot manager allowing you to boot Linux and to choose between different kernels and options.

Configuring system components

The setup routine will automatically detect your networking device, as long as it is supported. If you should have difficulties with your network card then consider having a look at the SuSE hardware support database (http://cdb.suse.de/cdb_english.html).

Section	Task to do	Status/Value
2.13	Choose activate network	
2.14	Configure the following network settings:	
	Hostname	
	IP-Address	
	Netmask	
	IP-Address of default gateway IP-Address of DNS[5]	

3.Updating the system

General Information

First of all I want to point you to some general information about SuSE since this information will be of value for you.

The documentation for all packages that are installed on a SuSE Lin ux system can be found in the /usr/share/doc/packages/<name of packet> directory.

You can get nearly always help and additional information about a command or configuration file by using the *man <name of command or name of configuration file>* command.

The directories where you will find most of the configuration files for the system are /etc and its subdirectories.

The main configuration file is the /etc/rc.config file. It contains variables that define general settings and which services/daemons to star t Changes to this file have changes of other configuration files as result. To apply these changes you have to execute the /sbin/SuSEconfig executeable.

You can configure the system by editing the configuration files and by using yast and yast2 (Yet Another Setup Tool). In this paper we will edit the configuration files, because this is the best way to get to know the system. The only points where we will use yast are during the initial installation (already completed) and to update packets.

Disabling Services

[5] Domain Name Server

The next step is to disable all running services that provide access from the network (portmap, sshd, httpd). You can get a listing of all open files/ports with the command *lsof –i* (list open files/ports). The output should look like the following:

```
COMMAND PID USER   FD   TYPE DEVICE SIZE NODE NAME
portmap 2008 root    3u  IPv4   2711       UDP *:sunrpc
portmap 2008 root    4u  IPv4   2712       TCP *:sunrpc (LISTEN)
sshd    2022 root    3u  IPv4   2725       TCP *:ssh (LISTEN)
httpd   2159 root   16u  IPv4   2872       TCP *:https (LISTEN)
httpd   2159 root   17u  IPv4   2873       TCP *:http (LISTEN)
httpd   2160 root   16u  IPv4   2872       TCP *:https (LISTEN)
httpd   2160 root   17u  IPv4   2873       TCP *:http (LISTEN)
```

The output shows us in row #1(COMMAND) that portmap, sshd and httpd are running and that we could connect to these services remotely. Row #2(PID) shows the process ID of the program, row #3(USER) lists the user that owns the process, row #4(FD) shows the file descriptor information, telling us in line #1 that the file descriptor number is 3 and the file is opened with read and write access(u). The row #5(TYPE) tells us the type of the node of the file (here are all open files IP v4 sockets. Row #6(DEVICE) shows us the device n umbers. The row #7(SIZE) shows us the size of the file which is empty. The row #8(NODE) tells us the protocol which is used (e.g. TCP). The last row (NAME) tells us the address of the source and destination host, here it is represented by a "*" because we do not have an established connection right now, the name of the service (e.g. sunrpc, ssh, http, https) and the state the daemon is in, e.g. LISTEN (ESTABLISHED) i.e. it is waiting (has established) for a connection. For more details on *lsof* refer to the man page (*man lsof*).

Now we know what services are running and are accessible remotely. How do we find the correspondent scripts to shutdown the services? SuSE provides in its packages scripts for all stand alone services. These can be used by the links f ound in /usr/sbin, the names of them start all with "rc", that refer to the scripts in the /etc/init.d/ directory. These links are placed in the /usr/sbin directory because it is included in the path where to look for executables. If they wouldn't be there you would have to execute the scripts in the /etc/init.d/ directory having to include the whole path to able to execute them (e.g. */etc/init.d/portmap stop*). So we could use these scripts to stop the services or simply by killing the processes, you remem ber the row with the process IDs above. The relation between service name and script to start or to stop the corresponding service is not straightforward considering the *lsof* output. The script for portmap is called rcportmap, for the sshd it is rcsshd, bu t for the httpd it is rcapache because that is the name of the web server. All the scripts take at least two different parameters: stop and start. To stop the portmap service you would use the following command to stop it via the links to the script: *rcportmap stop* and analog to start it: *rcportmap start*. The execution of the script produces an output of the form:

```
"Shutting down <name of daemon> daemon                                   done"
"Starting <name of daemon> daemon                                  done"
```

However as mentioned before you could kill the p rocesses using the *kill* command. To do this you use *lsof* as mentioned above to get the corresponding process ID and execute kill like this: kill <process ID>, e.g. to kill portmap: *kill 2008*. You can check whether it worked out by executing *lsof –i* again, which should result in an output equal the one above except the missing lines referring to portmap. Perform this procedure for each service, either using the scripts or using the *kill* command, and at the end *lsof –i* should show only an empty line, i.e. all services providing remote access are stopped.

Section	Task to do	Status/Value
3.1	Check which services are providing remote access by executing the following command: *lsof –i*	
3.2	Execute the following commands to stop the daemons: *rcapache stop* *rcportmap stop* *rcsshd stop*	
3.3	Recheck that all network services providing remote access to the system have been disabled: *lsof –i* (the output should be empty)	

Updates

Since you don't provide any services to the outside, you can now connect the machine to the network. Updates and their descriptions can be found on the SuSE homepage (http://www.suse.com/us/support/download/updates/index.html) with links to the files on their ftp server (ftp.suse.com/pub/suse/i386/update/7.1). Perform the following step to download, verify and install the packages:

Section	Task to do	Status/Value
3.4	Download the updates via ftp from ftp.suse.com/pub/suse/i386/update/7.1	
3.5	Check the rpm signature: *rpm –v –checksig <name of rpm>*	
3.6 3.7	Compare the fingerprint with the one on the SuSE homepage http://www.suse.de/de/support/security/ Update the packages: with *yast*	
	Choose Packet Management	
	Select Install Packages	
	Change source to directory	
	Enter the directory where the downloaded rpms reside	
	Check the packages	
	Press F10 to install the packages	

There are other possibilities to update packages, but you can't verify the signatures by using these ways. One way is to start *yast*: Choose Packet Management -> Select Install Packages -> Change source to ftp -> Enter: ftp.suse.com/pub/suse/i386/update/7.1 -> Enter the series directory -> Select your packages -> Press F10 to install. The other one is to use *yast2* (in text mode): Choose Online Update -> Manual Update -> Recheck the chosen packages ->Install.

Note: If you install packages from CDROM at a later point of time check out if there is an update for it, otherwise you might use a version that has known vulnerabilities. You should frequently check the SuSE security web pa ge http://www.suse.de/de/support/security/ and/or subscribe to the security mailing list (details will be found on the link provided earlier).

After having completed the update disconnect the server physically from the normal network and connect it to a network only you have access to.

Additional security Programs

After having completed the update process you should consider downloading "Harden SuSE" and "SuSEfirewall2" (iptables based) or "SuSEfire wall" (ipchains based) from Marc Heuse's web page (www.suse.de/~marc/SuSE.html). "Harden SuSE" is a script which allows you to harden your system, by automatically restricting rights on files, checking for world write able files, removing SUID / SGID bits, disabling services and configuring securely openssh. Running the script you will be asked which actions should be performed on your system. After having run the script your system is hardened in a way tha t most services won't work anymore. You have to configure them before you can use them!"SuSEfirewall2" is a set of scripts that allow you to easily configure an iptables firewall on your system, using entries in the /etc/rc.config.d/firewall2.rc.config fil e. Details on the "Harden SuSE" script and the "SuSEfirewall2" will be given later.

Section	Task to do	Status/Value
3.8	Download "Harden SuSE" from www.suse.de/~marc/SuSE.html	
3.9	Download "SuSEfirewall2 (i ptables)" or "SuSEfirewall" (ipchains) from www.suse.de/~marc/SuSE.html	

Since you have all the files you need, you should disconnect the server from the network or put it into a separate LAN to which o nly you have access.

Section	Task to do	Status/Value
3.10	Disconnect your server from the network or connect it to a separate LAN only you have access to.	

4.Configuring the System

Hardening

Before starting to configure the services you would like to of fer, you need to create a more secure basis on which you can build your system and perform some initial install routines. I highly recommend the harden_suse perl script made by Marc Heuse because it changes the system configuration to a more secure state a nd saves you a lot of time. I will discuss what the script does in this section but I won't comment on every line.
The actions performed by the script get logged in the /etc/harden_suse.log file. All files that get altered by the script get backuped with t he extension ".harden_suse". The script automatically generates an undo script (/etc/undo_harden_suse) that will bring your system back to the state it was before.

The harden_suse perl script takes care of 10 areas as described in the documentation provid ed at Marc's homepage http://www.suse.de/~marc/harden_suse.html . The following areas are covered:

1. Deactivation of all network services except very few security services (e.g. SSH, Firewall, VPN)

This is done by manipulating the /etc/rc.config file that contains system settings for services/daemons to start. The script simply sets the "START_" entries for daemons to yes or no to enable/disable services beginning with the next system start. Therefore the script sets the entries to "no" for network services that are not related to security or mandatory (e.g. sshd, fw, scanlogd). This requires you as administrator to explicitly change the entries to get a service/daemon running, i.e. you have to get act ive for services running on your system and you can't forget about services that you haven't configured.

2. Changing the file permissions to a secure state
   This is achieved by setting the PERMISSION_SECURITY="secure local" entry in the /etc/rc.config file. Therefore the permissions on files will be set according to the settings in the /etc/permissions.secure file. The /etc/permissions.secure file contains the rights that have to be associated with directories/files and is delivered by SuSE. In total SuSE offers you different settings for your file permissions: easy, secure and paranoid. The paranoid setting is very strict and makes it hard to even administer the server. The secure setting restricts access to /var/log/messages (the main log file) and many other files to root. These permissions get set each time /sbin/SuSEconfig is executed (it gets automatically executed if you use yast/yast2 to change settings). For a complete list of the permission settings to the files have a look at /etc/permissions, /etc/permissions.secure and /etc/permissions.paranoid

3. Commenting out all services in /etc/inetd.conf and secure the tcp wrapper to allow only localhost access

   The script disables all services offered by inetd by commenting out, i.e. by putting the "#" character in front of a line, the lines in the /etc/inetd.conf file referring to a service (e.g. telnet, ftp, rsh, echo). This forces you as administrator to change the inetd configuration to get such a service running. Therefore you have to decide actively and can't forget that there is a service running you didn't configure.
   It replaces the /etc/hosts.deny file with a file containing the line: "ALL:ALL except localhost". This has as result that access to all services that check the /etc/hosts.deny file to see whether access should be allowed or not (e.g. openssh), is denied to all sources that come from other hosts than the localhost (127.0.0.1). This forces you to change the /etc/hosts.deny file to allow connections from specific hosts to specific services and therefore making an active decision about who should get access to your server.

4. Securing the login process (log all login attempts, show last/failed logins, root login only from console)

   The script changes the login configuration file /etc/login.defs to enable logging of login attempts (FAILLOG_ENAB yes), to display last login information after login (LASTLOG_ENAB yes), to warn if a weak password has been chosen (PASS_ALWAYS_WARN yes) and to disable logins for user that do not have a home directory (DEFAULT_HOME no). In addition the script set limits on coredumps (no coredumps: *ulimit -S -c 0*) in the /etc/profile file. Coredumps are created when a process crashes and are files that contain a copy of the memory at the time the process crashed. Since these coredumps may contain passwords or other security related information, the setting prevents the creation of coredumps for security reasons.

5. secure the passwords (long passwords enforced, password change after 40 days, weak password warning)
   The script makes changes to the /etc/login.defs file to set the minimum length of passwords to 7 characters (PASS_MIN_LEN 7), to set the maximum age of passwords to 40 days (PASS_MAX_DAYS 40), to warn about obscure passwords (OBSCURE_CHECKS_ENAB yes). In addition the s cript enables MD5 passwords. MD5 passwords are used to be able to use passwords with a length of more than 8 characters (MD5 passwords can be up to 128 characters) and therefore providing more security for your system. The bad thing about MD5 passwords is that they are not supported by the standard crypt function used by commercial Unices and some programs. So you only can use this if you do not use software that requires passwords supported by the standard crypt function. However for our setup we can and s hould use MD5 passwords. For more information have a look at /usr/share/doc/packages/pam/README.md5

6. strong permissions on /home directories of users and a strict umask (077) for all users

   The script sets the umask to 077 in the /etc/profile file. This h as the effect that all newly created files will have 700 (read/write/execute for the owner, no rights for the group, no rights for others) access rights. This ensures that only the owner has access to the file. The access rights for the home directories ge t set to 700 by executing the command: *chmod 700 –R /home* to avoid the possibility of access to files in the users home directories for others than the owner of the file.

7. Securing the configuration of SSH/SSHD (disables/enables options for better security)

   The script sets the following entries in the /etc/ssh/sshd.config file:
   PermitRootLogin no
   This disables the possibility to login as user root via ssh. Therefore it is required to login as normal user and then to use sudo or su to perform tasks with ro ot privileges.
   IgnoreRhosts yes
   This setting is set to prevent the usage of rhost files for authentication. This is necessary since this form of authentication is very weak.
   StrictModes yes
   This setting requires the users files and directories to only be w rite able by the owner, otherwise ssh denies access. This is to prevent that access is granted due to the manipulation of configuration files (if the files are world write able everyone could manipulate them).
   X11Forwarding no
   This setting disables the X11 Forwarding that would allow to forward a X11 session to a remote host.
   RhostsAuthentication no
   This setting denies the authentication via rhost files.
   RhostsRSAAuthentication no
   This denies authentication via rhost in combination with RSA authentication.
   PermitEmptyPasswords no
   This prevents that logins are allowed to accounts that do not have a password set.
   LoginGraceTime 300
   This setting sets the timeout for login sessions to 300 seconds.
   PrintMotd yes
   This setting enables the display of the content of t he message of the day file

(/etc/motd). The content gets set to a messages that clearly tells the user that non authorized logins are not allowed.

8. Removing privilege of all unknown suid files on the system
   The script searches for files with SUID bits set with the following command (see *man find* for details):
   find / -type f -perm -04000 -o -perm -02000 ! -fstype nfs
   This tells us that the search start in the root directory (/), the type it searches for if files (-type f), the permissions on the files have to be 04000 or (-o) 02000 and the file system (-fstype) type has to be explicitly not (!) nfs (network file system). The numeric value of 04000 is the suid bit for the group and the 02000 is the suid bit for the others. The suid bit allows other users to e xecute the script/program with the privileges of root. If you should not be familiar with Unix permissions then have a look at Dru Lavigne's introduction to Unix permissions at http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html?page=1 . The script then compares the output from the above mentioned *find* command with the file list found in /etc/permissions. Those files not listed in the /etc/permissions file get their suid bits removed, since they obviously do not belong to the system files.
9. Removing world write permissions on all unknown world write able files on the system
   The script removes the world write permissions of the floppy drive (/dev/fd*), tape drive (/dev/rmt*, /dev/sr*) and audio device (/dev/*audio*). With following *find* command: *find / -perm -2 -type f -or -type d -not -perm -01000 -not -fstype nfs* the script searches for files (-type f) that have the write permission set for others (-perm –2), are not stored in directories that have the sticky bit set (-type d –not –perm –01000) and are not stored on a network file system (-not –fstype nfs). Then it removes the world write permissions of those files by using the command: *chmod o-w*

10. Showing legal d isclaimer in the login banner, motd and lilo boot menu.

    The script changes the /etc/motd, /etc/issue.net, /etc/issue to display the following message when a user logs in:
    Warning: Unauthorized access to this system is forbidden and will be
    prosecuted by law! By accessing this system, you agree that your
    actions can be monitored if unauthorized usage is suspected.

    The issue.net file gets displayed before the login for the telnet session appears. The motd file gets displayed after a successful login but be fore the shell prompt appears. The issue file gets displayed before a login appears.

The table below shows you how to install the "Harden SuSE", "SuSEfirewall", and „SuSEfirewall2" script and to run the "Harden SuSE" script. More details about the "SuSEfirewall2" script will be given later.

Section	Task to do	Status/Value
4.1	Run the "Harden SuSE" script: *tar xvfz harden_suse-<version>.tar.gz* *cd harden_suse-<version>* *./harden_suse*	
4.2	Choose your hardening options. The script will prompt you for what you wan t to secure and how. The most secure and	

	recommended choice is to answer, "yes" to all questions. See above for the descriptions.	
4.3	Install the SuSEfirewall/SuSEfirewall2: *tar xvfz <filename>* *cd SuSEfirewall<version>* *./INSTALL*	
4.4	Reboot the machine to ass ure that all changes take effect.	

Configuring Apache

The packages have already been installed. First of all you have to decide whether you want to use SSL with apache.

There are two possibilities:
1.) You create your own root certificate and sign your si te certificate with it.
2.) You request a signed certificate from a certificate authority (CA).

For both variants you'll find the instructions on the mod_ssl homepage (http://www.modssl.org/docs/2.8/ssl_faq.html#ToC28).

Be aware that if you encrypt your certificate with a password you are required to enter the password each time apache is started! If you fail to do so, apache will not run. Therefore assure that after rebooting the machine or restarting apache you enter the password (you will be prompted for it, but you have very little time to enter it).

The directories related to our web server are the following:

/usr/local/httpd	Root directory of the web server files
/usr/local/httpd/ htdocs	Directory in which the HTML [6] files reside
/usr/local/httpd/cgi-bin	Directory in which the CGI [7] files reside
/usr/local/httpd/icons	Standard directory in which reside some sample icons
/etc/httpd	Directory in which the configuration file and the SS L certificates reside
/var/log/httpd	Directory in which the log files reside

Then you'll need to configure your apache server.
The configuration file resides in the /etc/httpd/ directory and is named httpd.conf. First of all I recommend always making a co py of the configuration file before editing it (this can save you a lot of time if you did changes that did not work as intended to be able to return to a known configuration state). To make a copy use the *cp* (copy) command as following:
cp /etc/httpd/http d.conf /etc/httpd/httpd.conf.orig

The default setup has already some security in mind, but a few changes have to be made. We will discuss here only a few general settings which are important to know about and security related changes we need to make. The apache server has a lot of capabilities and to discuss all features is far beyond the scope of this document. For full documentation have a look at the /usr/share/doc/packages/apache directory, at the apache online documentation

[6] Hyper Text Markup Language
[7] Common Gateway Interface

http://httpd.apache.org/docs/ and especially at the apache security tips http://httpd.apache.org/docs -2.0/misc/security_tips.html .

The next step is to edit the configuration file. You can do this with your favorite editor e.g. *vim*, using the following command: *vim /etc/httpd/httpd.conf*

The commands within *vim* you will need are the following:

/<String>	will take you to the next occurrence of "String" in the doc ument, e.g. /Section 1 will take you to line 41 in httpd.conf where the first occurrence of "Section 1" is
i	will enable insert mode, allowing you to enter text in the document
ESC	will disable the mode you are currently operating in
x	will delete one character at the current position of your cursor
dd	will delete a whole line
|u	will undo the last thing you did, e.g. restoring a deleted character
:w	saves your changes to the file
:q!	quits the program without saving changes
:wq	writes the changes and quits the program
:help	will show you the help information within *vim*

For more commands see the manual page (*man vim*) or the help within *vim*.

Now, since we opened the configuration file we will start to have a look at the global environment settings. Search for "ServerType" by using: */ServerType*, this will take you to line 52. Since the main purpose of this machine is to serve as web server we will start apache as standalone server, i.e. the daemon is invoked at startup and not via inetd. The difference between these two modes is that starting the daemon via inetd will result in the effect that each time a connection is established inetd will handle the incoming request and start apache. This takes significantly more time and therefore can result in client requests not being serviced if there are too many requests and the performance of the server is pretty bad. You should only use the "inetd" mode if there are only few requests being expected. The "standalone" mode will provide the better performance since the daemon and a few instances will permanently reside in memory being always ready to serve requests. So we will leave the ServerType setting to "standalone" as it is already set in the configuration file.

Since you know now how to sear ch for a specific entry in the file using vim, I will not repeat this procedure.

The next variable to look at is the "ServerRoot" that sets the directory under which the server related files such as HTML[8] files. For our setup we leave the value as it is t o "/usr/local/httpd", you remember that you created a separate partition for */usr/local/httpd*.

The number of minimum servers, i.e. this is the number of servers that will always be running, is set in the directive "MinSpareServers". The value is an "it d epends" value. The number of spare servers should be determined by the hardware you have (memory) and the load you expect, i.e. the number of concurrent requests to your server. A value to start with is 3; however if you should notice bad performance and a high load you set this value higher. To change the value move with your arrow keys to the default value which is 1, press "x" to

[8] Hyper Text Markup Language

delete it, then press "i" for insert mode, type "3", that's the actual number we want to insert, press the "ESC" key to return to normal mode and finally save your changes by using the following sequence ":w".

```
MinSpareServers 3
```

The number of maximum spare servers, i.e. this is number of servers that will be allowed to reside in memory waiting for connections, is set in the "Max SpareServers" directive. The value to choose is dependant as is the number of minimum spare severs. A value to start with is 10; however if you should notice bad performance and a high load you set this value higher.

```
MaxSpareServers 10
```

The next value is the number of servers you want to start initially. The corresponding directive is "StartServers". A value to start with is 10; however if you should notice bad performance and a high load you set this value higher. This will result in 10 server processes being available permanently, if you take the number of initially started servers and the "MaxSpareServers" and "MinSpareServers" settings we did above.

```
StartServers 10
```

The number of maximum clients we will serve is set in the "MaxClients" directive. This means that this is the number of maximum servers that will be started. If the number of clients exceeds this value, the remaining clients will not be able to connect. This is to avoid a server crashing due to running out of resources. We should leave the d efault value of 150 unless the numbers of requests require it and the hardware (memory) is sufficient.

```
MaxClients 150
```

Now we are going to discuss some values in section 2, the main server configuration, starting at line 453 of the configuration file. Thi s part is only necessary if you start the apache "standalone".

The "Port" directive sets the port the server should listen on. The default is port 80 and we shouldn't change this unless you have some special setup requirements. Being set to port 80 the server needs to be started initially with root privileges to be able to bind to port 80. This is necessary since all ports below 1023 are so called privileged ports and therefore require special privileges. As you should know it is generally a bad idea to le t services being run as root while being exposed to the Internet, but sometimes you can't avoid this. The apache server handles this by starting the server as root and then decreasing its privileges to a special user we have to define some time later. This procedure is a one -way road, i.e. once having decreased the privileges they can't be increased anymore.

```
Port 80
```

In the case of usage of SSL (Secure Socket Layer) we need an extra port to handle the encrypted connections. This is defined with the "<IfDef ine SSL>" directive. It is already set up and tells us that while SSL being enabled we will use ports 80 and 443. Port 443 is the standard port for SSL (https). We leave this entry untouched.

```
<IfDefine SSL>
```

```
Listen 80
Listen 443
</IfDefine>
```

Now we will ha ndle the setting for the user apache will be run with. This is set with the "User" and "Group" directive. The default settings are "User wwwrun" and "Group nogroup". The user (UID:30) and the group (GID:65534) are already set up. You may use an alternative user and group, but it is not necessary (How to set up a new user is explained in detail later). The group and the user have very few rights and especially no write rights to the pages you want to be accessible from the server. If you would like to find a ll files that are owned by a specific user or group you could use the command *find* as following:

find / -user <username or userid> e.g. *find / -user wwwrun*
find / -group <group name or groupid> e.g. *find / -group nogroup*

The output of this command is a l ist of files that are owned by the user/group. For further details on find refer to the manual page (*man find*).

This protects the files from being altered by exploiting a potential bug in the server, because the user rights associated with the server do n ot allow manipulating the files.

The next directive to manipulate is the "ServerAdmin" directive. The value needs to be an e - mail address of the administrator of the server. It serves as contact address in case a visitor runs into an error or something s imilar. The address given here will be shown on error pages. So change the default value root@localhost to your e-mail address.

```
ServerAdmin admin@mydomain.com
```

The "ServerName" directive sets the name that will be s ent back to clients connecting to the server. This has to be a valid DNS name. The default value is "<hostname>.local"; commonly it is set to "www.mydomain.com", where mydomain.com is the name of your domain (please don't use mydomain.com, replace it with the name of your domain!).

```
ServerName www.mydomain.com
```

The next important setting is done via the directive "DocumentRoot". It is set to the directory where your actual web files will be located. The default is " /usr/local/httpd/htdocs" and you don't need to change this.

```
DocumentRoot "/usr/local/httpd/htdocs"
```

Now we get to the part where you set some restrictive rights for access to the directories accessible via the server. The directive "<Directory />" contain s the rights associated as defaults to all directories. The complete directive looks like this:

```
<Directory />
AuthUserFile /etc/httpd/passwd
AuthGroupFile /etc/httpd/group
Options –FollowSymlinks +Multiviews
AllowOverride none
</Directory>
```

The "<Directory /> " is the beginning and the "</Directory> is the end of the directive. "AuthUserFile" points to a file that contains usernames and passwords. This is only used if you wish to protect specific directories by requiring user authentification. If you want to use this feature read the manual for htpasswd (*man htpasswd*) that is a program to maintain the users and passwords for apache. However using this kind of password protection is not very good since the passwords may get sniffed and could therefore be used by others to gain access to your directory. Since we don't need those for this setup, please delete the "Auth*" lines by pressing twice "d" in the appropriate lines. At the end of the description of the settings in the httpd.conf file is a small section that will show you how to use basic authentication with apache.

Now we will have a closer look on the options you could set within a directive. To begin with here is a short explanation of the options taken from the SANS "Securing Linux Step -By-Step" guide:

ExecCGI	Should only be allowed for CGI directories
FollowSymLinks	If users have write access to the HTML directories, they can set symbolic links to areas that contain sensitive data
Includes	Server side includes can be used to bypass default file access restrictions
IncludeNOEXEC	Safer version of Includes that disable the *#exec* statement and *#include* of CGI scripts
Indexes	The daemon will print a directory listing for any directory without an index file (index.html). This may expose the names of data files ordinarily hidden

The „All" setting is the default that is assumed if the option setting is not set manually. The "All" and some other options are described in the apache documentation at http://httpd.apache.org/docs-2.0/mod/core.html#options as following:

All

All options except for MultiViews. This is the default setting.

MultiViews

Content negotiated MultiViews are allowed.

SymLinksIfOwnerMatch

The server will only follow symbolic links for which the target file or directory is owned by the same user id as the link.

Note: this option gets ignored if set inside a <Location> section.

In addition the "None" setting for options makes sure that no option is set, which is the safest since is does not provide any additional functionality that could be exploited.

Our directive now looks like this:

```
<Directory />
Options –FollowSymlinks +Multiviews
AllowOverride none
</Directory>
```

The "Options –FollowSymlinks" denies following symbolic links to other directories that might be somewhere else in our file system. This protects you from disclosing information from other directories.
The "+Multiviews" parts tells the server to look for files that look like the one requested if the requested file does not exist and display this instead. This is not a very good idea since this might reveal information you don't want to be revealed. Therefore delete the "+Multiviews" part.
The "AllowOverride none" part tells the server that the settings set here in the directive can not be overridden by the usage of ".htaccess" files where you could set rules for the specific directory this file resides in. This is good since this grants that all directives for the directory are set here for this directory and nowhere else.

Your directive now looks like this:
```
<Directory />
Options –FollowSymlinks
AllowOverride none
</Directory>
```

To be more restrictive we set the "Options" to "None" as recommended by the SANS "Securing Linux Step-By-Step" guide and therefore disabling all options to be on the secure side. Your directive now looks like this:

```
<Directory />
Options None
AllowOverride none
</Directory>
```

Since we want to be restrictive we will deny access for everyone unless especially specified for each directory. To accomplish this we have to include the "order" "deny" and "allow" settings.
The order setting allows us to set the sequence in which access is granted. By using "order deny, allow" the deny rule will be examined first and then the allow part. Now we add this line and the deny line which should be "deny from all". This denies access to the directory to all. Now this requires you to explicitly set an allow rule for each directory you want others to have access to. Our directive looks like this now:

```
<Directory />
Options None
AllowOverride none
order deny, allow
deny from all
</Directory>
```

The next part we have to look at is the configuration for your "DocumentRoot" directory. These settings are in the "<Directory "/usr/local/httpd/htdocs">" directive beginning at line 562.

The “Options” line looks like the one you saw in the “<Directory/>” setup:

```
Options Indexes –FollowSymlinks +Multiviews
```

Since we want to avoid showing files that have a similar name as the one requested we simply delete the “+MultiViews” part and to avoid the listing of the directory if the requested file is not found we delete the “Indexes” entry and to be very restrictive we set the “Options” to “None”, resulting in the following “Options” line:

```
Options None
```

The following directive:

```
AllowOverride none
```

can stay as it is since it reflects the setting we did earlier.

The settings in lines 583 and 584:

```
order allow, deny
Allow from all
```

Can stay as they are as long as you do not want to restrict access to the files in the directory to specific IP - addresses. If want to restrict access to certain IP - addresses you would have to change the setting to e.g.:

```
order deny, allow
deny from all
allow from 10.
```

To allow access to the directory solely for clients connecting from an IP- address within this specific IP - range.

Remember, these settings can be set for each directory and can therefore give some security if necessary. Checks that rely upon IP - addresses could be circumvented by spoofing (faking) the source IP - address, so it’s only a weak protection, but better than none.

The next part in the configuration file is related to the web_dav module, which we did not install. However, if this module should ever be installed it provides unauthorized read and write access to the web directories (it is used for web based remote distributed authoring and versioning)! Therefore it is explicitly disabled by the following lines:

```
<IfModule mod_dav.c>
DAV off
</IfModule>
```

The following lines in this directive have to be disabled since they are only there for the test page. To disable entries in the httpd.conf simply place a “#” at the beginning of the line and it will be ignored by apache.

```
<Files /usr/local/httpd/htdocs/index.htm*>
	Options -FollowSymLinks +Includes +MultiViews
</Files>
<Files test.php3>
	Order deny, allow
	deny from all
	allow from localhost
</Files>
```

When disabled they will look like this:

```
#<Files /usr/local/httpd/htdocs/index.htm*>
#	Options -FollowSymLinks +Includes +MultiViews
#</Files>
#<Files test.php3>
#	Order deny, allow
#	deny from all
#	allow from localhost
#</Files>
```

The module mod_userdir defines the directory that should be accessed if a request like the following is made: http://host/~username where username is the n ame of a real user on your system. Since we do not provide such functionality and we don't allow to override security settings by allowing to define options in the .htaccess files, disable them by placing "#" in front of the following lines:

```
<IfModule mod_userdir.c>
   UserDir public_html
</IfModule>
AccesFileName .htaccess
```

It should look like this:

```
#<IfModule mod_userdir.c>
#   UserDir public_html
#</IfModule>
#AccesFileName .htaccess
```

In line 728 there is defined that hostname lookups won't be perform ed. We leave that as it is since this reduces traffic. Therefore you will have only IP - addresses in your log files. If you would like to have resolved domain names in your log files then set the following entry to "yes", but this results in additional tra ffic since the IP-address gets resolved when a client access the server and therefore a lookup of the IP -address is performed. The line is as following:

```
HostnameLookups no
```

The destination and level of error messages if as following defined in lines 788:

```
ErrorLog /var/log/httpd/error_log
```

```
LogLevel warn
```

This tells you that in case of malfunctions you should have a look in the /var/log/httpd/error_log file to be able to get more information about possible errors that occurred. The "LogLevel" is set to "warn " to only log warning messages such as that a child process did not exit properly. You could set this level to "debug" to get all messages but that would result in a very large log file and should only be done if you really have problems you can't solve otherwise. For further details have a look at the apache documentation.

In line 813 is defined where the custom log file will be stored. This is the log file where you can find entries about what was accessed by whom at what time. The following line:

```
CustomLog /var/log/httpd/access_log common
```

tells us that the log file named access_log is located in the /var/log/httpd/ directory and will be written in the common format. There are possibilities to change the format of the log file. If you would like to do so, please refer to the apache manual.

The "ServerSignature" option in line 835 allows you to choose whether you want to have additionally the version of your server and the e -mail address of the hostmaster added to error pages. Since we don't need it we can switch it off. But be aware that disabling this option doesn't hide your sever version completely from a visitor. You can get the version of a web server by doing the following steps:

```
telnet www.domain.com 80
head http 1.1
and press enter twice
```

this will result in an output like the following:

```
HTTP/1.1 400 Bad Request
Date: Wed, 10 Sep 2001 10:38:18 GMT
Server: Apache/1.3.19 (Unix) (SuSE/Linux) mod_ssl/2.8.1 OpenSSL/0.9.6
Connection: close
```

If you want your visitors to be able to see the additional information then enable the option otherwise leave it as it is:

```
ServerSignature off
```

There should exist a way to eliminate the information given back as shown above by manipulating the source code. However this would only result in security by obscurity and would not prevent a cracker[9] from trying to break your server anyway. Therefore we do not consider this option.

We have still some settings to improve for the remaining directories related to our web server.

The /usr/local/httpd/icons directory stores the sample icons that are installed by default. The default configuration starting at line 849 is as following:

[9] A cracker is a person that intrudes and destroys computer systems with the intention to perform vandalism and/or steal information.

```
Alias /icons/ "/usr/local/httpd/icons/"

<Directory "/usr/local/httpd/icons">
Options Indexes MultiViews
AllowOverride none
Order allow,deny
Allow from all
</Directory>
```

As we already learned early we should restrict access to the directory in that way that a directory listing cannot be obtained. Therefore it is necessary to make changes to "Options" setting as following:

```
Alias /icons/ "/usr/local/httpd/icons/"

<Directory "/usr/local/httpd/icons">
Options None
AllowOverride none
Order allow,deny
Allow from all
</Directory>
```

The /usr/local/httpd/cgi-bin directory is a standard directory for common gateway interface files that are executables, normally perl scripts. These are used to generate or process information. If you want to use such scripts be aware that programming errors or flaws of other kinds could easily lead to a compromised system since they are executed on the web server! So either you disallow the usage of those scripts or make sure that they cannot be used to compromise your system. For our setup we won't allow the usage of such scripts due to the above-mentioned risks.
Therefore we have to make the following changes to the configuration file:

Search for the first occurrence of cgi-bin within the configuration file. This should be in line 866 and looks like the following:

```
ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
<IfModule mod_perl.c>
…
</IfModule>
<Directory "/usr/local/httpd/cgi-bin">
…
</Directory>
```

We need to disable these lines to avoid having an alias for the directory and to disable the associated rights with the alias. You do this by inserting the "#" sign in front of these lines. An alias makes it possible to access a certain directory by the usage of another name, e.g. like above the directory /usr/local/httpd/cgi-bin gets accessible via /cgi-bin/. The "<IfModule mod_perl.c>" directive gets executed if the mod_perl.c module is installed.
The lines should look this:

```
#ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
```

```
#<IfModule mod_perl.c>
#…
#</IfModule>
#<Directory "/usr/local/http/cgi-bin">
#…
#</Directory>
```

Then we search for the next occurrence of cgi -bin which is in line 894. There we can find the general setting for the /cgi-bin location that we have to disable to not allow the execution of scripts within this location. From the Apache online documentation (http://httpd.apache.org/docs/mod/core.html#location) we get to know that "The location directive provides access control by URL.". This tells us that this directive is only relevant when the according URL /cgi-bin is used to access the directory. We therefore disable the appropriate lines:

```
<location /cgi-bin>
…
</location>
```

to

```
#<location /cgi-bin>
#…
#</location>
```

With this last step for the apache configuration file, we have finished configuring apache. After the summary list you'll find some additional information on how to configure basic authentication.

In the following list you'll find a summary of the tasks that have been described above.

Section	Task to do	Status/Value
4.5	Check out the information on the mod_ssl homepage (http://www.modssl.org/docs/2.8/ssl_faq.html#ToC28) on how to create or obtain a signed SSL certificate	
4.6	Edit /etc/httpd/httpd.conf	
	Change the "SSLEngine of" entry to "SSLEngine on" to enable SSL	
	Change the "SSLCertificateFile" entry according to where your server certificate resides.	
	Change the "SSLCertificateKeyFile" entry according to where your private server certificate resides.	
	Depending upon your configuration you might need to change the "SSLCA*" entries	
	Depending upon your configuration you might need to change the "SSLCertificateChainFile" entry	
	Change the "MinSpareServers" from 1 to 3 Change the "MaxSpareServers" from 1 to 10	
	Change the "StartServers" from 1 to 10	
	Change the "ServerAdmin" setting	
	Change the "ServerName" setting	

	Change the <Directory /> directive to the following: <Directory /> Options None AllowOverride none Order deny, allow Deny from all </Directory>	
	Change the <Directory "/usr/local/httpd/h tdocs"> to the following to restrict access to the default directory for web pages: <Directory "/usr/local/httpd/htdocs"> Options None AllowOverride none Order deny, allow Allow from all </Directory>	
	Disable the following entries that are related to the default test web page by putting the "#" in front of each line. The lines should look like the following: #<Files /usr/local/httpd/htdocs/index.htm*> #Options –FollowSymLinks +Includes +Multiviews #</Files> #<Files test.php3> #Order deny, allow #deny from all #allow from localhost #</Files>	
	To avoid exposing user directories and to avoid putting individual access rights for directories within .htaccess files you have to comment out the following lines by putting the "#" in front of each line as shown below: #<IfModule mod_userdir.c> #UserDir public_html #</IfModule> #AccessFileName .htaccess	
	To avoid exposing the content of the icons directory change the <Directory "usr/local/httpd/icons/"> directive as following: Alias /icons/ "/usr/local/httpd/icons /" <Directory "/usr/local/httpd/icons"> Options none AllowOverride none Order allow, deny Allow from all </Directory>	
	To disallow the usage of CGI scripts you have disable the CGI related entries in the configuration file according to the following: #ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi -bin/" #<IfModule mod_perl.c> #… #</IfModule> #<Directory "/usr/local/httpd/cgi -bin"> #…	

	#</Directory>	
	…	
	#<location /cgi-bin>	
	#…	
	#</location>	

In addition you might have a look at the diff (diff –u0) output from the standard configuration file and the new one.

As stated earlier you will find below additional information concerning the basic authentication that can be used with apache.

Authentication Method

The following is a short description about how you could use simple password protection for a directory: The sample used here can be applied to all directory directives in the /etc/httpd/httpd.conf file.

This sample shows us the AuthUserFile entry that points to the file where you store the users and according passwords to control access to the directory. The AuthGroupFile entry points the file that contains the groups of users you want to grant access to the directory. Therefore you have the possibility to grant acce ss for whole groups or users (I stripped the other things away since we are only talking about the password protection now):

```
<Directory />
AuthUserFile /etc/httpd/passwd
AuthGroupFile /etc/httpd/group
….
</Directory>
```

First we need to add some information we want to provide to the client to be able to determine the area/directory that is about to be accessed. This is done with the "AuthName" setting. Adding an area name "top secret" that will be shown to the client would look like this:

```
<Directory />
AuthUserFile /etc/httpd/passwd
AuthGroupFile /etc/httpd/group
AuthName "top secret"
….
</Directory>
```

Now we need to set who is allowed to access. This can be done using one of the following settings. "Require group" will grant access to authenticated users tha t belong to the group, "require user" will only allow specific authenticated users. Now we have to create users and passwords. For this task you have to switch to another console by pressing ALT+F2 you will get to the second console (You can always switch back by pressing ALT+F1). Login and execute *htpasswd* that resides in the /usr/bin directory (to find the directory in which executables reside that can be executed without the need to provide the whole path can be determined by *which <name of executable>* the output is simply the full path to the executable) with the following parameters:

htpasswd –c /etc/httpd/passwd <username>

You will get prompted to provide a password. If you want to add more than one user leave the "-c" parameter away since this is us ed to create the file and would result in a new file with only the last user in it! Repeat until you have all your users set up.

If you want to use the group file then you to edit the /etc/httpd/group file and add the groups and users in the following for mat:

Group name: user1 user2 user3

e.g.

friends: john jack

Now you have your users and groups set up. Lets go back to our directive by switching to the first console (ALT+F1):

```
<Directory />
AuthUserFile /etc/httpd/passwd
AuthGroupFile /etc/httpd/group
AuthName " top secret"
Require user john jack
….
</Directory>
```

This tells us that only the users john and jack are allowed to get access after having provided the correct password. It's equivalent with the group. This can be done for each directory. For further details refer to the apache documentation. But remember the usernames and passwords can be sniffed and misused unless they are only provided over an encrypted connection!

Web server protection on file system level

For the updates of the web pages you need to create another user, e.g. "wwwadmin". This user will own all documents. It is important, that this user is different from that user with whose rights the apache web server is running, because if it should be possible to trick the server to exec ute programs these can't change the sites documents. The following steps and the idea have been derived from Marc Heuse's "Installation of a Secure Web Server" guide.

The first step is to create the user wwwadmin that will serve as owner of all files bein g displayed by the web server. As mentioned above this prevents the possibility of manipulation of web pages by exploiting a bug in the web server since the associated user rights with the user the server is running with do not allow the alteration of the web pages. To create the user you should use the command *useradd* (see *man useradd* for details) as following:

useradd –m wwwadmin

The –m option is used to automatically create a home directory for the user.

Provide a password for the user:

passwd wwwadmin

The next step is to change the ownership for the web documents to the wwwadmin user we created and to protect the files from other users. For this we use the chown and the chmod command. The chown command changes the owner of files/directories. The –R option applies recursively, i.e. including all subdirectories, the ownership of all files and directories including and below the given directory to the user given in the command line. The –h option tells us that the action is also performed on symbolic links that are included in the directory. The chmod command changes the modus/rights of directories/files. The –R option has the same effect as the one described with the chown command. The go -w parameter tells us that the write right (w) will be taken away (-) from the group owner (g) and the others (o). The parameter a+r tells us that all will be granted the right to read in the given directory. This assures that only the wwwadmin user can write/modify/delete and others can read the content of the directories.
Perform the following steps:

chown –R –h wwwadmin /usr/local/httpd/htdocs
chown –R –h wwwadmin /usr/local/httpd/cgi -bin
chown –R –h wwwadmin /usr/local/httpd/icons
chmod –R go-w /usr/local/httpd/htdocs
chmod –R go-w /usr/local/httpd/cgi -bin
chmod –R go-w /usr/local/httpd/icons
chmod –R a+r /usr/local/httpd/htdocs

The above shown commands could be easily performed regularly to assure that these rights stay as they are by writing a shell script that gets automatically executed via cron every e.g. hour. To achieve this you have to create a script and put it in the /etc/cron.hourly directory. All scripts within this directory that are owned by root get executed once each hour. The right to write to the directory is reserved to the root user. Therefore no other user is allowed to put scripts into this directory.
The script would look like the following:

```
#!/bin/sh
chown –R –h wwwadmin /usr/local/httpd/htdocs
chown –R –h wwwadmin /usr/local/httpd/cgi -bin
chown –R –h wwwadmin /usr/local/httpd/icons
chmod –R go-w /usr/local/httpd/htdocs
chmod –R go-w /usr/local/httpd/cgi-bin
chmod –R go-w /usr/local/httpd/icons
chmod –R a+r /usr/local/httpd/htdocs
```

To assure that the file can only be accessed/altered by root you have to restrict the rights associated with the file. To do this perform the following task:

chmod 700 /etc/cron.hourly/wwwadmin

This sets the read, write and execute flag for the owner of the file, in this case root and takes all rights for other users and the group away. The modus used here is numeric and the same could be done with the symbolic parameter: *chmod u+rwx,go -rwx /etc/cron.hourly/wwwadmin*

Section	Task to do	Status/Value
4.7	Create a user for the administration of web documents: *useradd –m wwwadmin*	
4.8	Provide a password for the user: *passwd wwwadmin*	
4.9	Change the owner for the web documents: *chown –R –h wwwadmin /usr/local/httpd/htdocs* *chown –R –h wwwadmin /usr/local/httpd/cgi -bin* *chown –R –h wwwadmin /usr/local/httpd/icons* Modify the access rights for web documents(take write permission away from group and others and give everybody read access for web documents): *chmod –R go-w,a+r /usr/local/httpd/htdocs* *chmod –R go-w /usr/local/httpd/cgi-bin* *chmod –R go-w /usr/local/httpd/icons*	
4.10	To make sure that these permissions stay as they are, even though changes to the permissions might occur, you may add these commands in a shell script and place it in the /etc/cron.hourly directory, it will then be executed every hour. Protect the file by assigning all rights only to the user root: *chmod 700 /etc/cron.hourly/wwwadmin*	

Configuring Openssh

Openssh is a tool that provides an encrypted connection for remote access. It can be used as a replacement for telnet and ftp. This offers the possibility to update the web documents over a secure (encrypted) c onnection in contrast to the normal ftp transfer during which the username and password get send in clear text, i.e. every computer between the sender and the server gets the username and password and can therefore misuse it. To use this program you need to setup a few things. You should have a look at the manual page (*man ssh*) and the documentation located in the /usr/share/doc/packages/openssh directory.
This configuration uses protocol version 2, since version 1 is obsolete and is held to be insecure. Th e version 1 is still supported due to historic reasons but won't be used here.

In the lines below we will discuss the options to set in the ssh server configuration file /etc/ssh/sshd_config and the steps you need to make to get it running.

First of all you have to create keys for the user who will be updating the web pages. This user is as described before the user wwwadmin. To create the keys you have to perform the following steps:

Login as user wwwadmin

Create the keys and provide a password for the keys:

ssh-keygen –d

The keys get automatically stored in the /home/wwwadmin/.ssh/ directory and are named id_dsa.pub and id_dsa.

Copy your public key, it has the ".pub" ending, to the authorized_keys2 file. This is necessary because ssh will look for the key in this file and if the public key is not present it won't allow you to connect.

cp /home/wwwadmin/.ssh/id_dsa.pub /home/wwwadmin/.ssh/authorized_keys2

Check the permissions on the key files:

ls –l /home/wwwadmin/.ssh/

The output should look like the following (only filename and the permissions are important):

```
-rw-r--r--    1    wwwadmin    users    ……..        authorized_keys2
-rw-r--r--    1    wwwadmin    users    ……..        id_dsa.pub
-rw-------    1    wwwadmin    users    ……..        id_dsa
```

If the permissions should be different then apply the correct permissions as displayed above by using the *chmod* command as described earlier.

To create the server keys perform the following steps:

Login as user root

ssh-keygen –d

Choose the /etc/ssh/ directory for storage.

Check the permissions on the keys as described above.

Now we have to make changes to the sshd configuration file /etc/ssh/sshd_config to point to the server key files and enable the sftp -server to be able to use ftp over the encrypted connection provided by openssh.

Open the /etc/ssh/sshd_config with your editor (e.g. vim) and change/look for the following entries:

vim /etc/ssh/sshd_config

```
Subsystem    sftp    /usr/lib/ssh/sftp -server
```

This setting allows you to use ftp over the encrypted ssh connection and therefore having a secured ftp transfer.

```
HostKey /etc/ssh/id_dsa
```

This entry defines the location of the private key.

```
Port 22
```

This defines the port the ssh server will listen on for connections. Therefore you have to connect to this port from your client to be able to establish a connection.

```
Protocol 2
```

Here we restrict the usage to the ssh protocol 2 that is far more secure than the protocol version 1 and in addition the usage of the sftp server is only possible within a protocol 2 connection.

```
LoginGraceTime 300
```

This setting sets the timeout for connections to 300 seconds to avoid having server processes waiting for client side authentication too long. Waiting for authentication binds resources and should therefore be limited.

```
PermitRootLogin no
```

This setting disa llows direct root logins via ssh. This is a security feature since to gain root access to the system via ssh is only possible by first connecting as normal user that has a ssh account on the system and having his key being stored in the authorized_keys2 fi le. Then the user needs the password for his private key and when having successfully connected, the user needs the privilege to obtain root privileges e.g. by using sudo. On the one hand this makes it more difficult for the administrator to administer the server, but on the other hand it makes far more difficult for an attacker to gain root access.

```
IgnoreRhosts yes
```

This tells the server to disregard entries in the rhost and hosts.equiv files that are normally used by the r-commands that provide remote ad ministration capabilities without further authentication (see man rhosts). The usage of the r -commands is not recommended since the authentication is solely based on the name/IP -address of the connecting host, without the need to provide a password.

```
StrictModes yes
```

This makes the server check whether the files in the .ssh directory are only write able by the owner of the files and not by others. If they are write able by others the server denies access.

```
X11Forwarding no
```

This setting is only done for com pleteness since we didn't install any X servers. However this setting denies the forwarding of X11 sessions to other hosts.

```
PrintMotd yes
```

This setting enables the displaying of the content of the /etc/motd file during login that tells the user that unaut horized access is forbidden.

```
KeepAlive no
```

This setting has two sides. One side is that keep alive information will be send between client and server to be able to determine if the connection has been interrupted and the server can close the connection. T his results in not having server resources bound to connection that are already dead. On the other side these keep alive packets are not sent within the encrypted connection and are therefore spoofable. The usage of ClientAliveInterval is therefore recommended (see below).

ClientAliveInterval 30

This set the interval in which the server will request a packet from the client over the encrypted connection to be able to determine whether the client is still connected. This is the more secure variant to see whether the client is still alive (compare with KeepAlive).

ClientAliveCountMax 3

This sets the maximum number of intervals the server will keep the connection while not receiving response from the client. In combination with the ClientAliveInterval the server will keep the connection open for a maximum of 90 seconds if the client does not respond.

LogLevel INFO

Here the level of logging information is set to INFO. Information concerning incoming connections will be logged (connection from/to, name of user, success/failure). It is not recommended to set the LogLevel to DEBUG since this might violate the privacy of the users.

PasswordAuthentication no

This disables authentication via tunneled unencrypted passwords. Since this form of authentication is not very secure, it is disabled.

PermitEmptyPasswords no

This disables the possibility to login as user that has no passwords. This would obviously be a very insecure setting if enabled.

PubkeyAuthentication yes

This enables authentication using public/private keys. This is enabled by default. This is the most secure form of authentication since it requires:
Client: private key, password for private key
Server: public key stored in authorized_keys2

Therefore the method requires possession (private key), knowledge (private key password) and authorization (public key must be authorized by server) to establish a successful connection.

Now you still cannot use ssh to connect to your server because ssh is compiled against the libwrap library and therefore looks into the /etc/hosts.allow and /etc/hosts.deny files to determine whether the source IP-address (client) is allowed to connect to the server. How to set this up will be discussed in the next section.

Section	Task to do	Status/Value
4.11 4.12	The first step is to generate keys for the user account (wwwadmin) that needs to have either shell or ftp remote access: For ssh2(DSA-key): *ssh-keygen –d* Provide a strong password for the key. Assure that the private key is not world readable! The public key should be world readable. Copy your public key to authorized_keys2: *cp /home/<userdir>/.ssh/id_dsa.pub /home/<userdir>/.ssh/authorized_keys2* The user root has to create a host key for the ssh server: For ssh2(DSA-key): *ssh-keygen –d* Use an empty password for the host key! Store the host key in the /etc/ssh/ directory. Assure that the private key is not world readable! The public key should be world readable.	
4.13	Configure the sshd by editing the /etc/ssh/sshd_config file	
	Set/check the following settings: PubkeyAuthentication yes PermitEmptyPasswords no PasswordAuthentication no LogLevel INFO ClientAliveCountMax 3 ClientAliveInterval 30 KeepAlive no PrintMotd yes X11Forwarding no StrictModes yes IgnoreRhosts yes PermitRootLogin no LoginGraceTime 300 Protocol 2 Port 22 HostKey /etc/ssh/id_dsa Subsystem sftp /usr/lib/ssh/sftp-server	
	Refer to the example sshd_config file in the Appendix	
4.14	Start openssh: *rcsshd start*	
4.15	Make a copy of your configuration file: *cp /etc/sshd/sshd_conf sshd_conf.<date>.save*	

If you want to use sftp from a MS Windows client you should consider trying MindTerm[10], since it is the only client I know of that provides a bridge for ftp connections via openssh. www.freessh.org is a site where you can find secure shell clients and server for many platforms.

Tcp wrapper

A tcp wrapper is normally used with daemons started via inetd. It provides security before a connection is made to the destination service. The handling is as following: a client wants to

[10] AppGate Inc., www.appgate.org/products/mindterm/

connect to a daemon, let's say a ftpd residing on port 21. The inetd handles all incoming connection requests and looks up in his configuration file /etc/inetd.conf which daemon to start. As you can see in t he sample inetd.conf in the appendix, the first statement is the service name e.g. ftp, second the socket type, third the protocol used, followed by the flags, user, server path and finally the executable. The last statement is where the tcp wrapper drops in. The tcp wrapper is invoked by inetd, the wrapper looks up in /etc/hosts.allow and /etc/hosts.deny to see whether the source IP address is allowed to connect to the service. If there is a rule that denies a ccess the daemon simply does not get invoked and if it is permitted the daemon gets started.

Openssh is compiled against the libwrap library and therefore uses the /etc/hosts.allow and /etc/hosts.deny rules. The first thing to set is a "deny all" rule (se e /etc/hosts.deny in the appendix). Now all connections which are controlled via /etc/hosts.deny and /etc/hosts.allow are protected in such a way that no connections are allowed. Now we can specify some exceptions in the /etc/hosts.allow file for those addresses which should be allowed to connect to sshd (see /etc/hosts.allow in the appendix).

Section	Task to do	Status/Value
4.16	Edit /etc/hosts.deny and set a "all deny" rule: ALL:ALL	
4.17	Edit /etc/hosts.allow and set a rule for openssh to allow remote administration from specific IP addressees e.g.: SSHD: 10.0.1.1 192.168.0.6	

The same procedure can be done for other services you might want to offer via inetd or which use the libwrap library.

Global configuration

The file /etc/rc.config is the global configuration file that is used to set the system configuration (Network interfaces, daemons to start etc.). Some services that require further configuration (SuSEfirewall, SuSEfi rewall2) have their own configuration file that is located in the /etc/rc.config.d directory, but whether they are started or not is set in the /etc/rc.config file. Edit the /etc/rc.config file and perform the following changes.

```
START_HTTPD=yes
```

This configures the system to automatically start the apache web server after booting.

```
START_SCANLOGD=yes
```

Scanlogd is a daemon that will warn you if your sever gets scanned via a port scanner (e.g. nmap). This is a good tool to detect sources that might prepare a n attack on your server.

```
START_SSHD=yes
```

This configures the system to automatically start the ssh server that allows you to administer your system remotely over a secure (encrypted) connection.

```
START_FW2=yes or START_FW=yes
```

This configures the system t o automatically start your local firewall, either SuSEfirewall (ipchains, for kernel 2.2.x) or SuSefirewall2 (iptables, for kernel 2.4.x) after booting.

START_IPPL=yes

This configures your system to automatically start the ippl (IP protocol logger) that logs all by default all tcp and icmp connections. This makes it easier for you to control what connections have been performed from and to your server.

START_ARKEIA=yes

Since we are going to configure arkeia the backup program we can set the entry to yes to enable the automatic starting of it at boot time.

The following remaining servers/services can be disabled by changing the entries to the following:

START_NSCD="no"

This is the name service cache daemon (*man nscd*) that can be used to cache the IP -addresses and host names as well as user information. Since we don't need it, we can disable it.

START_KERNELD="no"

The kernel logging daemon is a daemon that intercepts the kernel messages and logs them. We do not need it since syslogd does the logging fo r us.

START_GPM="no"

The gpm is a daemon that enables the usage of a mouse on the console. We don't need it therefore we disable it.

START_PORTMAP="no"

Portmap provides a service comparable to inetd but for remote procedure call (RPC) server. The servers do not have a static port they listen on; therefore they register with the portmap daemon. The clients ask the portmapper on which port a specific server is listening on to know where to connect. Since we don't use such server we can disable it.

START_INETD="no"

Inetd is a daemon that can handle requests for many daemons. It invokes the daemon when a connection is attempted on the port the daemon is registered for. Since all our daemons work in standalone mode we don't need it and turn it off.

Execute /sbin/SuSEconfig to apply the changes you have made.

Section	Task to do	Status/Value
4.18	Edit the /etc/rc.config file to enable / disable the following services:	
4.19	Set the variable "START_HTTPD" to yes	
4.20	Set the variable "START_SCANLOGD" to yes	
4.21	Set the variable "START_SSHD" to yes	
4.22	Set the variable "START_FW2" (iptables) to yes or the variable "START_FW" (ipchains) to yes	
4.23	Set the variable "START_IPPL" to yes	
4.24	Set the variable "START_ARKEIA" to yes	
4.25	Set all other "START_*" variables to no	
4.26	Execute /sbin/SuSEconfig for updating all scripts and settings that are related to the system configuration	

The /etc/inittab describes the scripts that are started at boot up and normal operations. There is defined e.g. that you are prompted for a passwo rd when entering runlevel 1. But you can still improve security by performing the following step.

Section	Task to do	Status/Value
4.27	To avoid the possibility of rebooting via the Ctrl+Alt+Del key combination you have to edit the /etc/inittab file and put a n"#" in front of the line, which will deactivate it: *ca::ctrlaltdel:/sbin/shutdown -r -t 4 now*	

SuSEfirewall2 configuration

Even though you have a firewall in front of your web server you should consider configuring a local firewall since it can give yo u additional protection e.g. in the case your normal firewall is compromised or has a flaw you haven't fixed. SuSEfirewall2 is based on iptables and offers an easy to configure firewall.

A packet filter is software that looks at the header of a packet an d then decides whether the packets gets dropped (thrown away), gets passed on to its destination (accepted), or gets treated in another way. Therefore we can use it to protect our server in that way that it only accepts those packets we want it to accept. Another thing it is good for is to inform us about packets that were denied, so we can detect attacks or simply misconfigured computers or routers. To tell the kernel what to do with the packets we have to define rules. This is done with the user space pro gram *iptables* (see *man iptables*). If a packet arrives from the Internet it gets to our INPUT chain. This chain should contain rules that decide what to do with the packet (drop it, accept it, forward it to another computer). If the packet is not for us but for another computer that is connected to another network card in our server we give it to the FORWARD chain. If we want to send a packet to the Internet we put it in the OUTPUT chain.
So we have three chains that handle our packets: INPUT, FORWARD, OUTPU T.
So what possibilities does us iptables give to do with our chains:

iptables –N	create a new chain
iptables –X	delete an empty chain
iptables –P	change the policy for a chain
iptables –L	list the rules we have set in our chain
iptables –F	flush our rules
iptables –Z	reset our packet and byte counter for the chain

iptables –A	add a new rule
iptables –I	insert a new rule
iptables –R	replace a rule in our chain
iptables –D	delete a specific rule

Now to create a rule we have to imagine what we need: A sourc e (-s sourceaddress, -sport source port) where the packet comes from, what the packet looks like (-p protocol), a destination (-d, -dport destination port) and what we do with the packet (DROP, ACCEPT). But what do we do when we request a packet and we wa nt to get the answer? There comes the new feature of state tracking (statefull). To determine whether a packet that is coming in is an answer to a packet I sent, we have to keep track of what we did before (-m --state).
Now we know something about iptables and some options we can use to generate rules. There are far more than those. The rules we create are only temporary. If we reboot the computer after having set the rules we loose them all. Therefore it is necessary to save the rules to a shell script tha t gets executed every time the system starts up. To show how to build rules and to show that this takes quite some time are some rules below. These rules are not sophisticated and do not use all the possibilities iptables offers.

First we have to set our policy on the chains and to be on the secure side we drop all packets by default. That means if our rules do not match the packet it gets thrown away: this is to prevent that we miss to write a rule and therefore grant access to our server.

```
iptables –P INPUT DROP
iptables –P OUTPUT DROP
iptables –P FORWARD DROP
```

Now what do we want to let in:

We want our web server to be able to be reached on port 80 (http) and port 443 (https).

```
iptables –A INPUT –p tcp –d 10.0.0.1 –dport 80 ACCEPT
iptables –A INPUT –p tcp –d 10.0.0.1 –dport 443 ACCEPT
```

We want to be able to receive packets that are answers to packets we sent.

```
iptables –A INPUT –d 10.0.0.1 –m –state ESTABLISHED ACCEPT
```

What do we want to let out:

We want our web server to respond to requests that have bee n made and are therefore sent from our web server from port 80 or 443.

```
iptables –A OUTPUT –p tcp –sport 80 ACCEPT
iptables –A OUTPUT –p tcp –sport 443 ACCEPT
```

This is only a small example of rules that we would have to write to protect our server and define what packets should get in and out. To list all rules would take a lot of space and a lot of time.

This time can be saved by using the SuSEfirewall2 written by Marc Heuse. This firewall is a set of scripts that allow us to define the services/ports we w ant to let in/out very quickly.

The configuration file for the SuSEfirewall2 is /etc/rc.config.d/firewall2.rc.config.

The first thing we have to do is to define our network device pointing to the area we want to be protected from. In our case it is the n ame of our only network device (eth0):

```
FW_DEV_EXT=”eth0”
```

If we would have a second network device pointing to a secure or trusted network we would make the following entry, for example:

```
FW_DEV_INT=”eth1”
```

But as we don’t have one we leave it empty:

```
FW_DEV_INT=””
```

We don’t have a demilitarized zone (DMZ) since this is a local firewall. Therefore we leave the following entry empty:

```
FW_DEV_DMZ=””
```

Our web server doesn’t provide routing and therefore we have to disable routing:

```
FW_ROUTE=”no”
```

We do not ma squerade other hosts, i.e. we would hide the IP -addresses of other host behind the address of the web server. Therefore we disable masquerading:

```
FW_MASQUERADE=”no”
```

Now we define which services we want to provide to the outside (http, https):

```
FW_SERVICES_EXT=”www 443”
```

Now we have to define the IP address from that we want to administer the web server. This will be the only IP -address we will allow to connect to ssh (port 22) and the remote administration port of the arkeia backup (port 617) software (ple ase change the IP -address 10.10.10.1 to yours):

```
FW_TRUSTED_NETS=”10.10.10.1,22 10.10.10.1,617”
```

To be able to receive our DNS information we will accept packets from defined name servers: Those have been set up during the network configuration.

```
FW_ ALLOW_INCOMING_HIGHPORTS_UDP=”DNS”
```

To log all events (accepted, rejected and dropped packets) we have to change the following entry:

```
FW_LOG_DROP_CRIT=”yes”
FW_LOG_ACCEPT_CRIT=”yes”
FW_LOG_ACCEPT_ALL=”yes”
FW_LOG_DENY_ALL=”yes”
```

If you don’t want to have this much logging information you should reduce logging by setting the following entries:

FW_LOG_DROP_CRIT=”yes”
FW_LOG_ACCEPT_CRIT=”yes”
FW_LOG_ACCEPT_ALL=”no”
FW_LOG_DENY_ALL=”yes”

If you want to be able to ping your server you have to set the following:

FW_ ALLOW_PING_FW=”yes” otherwise leave it to “no”

To disallow traceroutes (see *man traceroute*), i.e. someone on the net could track the way packets go from his computer to your server, you have the following setting:

FW_ALLOW_FW_TRACEROUTE=”yes”

If you want to ignore broadcasts, especially when you have MS Windows machines in the same subnet, because they like to send a lot of them and that could fill your log files, make the following change:

FW_IGNORE_FW_BROADCASTS=”yes”

If you want to add special cu stomized rules, you could save them in an extra file and point to this file to get your rules included:
e.g.
FW_CUSTOMRULES=/etc/rc.config.d/firewall2 -custom.rc.config

After having made these changes start the firewall:

/sbin/SuSEfirewall2 start

the SuSEfirewall2 script can take several options:

start	starts the firewall
stop	unloads all rules
test	loads the rules and only logs to syslog what would have happened if the rules were active
close	closes all incoming connections
file filename	is the same as “start” but uses an alternative configuration file
status	displays all current iptables rules
debug	displays all the iptables commands generated from the script

Section	Task to do	Status/Value
4.28	Edit the configuration file /etc/rc.config.d/firewall2 .rc.config and perform the following steps:	
4.29	Change the variable value of “ FW_DEV_EXT” to your network device connected to the Internet e.g. “eth0”	
4.30	Change the variable value of “ FW_DEV_INT” to your trusted	

	network device e.g. "eth1"; if you have none leave it empty!	
4.31	Leave the value of the variable " FW_DEV_DMZ" empty since it is a local firewall.	
4.32	Check out the variable " FW_ROUTE" it has to be set to "no"	
4.33 4.34	Check out the variable " FW_MASQUERADE" it has to be set to no Change the value of the va riable "FW_SERVICES_EXT_TCP" to "www 443" (www is port 80 and 443 is the SSL port)	
4.35	Change the value of the variable "FW_TRUSTED_NETS" to the IP addresses from which you want to administer the server and the destination port, e.g. "1.1.1.1,22 1.1.1.1,617 " Port 617 is the default port for remote administration of the arkeia backup server.	
4.36	Change the value of the variable "FW_ALLOW_INCOMING_HIGHPORTS_UDP" to "DNS"	
4.37	Change the value of the variables "FW_LOG_*" to "yes" This is intended for logging every thing. If you don't want to have excessive logs then do the following settings: Set "FW_LOG_DROP_CRIT" to yes Set "FW_LOG_ACCEPT_CRIT" to yes And leave the other "FW_LOG_" entries to "no"	
4.38	Change the value of the variable "FW_ALLOW_PING_FW" to yes if you need the server to respond to ping, but I highly recommend setting it to "no" or configuring your firewall between the web server and the Internet to allow pings only from special IP addresses.	
4.39	Change the value of the variable "FW_ALLOW_FW_TRACEROUTE" to "no" to disallow traceroutes	
4.40	Check the value of the variable "FW_IGNORE_FW_BROADCAST" it has to be set to "yes" especially if you have Windows machines in the same subnet!	
4.41	Change the value of the variable "FW_CUSTOMRULES" to a file containing customized filters if you need special settings; otherwise leave it empty.	

The configuration of the "SuSEfirewall" (ipchains) is not discussed here. Please read the appropriate documentation, which is included in the tar ball you downloaded.

Configuring secumod

The secumod is a kernel module (for kernels 2.2.x only) that gives you the possibility to restrict/control the following:

- o Definition of a trusted path, users are allowed to execute files only residing in these directories.
- o To protect the procfs i.e . normal users won't see other users processes.
- o Hardlink create Protection i.e. non -root users are not allowed to create hardlinks to files they don't own.
- o Symlink follow protection i.e. symlinks that are not owned by the process can't be followed unless they are owned by root.

- Fifo protection i.e. nobody is allowed to write to pipes he doesn't own unless they are owned by root
- Raw disk protection i.e. nobody is allowed to access raw block devices like /dev/sd*
- Process trace protection i.e. nobody is allowe d to trace another process with strace
- Syscall table checking i.e. while using insmod the syscall table is compared to a backup to be able to detect changes
- Socket protection i.e. actions performed on sockets are logged
- Additional logging i.e. any violatio n against these settings will be logged
- Capability settings i.e. you can set the global capability bounding

This security kernel module can provide additional security, but be aware that you can log yourself out of your server, too, if you make too secure changes! As consequence of such action it could be possible that booting from a floppy or even a new installation of the OS be required to gain access to the system.

Edit the configuration file /etc/rc.config.d/secumod.rc.config and perform the following changes. If you want to make your own custom settings, please refer to /usr/share/doc/packages/secumod/README.

Section	Task to do	Status/Value
4.42	Set "SECUMOD_PARAMS" to "hardlink=1 symlink=1 trace=1 systable=1 logging=1 texec=1 procfs=1 pipe=1 rawdisk=1 socket=1"	
4.43	Set "SECUMOD_CAPBITS" to "desktop" Be aware, that the strict mode would prevent logins of other users than root!	
4.44	Set "SECUMOD_TPATH" to "/bin /usr/bin /sbin" and be sure to include any directory you might need to execute files from!	
4.45	Test your configuration! *rcsecumod start* Try to perform everything you might need to do. To unload the module do the following: *rcsecumod stop* or *rmmod secumod*	
4.46	The setting "persist" for the "SECUMOD_PARAMS" will disable the possibility to unload the settin gs! Only use it if you are really sure that your settings don't block you out of your system. You might have a very secure sever, but it could happen that you can't administer it either!	
4.47	If you have tested your settings and are absolutely sure that all administrative tasks you need to perform and the original functionality still remains while using the module, you can let the system automatically load the module with the following: Change the "START_SECUMOD" value to "yes"	

Configuring sudo

With sudo you can assign rights to normal users to execute several files with other privileges. If you simply want to allow a user to perform specific administrative tasks then configure the /etc/sudoers file with the command *visudo*. For additional information refe r to the man page or the documentation provided in the /usr/share/doc/packages/sudo directory.

For a simple setup perform the following steps.

Section	Task to do	Status/Value
4.48 4.49	The user who should be able to use sudo has to be a member of the trusted group. If the user does not exist: *useradd –m –g trusted <username>* If it already exists: *usermod –g trusted <username>* Edit the /etc/sudoers file with the following command: *visudo*	
4.50	Define a command alias for the commands the user should be able to execute: Cmnd_Alias OP = /bin/ps, /bin/su –u wwwadmin, \ /usr/sbin/rcapache, /usr/sbin/rcsshd, /sbin/shutdown	
4.51	Define who may run the commands: <username> ALL=OP	
4.52	Set the following "Defaults" value: log_year, log_host, syslog=authpriv, syslog_badpri=alert, syslog_goodpri=alert, runas_default=nobody	

It is important to disable any other means by which normal users can execute files with other privileges, to control the usage of privileges. The options shown above only represent a few things you can set in the sudo configuration. For additional information have a look at the man page sudoers (5) and the documentation mentioned above.

Configuring password policy

The settings related to login and the password policy are set in the file /etc/login.defs. The "Harden SuSE" script will have done some changes to the file but there are still things to do. Change the following settings.

Section	Task to do	Status/Value
4.53	Check the "DEFAULT_HOME" setting it has to be set to "no", with that setting logins are not allowed if no home directory has been defined for the user.	
4.54	Change the "PASS_MIN_DAYS" setting to "2", that disallows the change of the password within two days after a change was required.	
4.55	Change the "PASS_MIN_LEN" setting to "8"	
4.56	Change the "PASS_MAX_LEN" setting to "127", since md5 password support is already installed. But recheck if you use an application that doesn't support md5 passwords.	
4.57	Check the "PASS_WARN_AGE" setting it should be set to "7"	
4.58	Change the "PASS_MAX_DAYS" setting to "30"	

Configuring Syslogd

Syslogd is the daemon responsible for logging system messages. The configuration used in this setup is the default configuration. However if you have a central logging server on another machine you might want to send the log entries directly to this host. For this purpose I recommend using syslogd -ng (www.balabit.hu/en/products/syslog -ng/).

Configuring the backup

The simplest way to do a backup is to tar all files. The s olution chosen here is to use arkeia [11] that is free for use on Linux and provides remote management tools for Linux and MS Windows. The documentation (/usr/share/doc/packages/arkeia) on how to set it up is quite good and easily understood. This software on ly works with SCSI tape drives.

If you want to use the remote management tools, then you have to make some security related changes. This is due to the fact that if you want to access the remote management daemon you open your system and can therefore be attacked. To minimize the risk you have to restrict access to the port of the daemon to the IP -address from which you want to connect to the server to administer your backup. Performing this still leaves the risk of someone being able to spoof (fake) your IP-address and to connect via this way. If this risk seems to be too high for your setup then you could use another variant by connecting via ssh, administering the backup via the command line tools and by restricting access to the daemon to localhost. The default configuration of arkeia is to deny any access. The first variant we will see is how to configure arkeia to able to administer it by using an ssh connection:

You have to edit the /usr/know/nlp/auth.cfg and add a line to allow connections from localhost, the standard entry looks like this:

```
*.*     DENY        *       *
```

so you have to add in above that line:

```
backup       ALLOW      localhost      root
```

Now you can connect via ssh to your server and use the *arkc* command line tool to administer your backup. Connections to the daem on are now only allowed for localhost and the user root.

To use the more insecure, but more comfortable way, by allowing one special host to remotely connect via the graphical remote administration tool, perform the following actions:

[11] Arkeia, Knox Software, www.arkeia.com/downloadfree.html

Section	Task to do	Status/Value
4.59	Edit the /usr/knox/nlp/auth.cfg. The default setting is to deny access to everyone. Set a rule to allow access via remote management tools from one or more IP addresses: <nameofbackupserver> ALLOW <IP address> <nameofuser> backup ALLOW 1.1.1 .1 wwwadmin	
4.60	Change your SuSEfirewall2 configuration as following: Add to the value of "FW_TRUSTED_NETS" the host from which you would like to perform the remote management and the destination port, which is 617: FW_TRUSTED_NETS=1.1.1.1,617	

Make some test backups, restore data and assure yourself that your installation works as intended. Keep a version of this backup in a safe place, I'd recommend in at least a distance of 3 miles in a bank safe or something similar, because then your backup could surv ive a local disaster.

Tripwire configuration

To be able to trace changes made to your system that could indicate that a breach of security has occurred, use tripwire [12].

The first step is to read the documentation of tripwire which resides in the /usr/share/doc/packages/tripwire directory.

The next step is to copy the sample configuration file to the tripwire directory:

cp /usr/share/doc/packages/tripwire/tw.conf.example.linux /var/adm/tripwire/tw.config

We take the sample configuration file as our sta rting point to make our individual configuration.

The configuration file is of the form:

Directory Options

The most common option is R. This tells tripwire to check for changes of the following attributes:

Permission bits
These are the rights that user s and groups have on the file

Userid, groupid
The userid of the owner of the file and the groupid of the groupownership.

Timestamps of modification and creation
The Time of the last modification and creation of the file

[12] Tripwire Inc, URL: www.tripwire.org

The signature
The signature of th e file that is created by tripwire

The inode number
This is a unique number for each filename. This is used to look up for information regarding the type, size, and location of the file and the userid of the owner of the file.

Reference count
The amount of references that are made to this file.

Let's have a look at the sample we just copied:

```
/ R
!/proc
/var
! /root
/root/bin
!/dev
!/etc/mtab
!/etc/ld.so.cache
```

This tells us that the directories/files with a preceding "!"(not) won't be checked and t hose without "!" will be checked. Those not checked, e.g. the /proc directory contains the processes and therefore the content is permanently changing. Therefore it doesn't make sense to check this directory because it would result in a very large list of changes that are normal behavior not providing us the information we want.

Now we will modify our configuration file to reflect our needs:

Sample configuration modified

```
/ R
!/proc
/var
! /root
/root/bin
!/dev
!/etc/mtab
!/etc/ld.so.cache

!/var/adm/tripwire/bin/databases R
>/var/log R
=/tmp N
/usr/local
/usr/local/httpd
/home
```

The changes we made are:
!/var/adm/tripwire/bin/databases R
We do not want the tripwire database itself to be included, since it changes when we run the program the first time and when we made changes we want to be included in the database. We keep a copy of the database on a read only media as reference.

>/var/log R
This checks modifications on log files but only if their size is decreased. That's definitely something that should make us getting nervous, since log files should only increase in size.

=/tmp N
This tells tripwire to ignore changes of content, but not to ignore the directory itself. This is necessary since the content of the /tmp directory will very likely change often.

/usr/local
We need to put an entry for each partition we create since tripwire requires that.

/usr/local/httpd
This is the directory of our web pages and resides on it's own partition.

/home
This tells tripwire to inspect the /home directory and every directory below. Using this we can easily detect if suddenly changes occur in user directories.

Now we need to initialize tripwire to build its database:

/var/adm/tripwire/bin/tripwire –initialize

That will take a while to complete, depending on the amount of files you have.

When finished, we make a copy of the database on a floppy disk (or other read only media):

Insert a floppy disk in your floppy drive.
mount /dev/fd0 /floppy
The databases' name always ends with the name of the machine, here Charlie.
cp /var/adm/tripwire/bin/databases/tw.db_Charlie /floppy

Now we need to copy the tripwire binaries to be sure that we always have unmodified programs to check the database:

cp /var/adm/tripwire/bin/ /floppy*
cp /var/adm/tripwire/tw.config

Now we need to protect our floppy:

umount /floppy
Take the floppy disk out of the drive and switch the write protecting switch on the floppy to write protect the floppy.
Put the floppy disk back into the drive

Now we want to have the check of our file system be performed regularly. Therefore we write a little shell script and let it be executed via cron:

/etc/cron.daily/tripwire

```
#!/bin/sh
```

```
mount /dev/fd0 /floppy
/floppy/tripwire –d /floppy/tw.db_charlie –c /floppy/tw.config > /dev/tty9
umount /floppy
```

This script will mount the floppy disk in the disk drive, check the file system against the database, print the result on console 9 (tty9) and after that unmount the floppy disk. Then you can switch to console 9 by pressing ALT+F9 and read the results. Feel free t o change the destination of the results to where you like it.

Then we put this script in the /etc/cron.daily directory, therefore it will get executed daily and name it e.g. tripwire. Don't forget to make it executable and protect it from other users by taking away all rights for others and the group:

chmod a-rwx,u+x /etc/cron.daily/tripwire
This takes the read, write and execute permissions from everybody and adds the executable right for the owner of the file, which is root.

Now list the directory cont ent (*ls –l*) and compare the permissions. They should look like the following:

```
---x------     1     root   root   x x x x      tripwire
```

Summary of the tripwire configuration steps:

Section	Task to do	Status/Value
4.61	Read the documentation of tripwire (usr/share/doc/pack ages/tripwire/README and the man pages) to fully understand how to set it up.	
4.62	Copy the sample configuration file to /var/adm/tripwire: *cp /usr/share/doc/packages/tripwire/tw.conf.example.linux /var/adm/tripwire/tw.config*	
4.63	Edit the tw.config file and a djust it to your needs. Have a look at the sample configuration here.	
4.64	Initialize tripwire: */var/adm/tripwire/bin/tripwire –initialize* It'll take a while to complete.	
4.65	Make a copy of the /var/adm/tripwire/bin/databases/tw.db_<machinename> either to a floppy disk if it fits, burn it on a CDROM or put it on a drive which can only mounted read -only after you put the db on it. Floppy: *mount /dev/fd0 /floppy* *cp /var/adm/tripwire/bin/databases/tw.db_Charlie /f loppy*	
4.66	Copy all the binaries from /var/adm/tripwire/bin/ and the configuration file /var/adm/tripwire/tw.config on the floppy or another media as stated above. *cp /var/adm/tripwire/bin/* /floppy* *cp /var/Adm/tripwire/tw.config /floppy*	
4.67	Write protect the floppy disk.	
4.68	Unmount and remount the floppy read -only (change the plastic	

	switch on the floppy to the read only position): *umount /floppy* *mount /dev/fd0 /floppy –r -n*	
4.69	Set up a daily cron job by putting a shell script that performs the tripwire check and sends you an e-mail with it's results in the /etc/cron.daily directory.	

Configuring lilo

LILO is the boot manager that is responsible for loading the kernel at boot time. Its configuration file is /etc/lilo.conf.

If you don't need to reboot your server remotely you can add some security by setting a password for lilo. This will result in the effect that your server won't start the kernel until you provided the password.

Be aware that you won't be able to reboot your server remotely if you set th is!

To achieve this we have to make the following entry in the /etc/lilo.conf file:

```
password=topsecret
```

Please choose your own password, do not use the example!

The next step is invoke lilo for making the changes according to the configuration file:

```
/sbin/lilo
```

This will result in an output like this:

```
Added linux *
```

This tells you that the configuration change has been applied successfully. Otherwise you would get an error messages indicating in which line of the configuration file the error occurred.

Last configuration step

The last step is to reboot the machine and perform changes to the BIOS to prevent booting from CDROM and floppy disk.

Section	Task to do	Status/Value
4.70	Reboot the machine.	
4.71	Enter the BIOS as described in section 1.	
4.72	Change the default boot sequence to disallow booting from floppy disk and save the setting.	
4.73 4.74	Enter the BIOS of the SCSI controller (you will get prompted to press a key sequence in order to gain access to it). Change the settings to disallow booting from the CDROM drive.	

5.Testing the configuration

External test

You should now look at the outside of your system. Examine what can be accessed on your server from the several locations connections could be done from (Management Client, normal internal user pc, external access from the internet). To be able to analyze this use the tool nmap[13] written by Fyodor that is available for Linux (series sec) and MS Windows NT [14].

Here we will see what we can do with nmap and how the results will look like. The outputs shown here reflect the configuration with remote access to the backup daemon from a remote administration host.

To be able to determine what our server (IP -address x.x.x.x) exposes to our administrative machine we run nmap from this host to scan all open por ts:

```
Nmap –P= -p 1-65535 x.x.x.x
```

The output should be like this:

nmap from your management machine

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on charlie (x.x.x.x):
(The 65531 ports scanned but not shown bel ow are in state: filtered)
Port      State      Service
22/tcp     open       ssh
80/tcp     open       http
443/tcp    open       https
617/tcp    open       unknown
```

The output tells us that from this machine we have access to the following services:
ssh for remote administration
http for normal web access
https for encrypted web access
unknown/617 for remote access to the arkeia backup server

Now we will have a look what is exposed to a normal computer connected to our network:

nmap from a normal machine within your network

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on charlie (1.1.1.1):
(The 65531 ports scanned but not show n below are in state: filtered)
Port      State      Service
80/tcp     open       http
443/tcp    open       https
```

[13] written by Fyodor, URL: www .insecure.org/nmap/
[14] MS Windows NT port written by eEYE, www.eeye.com/html/Research/Tools/nmapNT.html

This tells us that a normal computer in our network can connect to the server via http and https, so he can browse the web pages we offer.

Next we have a look at the exposures to a host located on the internet:

nmap from a machine located on the internet

Note: The result shown here may not reflect yours, since your protection measures might be different.

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on xx (1xx.1xx.1xx.1xx):
(The 65531 ports scanned but not shown below are in state: filtered)
Port      State     Service
80/tcp     open       http
443/tcp    open       https
```

This tells us that a computer connecting from the Internet to our server can access http and https, so he can browse the web pages we offer.

Section	Task to do	Status/Value
5.1	Run *nmap –P0 –p 1-65535 <IP address of your server>* from your remote management machine. The output should be like this. Compare the output to what you wanted to allow to be seen and done from this perspective.	
5.2	Run *nmap –P0 –p 1-65535 <IP address of your server>* from a normal machine within your network that shouldn't have any extended access. The output should be like this. Compare the output to what you wanted to allow to be seen and done from th is perspective.	
5.3	Run *nmap –P0 –p 1-65535 <IP address of your server>* from a machine located on the Internet, if the server should be accessible via internet, that shouldn't have any extended access. The output should be like this. Compare the output to what you wanted to allow to be seen and done from this perspective.	

To provide remote administrative access from the Internet is generally a bad idea. If you really need this then better check your system daily.

If you want to perform a complete security test then you should have a look at the Open Source Security Testing Methodology Manual (OSSTMM) the most recent development version is at uk.osstmm.org/osstmm.htm or download the complete v.1.5 manual from http://uk.osstmm.org/osstmm.rtf.zip .

Functionality Test

Now since you have tested the external side of your serve r, do a final recheck on its functionality. Test all of its functions to assure that you can administer securely your system and have access to everything you require. If something should fail review the configuration and perform the test described in sect ion 5 again.

Section	Task to do	Status/Value
5.4	Test all of the functions you require to maintain your system.	
5.5	Should something fail review the configuration and perform the test described in section 5 again.	

6.Periodical review / updates

It is important to perform periodical reviews of your system configuration and log files. You should establish a policy describing appropriate measures. One issue is to adjust the system time, if it shows a discrepancy to the official time. This could be done using xn tp, but since a manual review is required anyway and any additional program could possibly pose a risk, I recommend performing this manually in accordance with the policy.
It is very important that you check for updates especially security updates from Su SE (http://www.suse.de/de/support/security/) and to subscribe to the security maillinglist (suse -security and suse -security -announce at http://www.suse.de/en/support/mailinglists/index.html). The update procedure is the same as described in the section "Updating the system - Updates". For completeness the steps are shown here again.

Updates and their descriptions can be found on t he SuSE homepage (http://www.suse.com/us/support/download/updates/index.html) with links to the files on their ftp server (ftp.suse.com/pub/suse/i386/update/7.1). Perform the following steps to download, verify and install the packages:

Section	Task to do	Status/Value
6.1	Download the updates via ftp from ftp.suse.com/pub/suse/i386/update/7.1	
6.2	Check the rpm signature: *rpm –v –checksig <name of rpm>*	
6.3	Compare the fingerprint with the one on the SuSE homepage http://www.suse.de/de/support/security/	
6.4	Update the packages: with *yast* Choose Packet Management	
	Select Install Packages	
	Change source to directory	
	Enter the directory where the downloaded rpms reside	
	Check the packages	
	Press F10 to install the packages	

There are other possibilities t o update packages, but you can't verify the signatures by using these ways. One way is to start *yast*: Choose Packet Management -> Select Install Packages -> Change source to ftp -> Enter: ftp.suse.com/pub/suse/i386/update/7.1 -> Enter the series directory -> Select your packages -> Press F10 to install. The other one is to use *yast2* (in text mode): Choose Online Update -> Manual Update -> Recheck the chosen packages ->Install.

Note: If you install pa ckages from CDROM at a later point of time check out if there is an update for it, otherwise you might use a version that has known vulnerabilities.

Seccheck

You have installed the package “seccheck” that automatically starts checks periodically on your system and sends you an e -mail if changes occurred since the last time the checks have been performed.

Now we will discuss what is performed and where the scripts are located.

The user the mail is sent to is by default the user root. This can be changed by modifying the “SECCHK_USER=” entry in the /etc/rc.config file. The scripts get automatically executed by the seccheck file in the /etc/cron.d directory via cron. If you want to disable the execution of these scripts then simply remove the /etc/cron.d/se ccheck files by using the following command:

rm /etc/cron.d/seccheck

The scripts are located in the /usr/lib/seccheck directory. There are 5 scripts:

checkneverlogin
security-control.sh
security-daily.sh
security-weekly.sh
security-monthly.sh

The checkneverlogin file is a script that checks whether user do exist that never logged in. The security -control.sh is the control script that invokes the other scripts depending on the parameter that is given to the control script:

security-control.sh daily would invoke the security -daily script
security-control.sh weekly would invoke the security -weekly script
security-control.sh monthly would invoke the security -monthly script

The following checks are performed by the daily script (as described in the /usr/share/doc/packages/seccheck/README):

/etc/passwd check	:	length/number/contents of fields, accounts with same uid accounts with uid/gid of 0 or 1 beside root and bin

This is security related since the numerical userid and groupid grants the according rights to an account. Therefore if you have multiple users with the same userid and/or groupid they have the same rights. To be able to distinguish between users it is necessary that each user has its own userid.

/etc/shadow check	:	length/number/conte nts of fields, accounts with no password

Accounts without passwords are definitively a security threat and should never exist. An account without password can be easily used by unauthorized persons to gain access to your system.

/etc/group check	:	length/number/contents of fields

This checks which users belong to which groups and shows you changes within your groups.

user root checks	:	secure umask and PATH

This checks whether the umask representing the rights that a newly by root created file gets. Since files created by root normally contain sensitive information they shouldn't be readable but at least not write able by other users.

/etc/ftpusers : checks if important system users are put there
This check determines if normal ftp access is grante d for high privilege users such as root. The users listed in the /etc/ftpusers file are not allowed to connect via normal ftp.

/etc/aliases : checks for mail aliases which execute programs
This check examines whether there are user mail aliases defined that automatically execute scripts. This is important since this could allow an attacker to send an e -mail that could force a script that gets automatically executed if an e -mail arrives for that user to perform actions on the local system.

.rhosts check : checks if users' .rhosts file contain + signs
A + sign in a .rhost file allows any user from any system to login without the need to provide a password. Therefore this should absolutely not be in a .rhost file.

homedirectory : checks if homedirec tories are write able or owned by someone else
This is to check whether other users own other users home directory or may write to others home directories. This could significantly affect the security of your system since it might be possible to get ot her users to execute scripts with their privileges or to obtain information from them.

dot-files check : checks many dot-files in the homedirectories if they are write able or owned by someone else

Dot files, e.g. ".rhosts" do not get displayed wit h the normal "*ls*" command. To see them you would have to invoke "*ls –l*". So they are sometimes referred as hidden files. The reason for checking the permissions of those is equivalent to the check above.

mailbox check : checks if user mailboxes are own ed by user and unreadable
This is to determine whether e -mails could get read by other users.

NFS export check : exports should not be exported globally
NFS the network file system is used to export directories in that way that other users could mount them and therefore have access to their contents. This checks whether access to those exported directories is limited or not.

NFS import check : NFS mounts should have the "nosuid" option set
This checks whether the nosuid option is used with nfs. The no suid options disables the usage of suid files on exported file systems, to prevent others executing them with root privileges.

promisc check : checks if network cards are in promiscious mode
This checks whether network cards are in the promiscuous mode. The promiscuous is used when a network device is used to actively sniff the data that is send by the device. This would be an indicator for someone listening on your network traffic and trying to gain sensitive information.

list modules : just lists loaded modules

This gives you an overview of all modules that are currently loaded on your system.

list sockets : just lists open ports
This shows you all sockets, therefore ports that provide a possibility to connect to your system.

The weekly script performs the following checks:

password check : runs john to crack the password file, user will get an email notice to change his password asap
This is used to determine the strength of the passwords you use on your system. This check is disabled bec ause we did not install john (a password cracking program) because it could serve as utility for a local user to compromise the system.

rpm md5 check : checks for changed files via rpm's md5 checksum feature
this checks the md5 checksums of rpm files on your system and can therefore indicate manipulations on your files.

suid/sgid check : lists all suid and sgid files
This simply list you all suid and sgid files. These are files that can be executed by a non root user with root privileges. Therefore th ey can pose a risk. If the script informs about such files you should check whether you really need them to be suid/sgid.

exec group write : lists all executables which are group/world write able
This lists all executable files that are group/world writ e able. These files can pose a risk since the normal user could manipulate them and can force other users to perform manipulations they didn't intend to perform.

write able check : lists all files which are world write able (incl. above)
This is a more general check to detect all files that can be written by anyone to. This resulting list shouldn't include system configuration files since then everybody could manipulate the systems behavior.

device check : lists all devices
This gives you an overview of all devices that exist on your system. Should changes occur and you did not install any devices you should get very suspicious and perform a security analysis of your whole system.

The monthly script

The monthly script performs all of the checks abov e but without comparing the results to previous checks. Therefore you get once a month a complete list of the results of the above checks.

7.Comments

The setup described within this document is a secure basic configuration, but it may not fulfill your security needs since they might be higher. Additional security measures should be taken depending on the security level required for your environment. Since security measures taken at a certain point of time cannot provide protection against future risks and v ulnerabilities, there cannot be a guarantee that the system might not get compromised one day or another. There is no such thing as a 100% percent secure system.

You should always make copies of your configuration files prior to changing them. This can save you a lot of time.

The highest risks are services that are accessible from the Internet due to the number of potential attackers. You should therefore check the security site of the distributor (www.suse.de/de/support/security), subscribe to the security mailing lists and keep up to date with your knowledge and system updates (www.suse.com/us/support/download/updates/index.htm l). However this should not mean that the internal security could be neglected, since various statistics show that many abuses of systems have been done from within an organization (e.g. www.gocsi.com/prelea_000321.htm).

Assure that host-based protection measures are not your sole line of defense. I recommend setting up a multi line defense system because otherwise there is only one obstacle to compromising your system. The more often you perform reviews of your systems the more likely you will detect intrusions fast.

Things many people forget about are the simplest. Your server is always only protected as well as your management computers are. If the computer you perform your administ rative tasks from is not protected like your sever is, someone might take the easy road and compromise your workstation and get all the information needed to access your systems directly from you.

8.Appendix

Examples

The examples shown here are not for direct use for your system. These need modifications dependant on your IP addresses and special configurations you might have. Don't simply copy and use them! The examples are modified copies of the original configuration files. The comments included in th e files have not been tampered with, even though they might contain grammatical errors. Sanitation has been done on the files to protect the identity of the environment where this setup is used.

/usr/knox/nlp/auth.cfg

```
################################# #####################################
##
# File: auth.cfg
# Default Authorization file for Knox server applications (C) Knox 97
#
# The values in this file are used to allow or deny access to knox servers.
# For every connexion the file is scanned sequen tially. The first line whose
# left hand side part matches the connexion is used for granting or denying
# access to the server. If no line matches the connexion, access is denied.
#
# The syntax of this file is:
#  SERVER_LIST.SERVICE_LIST  ALLOW/DENY  HO  ST_LIST[RESERVED PORT]
USER_LIST
#
#        SERVER_LIST is a list of one or more servers separated by '|'
#        SERVICE_LIST is a list of one or more services separated by '|'
#        ALLOW_DENY is "ALLOW" or "DENY"
#        HOST_LIST is '*' or a  list of one or more hosts separated by '|'
#        [RESVPORT] optional: [1] if connected from a reserved port, else [0].
#        USER_LIST is '*' or a list of one ore more remote user names
#             separated by '|'
########################## #########################################
##
*.*     ALLOW   1.1.1.1    *
*.*     DENY    *       * # By default nothing is permitted
####################################################################
##
# Keep this last line, otherwise CR+LF en vironnements will fail
```

/etc/login.defs

```
# /etc/login.defs  - Configuration control definitions for the login package.
#
# All items are optional  - if not specified then the described action or
# option will be inhibited.
```

```
#
# Comment lines (lines beginning  with "#") and blank lines are ignored.
#
# Delay in seconds before being allowed another attempt after a login failure
#
FAIL_DELAY      3
#
# Enable logging and display of /var/log/faillog login failure info.
#
FAILLOG_ENAB    yes
#
# Enable display of unknown usernames when login failures are recorded.
#
LOG_UNKFAIL_ENAB        "no"
 # Enable logging and display of /var/log/lastlog login time info.
#
LASTLOG_ENAB    yes
#
# Enable additional checks upon password changes.
#
OBSCURE_CHECKS_ENAB             yes
#
# If defined, ":" delimited list of "message of the day" files to
# be displayed upon login.
#
MOTD_FILE       "/etc/motd"
#MOTD_FILE      /etc/motd:/usr/lib/news/news -motd
#
# If defined, file which maps tty line to TERM environment parameter.
# Each line of the file is in a format something like "vt100  tty01".
#
TTYTYPE_FILE    "/etc/ttytype"
 # If defined, login failures will be logged here in a utmp format.
# last, when invoked as lastb, will read /var/log/btmp, so...
#
#FTMP_FILE      /var/log/ btmp
#
# If defined, file which inhibits all the usual chatter during the login
# sequence.  If a full pathname, then hushed mode will be enabled if the
# user's name or shell are found in the file.  If not a full pathname, then
# hushed mode will be enabled if the file exists in the user's home directory.
#
#HUSHLOGIN_FILE .hushlogin
HUSHLOGIN_FILE  "/etc/hushlogins"
#
# The default PATH settings.
#
ENV_PATH        "/usr/local/bin:/usr/bin:/bin"
# The default PATH settings for root:
#
ENV_ROOTPATH    "/sbin:/bin:/usr/sbin:/usr/bin"
```

```
 # Terminal permissions
#
#      TTYGROUP        Login tty will be assigned this group ownership.
#      TTYPERM         Login tty will be set to this permission.
#
# If you have a "write" program which is "setgid" to a special   group
# which owns the terminals, define TTYGROUP to the group number and
# TTYPERM to 0620.  Otherwise leave TTYGROUP commented out and assign
# TTYPERM to either 622 or 600.
#
TTYGROUP        "tty"
TTYPERM         "0620"
# Password aging controls:
#
#      PASS_MAX_DAYS   Maximum number of days a password may be used.
#      PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#      PASS_MIN_LEN    Minimum acceptable password length.
#      PASS_WARN_AGE   Number of days warning   given before a password expires.
#
PASS_MAX_DAYS   40
PASS_MIN_DAYS    2
PASS_MIN_LEN     8
PASS_WARN_AGE   7
#
# If compiled with cracklib support, where are the dictionaries
#
CRACKLIB_DICTPATH       "/usr/lib/cracklib_dict"
# Min/max values for auto matic uid selection in useradd
#
UID_MIN         500
UID_MAX         65535
# Min/max values for automatic gid selection in groupadd
#
GID_MIN         100
GID_MAX         65535
# Max number of login retries if password is bad
#
 LOGIN_RETRIES   "3"
# Max time in seconds for login
#
LOGIN_TIMEOUT   "60"
# Maximum number of attempts to change password if rejected (too easy)
#
PASS_CHANGE_TRIES       "3"
# Warn about weak passwords (but still allow them) if you are root.
#
PASS_ALWAYS_WARN        "yes"
# Number of significant characters in the password for crypt().
# Default is 8, don't change unless your crypt() is better.
# Ignored if the "md5" option is given to the pam_pwcheck module.
#
```

```
PASS_MAX_LEN    8
 # Require password before chfn/chsh can make any chan  ges.
#
CHFN_AUTH       "yes"
# Which fields may be changed by regular users using chfn   - use
# any combination of letters "frwh" (full name, room number, work
# phone, home phone).  If not defined, no changes are allowed.
# For backward compatibility, "yes " = "rwh" and "no" = "frwh".
#
#CHFN_RESTRICT  "rwh"
# Should login be allowed if we can't cd to the home directory?
# Default is yes.
#
DEFAULT_HOME    "no"
# This enables userdel to remove user groups if no members exist.
USERGROUPS_ENAB         "yes"
```

/etc/sudoers

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#

# Host alias specification

# User alias specification

# RunAs alias
Runas_Alias OP = root, operat or

# Cmnd alias specification
Cmnd_Alias     SHELLS = /bin/sh
Cmnd_Alias     SU = /bin/su
# User privilege specification
root    ALL=(ALL) ALL
fex     ALL = SHELLS
Defaults log_year, log_host
#logfile=/var/log/log.sudo,
Defaults syslog=authpriv, syslog_ba dpri=alert, syslog_goodpri=alert, runas_default=nobody
```

/etc/rc.config.d/secumod.rc.config

```
#
# Should the secumod be loaded at system startup to restrict your
# system? (This param will be ignored, if rcsecumod is called manually.)
# For more information  see /usr/share/doc/packages/secumod/README
#
START_SECUMOD="no"
```

```
#
# Parameters for the module  - see /usr/share/doc/packages/secumod/README
# Default is "hardlink=1 symlink=1 trace=1 systable=1 logging=1"
# Think twice before setting persist=1!
#
SECUMOD_P ARAMS="socket=1 rawdisk=1 pipe=1 procfs=1 texec=1 hardlink=1
symlink=1
trace=1 systable=1 logging=1"

#
# Use kernel capabilities mode:
#       default, desktop, strict, paranoid
# Default is not to use capabilities.
#
SECUMOD_CAPBITS="desktop"

#
# If you enabled the check for trusted executable paths (texec=1),
# please set SEUMOD_TPATHS to a list of paths, where users are allowed
# to execute binaries. You may want to remove /sbin and /usr/sbin
# from the below example.
#
SECUMOD_TPATHS="/bin /usr/bin /s bin /usr/lib/man -db/"
```

/etc/inetd.conf

Everything should be commented out, i.e. you have a leading "#".

```
# See "man 8 inetd" for more information.
#
# If you make changes to this file, either reboot your machine or send the
# inetd a HUP signal with "/s bin/init.d/inetd reload" or by hand:
# Do a "ps x" as root and look up the pid of inetd. Then do a
# "kill -HUP <pid of inetd>".
# The inetd will re -read this file whenever it gets that signal.
#
# <service_name> <sock_type> <proto> <flags> <user> <server_  path> <args>
#
# echo  stream  tcp    nowait  root    internal
# echo  dgram   udp     wait    root    internal
# discard       stream  tcp     nowait  root    internal
# discard       dgram   udp     wait    root    internal
# daytime       stream  tcp     nowait  root    internal
# daytime       dgram   udp     wait    root    internal
# chargen       stream  tcp     nowait  root    internal
# chargen       dgram   udp     wait    root    internal
# time  stream  tcp    nowait  root    internal
# time  dgram   udp     wait    root    internal
#
```

```
# These are standard services.
#
# ftp   stream  tcp    nowait  root    /usr/sbin/tcpd  wu.ftpd   -a
# ftp   stream  tcp    nowait  root    /usr/sbin/tcpd  proftpd
# ftp   stream  tcp    nowait  root    /usr/sbin /tcpd  in.ftpd
#
# If you want telnetd not to "keep -alives" (e.g. if it runs over a ISDN
# uplink), add " -n".  See 'man telnetd' for more details.
# telnet       stream  tcp    nowait  root    /usr/sbin/tcpd  in.telnetd
# nntp  stream  tcp    nowait  ne ws    /usr/sbin/tcpd  /usr/sbin/leafnode
# smtp  stream  tcp    nowait  root    /usr/sbin/sendmail    sendmail  -bs
# printer      stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/bin/lpd   -i
```

/etc/hosts.allow

```
# See tcpd(8) and hosts_access(5) for a d escription.
#(ALL EXCEPT in.fingerd) EXCEPT in.identd : ALL : (safe_finger   -l @%h 2>&1| \
#              /bin/mail -s "%d-%h %u" root) &
SSHD: 196.1.2.3 196.1.2.6 196.1.2.88
```

/etc/hosts.deny

```
#
# See tcpd(8) and hosts_access(5) for a description.
ALL: ALL
```

ps –ax everything disabled:

```
Listing of all current processes
UID       PID  PPID  C STIME TTY          TIME CMD
root        1     0  0 Jun07 ?        00:00:10 init [3]
root        2     1  0 Jun07 ?        00:00:00 [kflushd]
root        3     1  0  Jun07 ?       00:00:05 [kupdate]
root        4     1  0 Jun07 ?        00:00:00 [kswapd]
root        5     1  0 Jun07 ?        00:00:00 [mdrecoveryd]
root      464     1  0 Jun07 ?        00:00:00 /usr/sbin/ippl
root      474   464  0 Jun07 ?        00:00:04 /usr/sbin/ippl
nobody    475   474  0 Jun07 ?        00:00:00 /usr/sbin/ippl
nobody    476   474  0 Jun07 ?        00:00:00 /usr/sbin/ippl
root      502     1  0 Jun07 ?        00:00:36 /sbin/syslogd
root      506     1  0 Jun07 ?        00:00 :41 /sbin/klogd  -c 2
root      680     1  0 Jun07 ?        00:00:01 /usr/sbin/cron
root      951     1  0 Jun07 tty2     00:00:00 login   -- root
root      952     1  0 Jun07 tty3     00:00:00 /sbin/mingetty tty3
root      953     1  0 Jun07 tty4     00:00:00 /sbin/mingetty tty4
root      954     1  0 Jun07 tty5     00:00:00 /sbin/mingetty tty5
root      955     1  0 Jun07 tty6     00:00:00 /sbin/mingetty tty6
root     2265     1  0 17:40 tty1     00:00:00 login   -- root
root     2266  2265  0  17:42 tty1     00:00:00  -bash
```

apache config diff:

```
--- httpd.conf.orig	Sat Aug 18 19:27:55 2001
+++ httpd.conf.last.140901	Sat Sep 15 00:36:43 2001
@@ -141,2 +141,2 @@
-MinSpareServers 1
-MaxSpareServers 1
+MinSpareServers 3
+MaxSpareServers 10
@@ -148 +148 @@
-StartServers 1
+StartServers 10
@@ -509 +509 @@
-#ServerAdmin root@localhost
+ServerAdmin serveradmin@mydomain.com
@@ -545,4 +545 @@
-    AuthUserFile  /etc/httpd/passwd
-    AuthGroupFile /etc/httpd/group
-
-    Options -FollowSymLinks +Multiviews
+    Options none
@@ -550 +547,2 @@
-
+    order deny,allow
+    deny from all
@@ -573 +571 @@
-    Options Indexes -FollowSymLinks +Includes MultiViews
+    Options None
@@ -585 +583 @@
-    Order allow,deny
+    Order deny,allow
@@ -600,3 +598,3  @@
-<Files /usr/local/httpd/htdocs/index.htm*>
-	Options -FollowSymLinks +Includes +MultiViews
-</Files>
+#<Files /usr/local/httpd/htdocs/index.htm*>
+#	Options -FollowSymLinks +Includes +MultiViews
+#</Files>
@@ -607,6 +605,6 @@
-<Files test.php3>
-	Order deny,allow
-	deny from all
-	allow from localhost
-</Files>
-
+#<Files test.php3>
+#	Order deny,allow
+#	deny from all
+#	allow from localhost
+#</Files>
+#
```

```
@@ -619,3 +617,3 @@
-<IfModule mod_userdir.c>
-    UserDir public_html
-</IfModule>
+#<IfModule mod_userdir.c>
+#    UserDir public_html
+#</IfModule>
@@ -652 +650 @@
-AccessFileName .htaccess
+#AccessFileName .htaccess
@@ -837 +835 @@
-ServerSignature On
+ServerSignature off
@@ -854 +852 @@
-        Options Indexes MultiViews
+        Options None
@@ -868 +866 @@
-    ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
+#    ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
@@ -870 +868 @@
-<IfModule mod_perl.c>
+#<IfModule mod_perl.c>
@@ -874 +872 @@
-    ScriptAlias /perl/          "/usr/local/httpd/ cgi-bin/"
+ #   ScriptAlias /perl/          "/usr/local/httpd/cgi -bin/"
@@ -876,2 +874,2 @@
-    ScriptAlias /cgi-perl/      "/usr/local/httpd/cgi-bin/"
-</IfModule>
+  #  ScriptAlias /cgi-perl/      "/usr/local/httpd/cgi-bin/"
+#</IfModule>
@@ -882,6 +880,6 @@
-    <Directory "/usr/local/httpd/cgi-bin">
-        AllowOverride None
-        Options None
-        Order allow,deny
-        Allow from all
-    </Directory>
+#   <Directory "/usr/local/httpd/cgi-bin">
+#        AllowOverride None
+#        Options None
+#        Order allow,deny
+#        Allow from all
+#    </Directory>
@@ -896,5 +894,5 @@
-<Location /cgi-bin>
-AllowOverride None
-Options +ExecCGI -Includes
-SetHandler cgi-script
-</Location>
+#<Location /cgi-bin>
+#AllowOverride None
```

```
+#Options +ExecCGI -Includes
+#SetHandler cgi-script
+#</Location>
@@ -1253 +1251 @@
-    Allow from localhost
+    Allow from all
```

/etc/sshd/sshd_config

With this configuration you can use MindTerm as well and therefore you can replace ftp with sftp even though you might have management client machines running a different OS.

```
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
# This is the sshd server system-wide configuration file.  See sshd(8)
# for more information.
Port 22
Protocol 2
HostKey /etc/ssh/id_dsa
LoginGraceTime 300
PermitRootLogin no
#
# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
RhostsRSAAuthentication no
StrictModes yes
X11Forwarding no
X11DisplayOffset 10
PrintMotd yes
KeepAlive yes
# Logging
SyslogFacility AUTH
LogLevel INFO
#
RhostsAuthentication no
#
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
PermitEmptyPasswords no
# Uncomment to disable s/key passwords
ChallengeResponseAuthentication no
Subsystem      sftp    /usr/lib/ssh/sftp-server
```

References:

Heuse, Marc. "SuSEfirewall2" URL:
http://www.suse.de/~marc/SuSE.html (18 July 2001)

Heuse, Marc. "Harden SuSE" URL:
http://www.suse.de/~marc/SuSE.html (18 July 2001)

Heuse, Marc. "Installation of a Secure Web Server", Oct. 24 2000. URL:
http://www.suse.de/en/linux/webserver/index.html (18 July 2001)

SuSE Inc., "Website" URL:
http://www.suse.com (18 July 2001)

SuSE Corp., "Hardware Support Database" URL:
http://cdb.suse.de/cdb_english.html (18 July 2001)

SuSE Corp., "SuSE Linux: Security Announcements" URL:
http://www.suse.de/de/support/security/ (18 July 2001)

SuSE Inc., "SuSE Linux: Updates, Patches, Bugfixes" URL:
http://www.suse.com/us/support/download/updates/index.html (18 July 2001)

The SANS Institute, Securing Linux Step by Step, Version 1.0
ISBN: 0-9672992-0-9

Brotzmann,Lee / Pomeranz, Hal, Linux/Solaris Practicum, May 2001

Herzog, Pete. "Open Source Security Testing Methodology Manual", v.1.5 URL:
http://uk.osstmm.org/osstmm.rtf.zip (18 July 2001)

BalaBit IT Ltd., "syslog -ng" URL:
http://www.balabit.hu/en/products/syslog -ng/ (18 July 2001)

Computer Security Institute, "2001 Computer Crime and Security Survey", March 12, 2001.
URL: http://www.gocsi.com/prelea_000321. htm (18 July 2001)

Fyodor, "Nmap" URL:
http://www.insecure.org/nmap/ (18 July 2001)

EEYE, "NmapNT" URL:
http://www.eeye.com/html/Research/To ols/nmapNT.html (18 July 2001)

Engelschall, Ralf S. "F.A.Q" URL:
http://www.modssl.org/docs/2.8/ssl_faq.html#ToC28 (18 July 2001)

Lavigne, Dru. "An introduction to Unix permissions" URL:
http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html?page=1 . (10 Sept. 2001)