Securing PowerDNS on Debian GNU/Linux 3.0
Simon Østengaard
August 26th 2004
GCUX Practical Assignment
Version 2.0
Option 1 - Securing Unix Step by Step

Contents

1 Abstract

In this paper I will describe how to build, deploy, and maintain a secure DNS server with PowerDNS and Debian GNU/Linux. The paper will also include general information on how to enhance security on a default Debian GNU/Linux installation. In the process of securing the server I will perform a risk analisys of the system to give you a picture of how critical the data on the system are.

The document will supply the needed instructions to verify that the server is secure and stays secure. I will describe how to manage security in Debian GNU/Linux on an ongoing basis. I will describe some procedures that can be used to protect a GNU/Linux server against the most common vulnerabilities in GNU/Linux such as buffer overflows and format string vulnerabilities. The procedures will also include information on how to avoid and detect the installation of LKM and regular rootkits on a GNU/Linux system.
Reading this paper requires substantial knowledge of GNU/Linux. I suggest the paper to be used by experienced Linux system administrators.

2 Server Specification and Risk Mitigation Plan

2.1 Services and software

The main function of this server is a DNS server. I have chosen to run the PowerDNS [1] server, to service DNS requests. The server will also need some additional services to make the DNS server operational. The PowerDNS server needs a backend to contain the zone data. Although powerdns can be configured to use text file backends, it will only show its full strength when using a database backend. I have chosen to use the MYSQL database server as the backend for this server. To make maintenance and upgrades easier I will install almost all the software from the Debian GNU/Linux distribution. I only install packaged software because it will let me get a list of software packages installed on a server. This is critical information in terms of upgrading and patching systems.
Below is a list of the software I will use to build the server. Software that is not on the list but is used later in this document will be installed from the Debian GNU/Linux 3.0 stable packages. I will use the most recent versions of the packages from this release.

- Debian GNU/Linux 3.0 stable
- Postfix MTA from Debian GNU/Linux 3.0 stable
- Aide intrusion detection tool from Debian GNU/Linux 3.0 stable
- MySQL database server from Debian GNU/Linux 3.0 stable
- Ntpdate timesynchronization software from Debian GNU/Linux 3.0 stable

- Openssh server from Debian GNU/Linux 3.0 stable
- Syslog-ng next generation syslog server from Debian GNU/Linux 3.0 stable
- Stunnel SSL software from Debian GNU/Linux 3.0 stable
- Linux kernel 2.4.26
- PowerDNS 2.9.16
- Grsecurity 2.0 for the 2.4.26 version of the Linux kernel[4]
- Knoppix-std 0.1 to audit the server security - this will not be installed on the server
- Nessus 2.0.12 to audit the server security - this will not be installed on the server

There is going to be two users with shell access on the server. One user with shell access via console and openssh and enabled password to use for remote and local administration of the server. This user will also be able to use su to gain root access to the server. The other active user on the server is the super user root. The root user will not be able to log in via ssh or from the console once the server is ready for production.
During the installation and configuration of the server I need access to a development server. I will use the development server to check ISO checksums, burn CDROMs, compile the Linux kernel and to scan the server with the nessus vulnerability scanner. I will not describe how to secure the development server and I will presume that the development server is secure and that it is located on the internal network.

2.2 Location in the production network

The server will be placed in a DMZ network when it is ready for production. The firewall protecting the DMZ allows new and established [1] connections from the internet to the server on tcp and udp port 53. Access from the internet to tcp port 53 is not strictly neccesary unless someone has to make zonetransfers from your nameserver, but many registries[2] require that they are able to make queries against the server on tcp port 53 to register the server as an authorative DNS server. The firewall protecting the DMZ should have the following rules concerning this server:

- drop invalid packets before they are processed further
- allow new and established connections from servers used for remote administration to the server port 1803 tcp

[1] I refer to the netfilter connection states "new", "established", "related", and "invalid"[11]

[2] See http://www.wordiq.com/definition/Domain_name_registry for more information on domain name registries.

- allow established connections from the server port 1803 tcp to servers used for remote administration
- allow new and established connections from anywhere to the server port 53 tcp and udp
- allow established connections from the server port 53 tcp and udp to anywhere
- allow new and established connections from the server to tcp port 80 on security.debian.org during a security update as shown in 4.3 - 194.109.137.218 at the time of this writing
- allow established connections from tcp port 80 on security.debian.org to the server during a security update
- allow new and established connections from the server to tcp port 25 on the corporate mail relay
- allow established connections from tcp port 25 on the corporate mail relay to the server
- allow new and established connections frm the server to udp port 53 on the corporate DNS servers
- allow established connections from udp port 53 on the corporate DNS servers to the server
- allow new and established connections from the server to udp port 123 on the corporate ntp server
- allow established connections from udp port 123 on the corporate ntp server to the server
- allow new and established connections from the server to tcp port 514 on the corporate syslog server
- allow established connections from tcp port 514 on the corporate syslog server to the server
- allow new and established connections from the server to tcp port 3306 on the master MYSQL backend server
- allow established connections from tcp port 3306 on the master MYSQL backend server to the server
- drop anything else

The server will not require access to the root servers or other public DNS server, as the server should only answer queries against zones hosted on the server.

2.3 Hardware Requirements

I will use a HP Net server LP 2000R. The technical specifications are:

- Single Pentium III 1000MHz CPU
- Two 18Gb 15000 rpm SCSI drives configured in a raid 1 array
- Netraid 1m raid controller
- 512Mb RAM

2.4 Risk Analysis

The server will be used to host customer domains for the DNS hosting provider GIAC Inc. If the server is compromised, the zone data might be changed. Changing data on live DNS servers can lead to e-mail hijacking if the mx records are changed and e-mails are relayed through other servers, where the attacker can read them. If A- or CNAME-records are changed, the attacker might be able to deface web sites[3] or perform a Man-in-the middle attack if one of the customers unknowingly tries to log in on a server with a changed A of CNAME record.
As the server is located in the DMZ it is also a risk that the server might be used to compromise other servers in the DMZ if it is compromised. I have experienced that attackers often use this way of bypasing firewall rules to attack service on the other servers in the DMZ that are not exposed to the internet.

2.4.1 Risk mitigation plan

It is therefore critical to mitigate the opportunities the attacker has to change the zone data. To obtain this I will lock the PowerDNS server in a chroot jail. The server will only be able to read the data from the MySQL server, and only through a UNIX socket in the jail. The MySQL server will not be listening on any network sockets, and the MySQL data files will be located outside the PowerDNS jail to keep them safe.
It is also critical to minimize the remote administration access to the server. To limit the access via openssh I will disable root login via ssh. I will also disable password login via ssh. Although the server will be protected from the internet by a firewall, there is also a risk that one of the other servers in the DMZ will be compromized and used in an attack against this server. Therefore I will install a local firewall on the server in order to restrict the access to the openssh service on the server to certain ip-adresses on the internal networks. The

[3]The A or CNAME records pointing to the web server could be changed to point to a web server controlled by the attacker. The customer web server will not be compromised, but the content of the web page will appear different.

local firewall will also prevent all other connections to all addresses other than a specified set of services on a specified selection of servers. This will make the server less vulnerable to attacks initiated from other servers in the DMZ. Once the server is operational it will not be possible to download any software to the server. The firewall will also drop downloads from servers in the DMZ to the server [4]. The local firewall will have the following rules:

- drop invalid packets before they are processed further
- drop packets from the network with a loopback (127.0.0.1) source address
- allow new and established connections from servers used for remote administration to the server port 1803 tcp
- allow established connections from the server port 1803 tcp to servers used for remote administration
- allow new and established connections from anywhere to the server port 53 tcp and udp
- allow established connections from the server port 53 tcp and udp to anywhere
- allow new and established connections from the server to tcp port 80 on security.debian.org during a security update as shown in 4.3 - 194.109.137.218 at the time of this writing
- allow established connections from tcp port 80 on security.debian.org to the server during a security update
- allow new and established connections from the server to tcp port 25 on the corporate mail relay
- allow established connections from tcp port 25 on the corporate mail relay to the server
- allow new and established connections frm the server to udp port 53 on the corporate DNS servers
- allow established connections from udp port 53 on the corporate DNS servers to the server
- allow new and established connections from the server to udp port 123 on the corporate ntp server
- allow established connections from udp port 123 on the corporate ntp server to the server
- allow new and established connections from the server to tcp port 514 on the corporate syslog server

[4] It will only be possible to download security updates after adding a rule to the local firewall. Otherwise an attacker would be able to download, unpack and use the contents of any packet on security.debian.org.

- allow established connections from tcp port 514 on the corporate syslog server to the server
- allow new and established connections from the server to tcp port 3306 on the master MYSQL backend server
- allow established connections from tcp port 3306 on the master MYSQL backend server to the server
- drop anything else

These strict local firewall rules will also make it more difficult to use the server as an attack point in the DMZ if it is compromised.
If the server is compromised it is critical to discover the compromise as soon as possible. The attacker will most likely try to hide his or her tracks with a regular file based rootkit or an LKM rootkit. I will install the aide intrusion detection tool on the server to discover changes in the installed binaries or the exsistence of new binaries on the server. This will not prevent the attacker from using a regular file based rootkit. But it will make it harder for him or her to hide with a regular file based rootkit. To avoid the installation of LKM rootkit and insertion of malicios code in the kernel. I have chosen to disable the ability to load kernel modules. I will also use the memory protection features developed by the grsecurity project. This will prevent the four know ways [5] to insert malicios code in the Linux kernel.

3 Steps to Install and Harden the Server

3.1 Physical security

The server is located in the server room in the basement of the Giac Inc. corporate head quarters. The alarm system in the head quarters only allows system administrators to access the server room. They need their digital token and a four digit code to turn off the alarm and enter the server room. As Management in Giac Inc. is satisfied with the physical security in the server room I will not take steps to passwords protect the BIOS or make the server impossible to boot from a CD. Such actions might also make a disater recovery of the server extremly hard if the BIOS password is lost.

3.2 Download The Installation Media

The first thing I do when installing Debian GNU/Linux is to download the installation media. I download the first Debian official cd from a Debian mirror[3]. As the writing on the website says, I check if the version I am downloading is the latest release[5]. At the time of this writing the latest stable release is version 3.0 rev2. After I have downloaded the first ISO image, I verify the MD5 sum of the image. To verify the integrity of the downloaded ISO image, I

[5]Write to /dev/kmem, /dev/mem, and /dev/port and insert modules

download the MD5SUMS text file found on the same mirror as the ISO files. To verify the files I use the md5sum program on a Debian GNU/Linux 3.0 development server:

```
sos@foobar: $ md5sum -cv MD5SUMS
debian-30r2-i386-binary-1.iso md5sum:  can't open debian-30r2-i386-binary-1.iso
debian-30r2-i386-binary-1_NONUS.iso OK
debian-30r2-i386-binary-2.iso md5sum:  can't open debian-30r2-i386-binary-2.iso
debian-30r2-i386-binary-3.iso md5sum:  can't open debian-30r2-i386-binary-3.iso
debian-30r2-i386-binary-4.iso md5sum:  can't open debian-30r2-i386-binary-4.iso
debian-30r2-i386-binary-5.iso md5sum:  can't open debian-30r2-i386-binary-5.iso
debian-30r2-i386-binary-6.iso md5sum:  can't open debian-30r2-i386-binary-6.iso
debian-30r2-i386-binary-7.iso md5sum:  can't open debian-30r2-i386-binary-7.iso
debian-30r2-i386-binary-7.is md5sum:  can't open debian-30r2-i386-binary-7.is
```

The check tells me that the file debian-30r2-i386-binary-1_NONUS.iso verifies against the md5sum in the MD5SUMS file. The other files are not needed and the rest of the output can be ignored.
As an attacker might have replaced both the ISO image and the file containing the md5sums of the images, I also need to verify the gpg signature on the file containing the md5sums. I need the Debian GNU/Linux keyring[6] to be able to verify the signature. I have no way of knowing that the Debian keyring has not also been compromised, but I downloaded the ISO images and the MD5SUM file from another server than the debian keyring. So an attacker should have compromised both the mirror and keyring.debian.org to be able to change the content of the downloaded images without my knowledge. The gpg signatures can be verified with this command.

```
sos@foobar: $ gpg --keyserver keyring.debian.org --verify MD5SUMS
gpg:  Signature made Fri Dec 5 22:32:11 2003 CET using DSA key ID DD9B9910
gpg:  requesting key DD9B9910 from keyring.debian.org ...
gpg:  key DD9B9910:  public key imported
gpg:  Total number processed:  1
gpg:  imported:  1
gpg:  Good signature from "Philip Hands <phil@hands.com>"
gpg:  sig DD9B9910.74:  duplicated certificate - deleted
Could not find a valid trust path to the key.  Let's see whether we
can assign some missing owner trust values.

No path leading to one of our keys found.

gpg:  WARNING: This key is not certified with a trusted signature!
gpg:  There is no indication that the signature belongs to the owner.
gpg:  Fingerprint:  8DF8 CE53 24F3 177C 2417 6C65 6203 8A4B DD9B 9910
```

This tells me that I have requested the key with keyID DD9B9910 from the keyserver keyring.debian.org. And that the file MD5SUMS contains a valid signature from that key. It also tells me that the key is not trusted, as I have no trust path to the key with keyID

DD9B9910. The Debian developers probably don't want to spend time to meet me in real life and exchange GPG keys with me so I will have to live with that. These checks might be circumvented if an attacker has been able to replace the key on the Debian keyserver and the ISO image and MD5SUMS file on the mirror I downloaded the ISO file from. I trust that these incidents will not happen at the same time and proceed with my installation. I can now burn the ISO image on a CD and get on.

3.3 Install debian GNU/Linux 3.0

During the installation all network cables will be unplugged from the server. The server will be connected to the network later on as described in 3.4. I have configured the server to boot from the CD drive. I type *vanilla* and press enter at the welcome screen and boot prompt to boot the 2.2 Linux kernel with support for the netraid controller[6] in the server. The Installation now presents the Release notes and I press enter to continue. I have now reached the Installation Main Menu. The first thing I do is to press enter to *configure the keyboard*. I select *qwerty/dk-latin1 : Danish* and press enter. I now select *partition the hard disk* from the main menu and press enter. I then choose the `/dev/sda` disk from the disk selection screen, press enter, and press enter to continue at the LILO Limitations screen. The disk partitioning utility now asks: `"Do you wish to start with a zero table [y/N]?"`. I type *y* to start with a zero partition table. This will assure that the hard drive in the server is cleaned of any previously installed software. I use the disk partitioning utility to create the partitions shown in 7.1. I then change the type of the swap partition to linux/swap, write the partition table to disk and quit the disk partitioning utility.

Now I select "Initialize and Activate a Swap Partition" to create the swap space and activate the swap partition. I use the "Initialize a Linux Partition" item on the main menu to create the file systems and mount the partitions, until all the partitions are initialized and mounted. I test all of the partitions for bad blocks and mount them according to the mount points defined in 7.1. I have gathered all my considerations concerning the partitioning scheme for the server in 3.3.1. The Debian installer will use the ext2 file system for all partitions when using the vanilla 2.2 Linux kernel. I will later convert the ext2 filesystem to ext3. This will add journaling to the filesystem and make the system less vulnerable to power outages. After the server is installed I will install the fstab file show in 7.1 and reboot the server. This will enable the diferent mount options I have chosen in 3.3.1.

The next thing I do is to install the kernel and device driver modules. I install them from the CDrom. I then choose to configure the device driver modules. The only module I add is the one for the network interface card. I press continue at the note about loaded drivers and choose the "net" item from the next menu. From this menu i now select to insert the eepro100 modules in the kernel. I do not specify any additional command line parameters. I then select exit two times to return to the installation main menu. I am now able to configure the network. I type the hostname ns1 at the "Choose the Hostname" box and press "ok" to continue. As the server is not yet connected to a network, I can't use DHCP to configure the network interface.

[6]The HP netraid 1m controller uses the megaraid driver in the Linux kernel

I select "no" at the "Automatic Network Configuration" box. I choose an IP adress that is not used on the internal network in GIAC Inc. and a netmask, default gateway, and two dns servers, that match the details of the internal network. I also configure the domain name of the system to giac.net. The next item I select from the "installation main menu" is "install the base system".´ This will install all the basic packages in Debian GNU/Linux on the server. I now select "Make System Bootable", install lilo on the master boot record of /dev/sda. I get a warning about the lack of security in the default lilo configuration. I ignore this warning as I will install a more secure lilo configuration before I deploy the server. The first part of the installation is now finished. I select "Reboot the System" and wait for the server to begin the post install configuration.

3.3.1 My considerations on the partitioning scheme

I have chosen a partitioning scheme according to the Debian security manual[7] and the LASG[8]. During the installation I will only create the partitions. The mount options I use on the server are not enforced during the installation. They will be applied later on in 3.18, by editing the `/etc/fstab` file and remounting all partitions.
I would like to be able to place a restrictive set of permissions[7] on the `/` partition, to retain a secure set of default permissions. That will require that `/dev`, `/lib`, `/sbin` and `/bin` are located on separate partitions. It makes the boot procedure a bit tricky, as they contain the device files used by the kernel before partitions other than `/` are mounted and some of the most vital binaries[8] on the system. It is possible to create a separate partition with an exact copy of the data from the i.e. `/bin` directory. This partition can the be mounted in `/bin` at boot time[9]. When the server has finished the boot procedure, the content of `/bin` will be located on our separate partition with the mount options specified in `/etc/fstab`[10]. It is possible to use separate partitions for `/dev`, `/bin`, `/sbin` and `/lib` through this procedure, but I have chosen not to do this for the following reasons:. It becomes tricky when I need to update the software on the server. The updates need to be applied to the files on the separate partitions and the files in the directories located on `/`. Practically this means that I have to unmount the `/dev`, `/bin`, `/sbin` and `/lib` partitions, install the updates to the original directories on `/`, mount the partitions again with rw option and reinstall the updates. The problem is that the partition unmounting may require a server reboot as many processes are using the files in `/dev`, `/bin`, `/sbin` and `/lib`. This makes the update process too cumbersome. The update process is one of the most central procedures in keeping a server secure. The reboot might delay the update installation because Management does not want unscheduled downtime. I consider this to be a severe reason to drop separate partitions for `/dev`, `/bin`, `/sbin` and `/lib` and end up with a less secure set of default permissions on the `/` partition. That will

[7]The noexec option to deny direct execution of files on this partition. The content can still be executed through the use of ld-linux.so.2. The nodev to deny the pressence of device files on this partition. The nosuid option to disable the effect of the suid bit. The ro to mount the filesystem read-only.

[8]Like `init`

[9]The usual way through `/etc/fstab`

[10]Should be mounted with the options nodev,ro

leave me with the mount option `erros=remount-ro` on the / partition.

Directories that contain variable data are allocated a separate partition. These directories include:

- /var
- /home

I have only created a small partition for `/home` as there is only one user on this system, which is only used to log in via ssh. These partitions are mounted with the noexec, nosuid, and nodev options. The `/home` partition is also mounted with the ro option I have allocated a separate partition to all directories that are world writeable. This will prevent un-priviledged users from filling the disk. They are mounted with the nodev, noexec and nosuid options to prevent malicious users from creating device files and accessing devices through them and to prevent the root user and other users from accidentially executing content written to these directories by malicious users. These directories include: /var/lock, /var/tmp and /tmp. To determine wich directories that are world writeable on an allready configured server I use a simple one line bash script:

```
#for dir in `ls / | sed "/dev/d" | sed "/proc/d"`; do find /$dir -follow
-perm -0002 -type d; done
```

It will find all directories in the / that are not in `/dev` or `/proc` and have the all write permission bit set.

Directories that contain static data are allocated a separate partition. These partitions contain many binaries that are a vital part of the system. All of these partitions are mounted with the nodev and ro option so I don't have to rely solely on filesystem security, to prevent user from adding to or changing the content of these partitions. Although these directories include the `/lib`, `/sbin` and `/bin`, these will not be on separate partitions for the reasons mentioned above. The directories I have enforced these restrictions on are:

- `/usr`
- `/usr/local`
- `/opt`

The server will not have any third party software installed that can not be managed via the Debian package system. However I will create a separate partition for `/opt`, just to be able to control the mount options on this directory. I will not create a separate partition for `/usr/local` as the mount options on this directory are enforced by the mount options on the `/usr` partition. I will use the following sizes for the partitions:

- /boot - 100Mb
- / - 500Mb

- /usr - 1000Mb
- /var - 7000Mb
- /home - 500Mb
- /tmp - 1000Mb
- /var/lock - 300Mb
- /var/tmp - 700Mb
- /opt - 100Mb
- SWAP - 1024Mb

3.4 Network connection

The server can and must now be connected to a network protected by a firewall. The server should be the only host on the network it is connected to. Furthermore the firewall should be configured to allow all traffic comming from the server to the internet but allow only established or related connections coming from the internet.

3.5 First Boot Configuration

I now remove the installation media from the cdrom drive and disable boot from removeable media in the BIOS. The Debian installer now starts the Debian system configuration dialog. I press enter at the welcome screen. I then select ”No - the hardware clock is not set to GMT”. To be able to compare log entries all the servers in GIAC Inc. have set the time zone to UTC. At the next configuration menu the system configuration dialog asks in which area I live in. I select ”none of the above” and ”UTC” at the following menu. At the password setup dialog I enable the use of MD5 passwords. This will make me able to use passwords that are longer than 8 characters which will give me a chance to enhance the password security. I also enable shadow passwords in the next dialog. This will make the system less vulnerable to password cracking attacks. An attacker will now need root access to the system before he or she can start a password cracker on the encrypted passwords in the `/etc/shadow` file. At the next configuration dailog I am asked to set the root password. I press enter and type the root password and type it again at the verification dialog. You should choose a root password of a certain password quality. A good standard for choosing root passwords and other passwords can be found at the SANS resource site [20]. I have created all passwords and passphrases for private keys used in the configuration of the server according to this policy. The system configuration now asks if I want to add a normal user account. I select ”no” as I will do this later on. I then press enter at the next configuration dialog to remove the pcmcia package. I also press enter to avoid using a ppp connection to install my system. The system configuration dialog now starts the apt configuration. I select to use http to access the debian archive. If the server is not in the US I can select ”yes - use non-US software”. I choose not to use non-free or contrib software as these archives are not supported by the debian security team[10]. As the server is located in Denmark I choose to use a debian archive located in Denmark and select the ftp.dk.debian.org archive. I configure apt not to use a proxy server and wait for apt to download the package lists from the debian archive. I then select ”No - Do not use another apt source” and ”Yes - Use security updates from security.debian.org”. Debian GNU/Linux has at least two other frontends to the package manager dpkg. I select not to use any of the two frontends tasksel and dselect proposed by the configuration dialog. The installer now wants to remove the pcmcia package and install the latest security updates from security.debian.org. I press enter to continue that operation and press enter to erase previously downloaded debian packages. I press enter to continue and press enter to configure exim[11]. The installer gives me five different configuration options and I choose number 5 - No configuration. I will later install the postfix MTA. This will also remove exim. The system configuration is now finished and I press enter to end the dialog.

3.6 Software installation

The applications and servers needed for operation, security and convenience are now installed. I only install the packages absolutely needed to operate, administrate and secure the server. These include the packages from Debian GNU/Linux:

- ntpdate - to keep the server clock synchronized with other servers in the network. This is a must if you want to be able to compare log entries or time stamps on files in a security audit or incident handling situation.
- mysql-server - as the backend for powerdns.
- ssh - to enable secure remote administration of the server.
- postfix - to send emails. Every UNIX or Linux system needs an MTA to send system notifications to the administrator. I don’t say that postfix is more or less secure than exim [12]. But it is more secure to install an MTA you are more familiar with, as you have better control of the MTA configuration. The company GIAC Inc. uses postfix on all it’s UNIX and Linux servers, so I will install postfix on this server.
- aide - to provide local integrity check based intrusion detection.
- syslog-ng - to provide a syslog daemon with a richer set of features than the default sysklogd. In particular the ability to send syslog messages over a tcp[13] connection to a central server. Much valueable information like usernames, cronjob output, etc. is sent over the network when you use a central syslog server. This ability can be used together with the stunnel[14] package to encrypt the traffic sent to the central server.

[11] The default MTA provided with Debian GNU/Linux
[12] The default MTA provided with Debian GNU/Linux
[13] Not the default udp
[14] Can encrypt tcp connection but not default syslog udp traffic

- stunnel - for reasons mentioned above.
- pdns-static - the powerdns server.

As PowerDNS is not an integrated part of Debian GNU/Linux yet, I will have to download the most recent release from the PowerDNS website[1]. I download the PowerDNS debian package from the PowerDNS download site[2]. The PowerDNS debian packages is not downloaded from a mirror, so I don't have to make MD5 integrity checks og verify gpg signatures for the package. You should always download the PowerDNS packages from the PowerDNS download site. The PowerDNS team do not distribute MD5 sums or gpg signatures for their packages. If you download the package from a mirror, you have no way to verify the integrity of the package. I place the downloaded package on the company apt repository, so I can install it on other servers in the organisation if I want to. When I place the package on the apt repository i need to run `dpkg-scanpackages` to rebuild the Packages.gz file [15]. This file contains information about the package on the repository including the MD5 checksum of the packages. I use apt to install all software on the server. All softare installed by apt will have the MD5 checksum validated by apt before they are installed. The package dependencies for the software mentioned above will also be installed by apt. Before I install the software, I edit the `/etc/apt/sources.list` file to include the company apt source. I use a small script to install all the software and dependencies. The script is shown in 7.7. Once the server is ready fo production the script `installpackage.sh` shown in 7.4 should be used to install new packages on the server. This script will open the firewall from the server to the apt repositories and install the package with apt. During the installation `dpkg` will ask me to configure some things. I will answer all the question with the default answer suggested by `dpkg`. Execpt on these questions

- Configure ntp server. I type the corporate ntp server ntp.giac.net
- Should MySQL start on boot? I answer yes
- Postfix. General type of configuration. I choose Internet with smarthost
- Postfix. SMTP relay host? I type the corporate smtp relay server smtp.giac.net
- Postfix. Where should mail for root go? I type administrator@giac.net

The configuration of the packages is hardened later on.

3.7 Harden The Base OS

The first thing I do is to set all shells of system users to `/bin/false`. This will render the accounts useless for logins locally or via ssh. In Debian GNU/Linux all passwords except root's are disabled by default. So I don't need to take care of that. I create a group called admins. I now create the user sos, a normal user account. The sos user is also added

[15] I will not describe this process as it belongs to the setup of the apt repository.

to the admins group. This account will be used for remote and local administration when the server is operational. The server is configured to limit su to the admins group through `/etc/pam.d/su`. This will effectively disable the su command for non-root users who are not members of the admins group. Some people like to chown and chmod the su binary but I don't like this way of protecting the use of su as an attacker might just use his or her own uploaded version of su. Extra logging of failed logins for unknown users and successful logins are enabled in `/etc/login.defs`. The minimum password age is set to 7 days and the maximum password age is set to 180 days in this file too. The minimum passsword age needs to be set to prevent someone from changing the password two or more times in a row to end up with the same password as before the change. I enable the use of `/etc/security/access.conf` in `/etc/pam.d/login`. I can then disable logins for all users except the members of the admins group in `/etc/security/access.conf`. I comment out all services in the /etc/inetd.conf. The services are not needed. I also disable the inetd server as it not going to be used on this system. I remove the startup links from the `/etc/rc*.d` directories. I then re-add stop links for inetd. These links are needed because an update of the netkit-inetd[16] packages will reinstall them if they are not in place. That would start up the unwanted service at next boot. I then stop the running inetd with the command:

```
# /etc/init.d/inetd stop
```

I modify the file `/etc/issue` as shown in 7.12 to create a legal warning that is shown on the console before a user tries to log in. I have scripted all these initial configurations in 7.5. The script also needs the patch 7.6 saved in `/etc.patch` to do its work.

3.8 Aide configuration

Aide is a small intrusion detection tool that builds a database of file permissions, sizes, checksums and other usefull details about system files. The default configuration is usable but I also like to have the home directories, the crontabs, the man pages, the doc directories, the header files and of course the chroot environment for pdns checked. This will keep track of any changes made to these files and send an email to root very day with the list of changes. The full configuration file can be found in 7.8. The aide database is initialized with the command:

```
# aide --init
```

and moved to the correct location with the command:

```
# mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
```

The aide database should be updated every time a package has been installed or updated on the server. In 4.3 I will show how this can be automated.

3.9 Stunnel configuration

Stunnel is used on the server to provide an encrypted and authenticated secure tunnel to the corporate syslog server. I configure stunnel to listen on localhost tcp port 514 and send all

[16]The package containing inetd

requests it gets on this port to the syslog server tcp port 514. A stunnel service on that server will then decrypt and redirect the traffic to locahost tcp port 514, where the syslog-ng server will be listening. The corporate syslog server is configured only to accept stunnel sessions from a client autheticated with a client certificate. The private key and certificate can be create with the command:

```
# openssl req -new -days 365 -x509 -outform pem -out syslog.pem
```

this will create a private key in `private.pem` and a certificate in `syslog.pem`. I have chosen not to use a paaphrase for the private key. The stunnel instance will not start on boot by itself if the private key is encrypted and protected with a passphrase. That would have severe consequences as the server would not be able to replicate the MySQL database or send logs to the central syslog server. To use them with stunnel they need to be placed in the same file. This can be done like this:

```
# cat private.pem >> syslog.pem
```

The stunnel setup consists of a startup script called `stunnels.h` shown in 7.9 and the pem file containing the private key and the certificate. I put the pem file[17] in `/etc/syslog-ng/` and make sure it has the permissions 0600 and root.root as the owner and group. The syslog server needs this private key and certificate to authenticate the clients [18]. The configuration mentioned in this paragraph is also used to SSL enable the MySQL replication. I have just used a different tcp port - port 3306 and a different private key and certificate. The private key and certificate for the MySQL stunnel instance is saved in `/etc/mysql/mysql.pem` on the server. This instance is also started in the `stunnel.sh` script. I create a separate user to run the stunnel instances like this:

```
# groupadd stunnel
# useradd -s /bin/false -g stunnel stunnel
```

To start the stunnel instances on boot I save the startup script `stunnel.sh` in the `/etc/init.d/` directory and use the command:

```
# update-rc.d stunnel.sh defaults
```

3.10 Syslog-ng configuration

I have chosen to log all messages to both local files and central syslog server. This can save valuable information on in the local files if the network connection or syslog server goes down. It can also save valuable information on the syslog server if an attacker deletes log entries in the local files to hide his/her traces. To be able to utilize the stunnel setup I have to use a syslog daemon, that speaks tcp. The default syslog daemon sysklogd on Debian GNU/Linux doesn't do that, so I have chosen the syslog-ng daemon for the job. The configuration I use for syslog-ng is shown in 7.10. Basically I removed filters, destinations and log actions that was not needed on this server. I also added `tcp("localhost" port(514))` to every destination except for the console devices. This will make syslog-ng able to log to the corporate syslog

[17] The file is called syslog.pem

[18] I will not describe how to install these files as it is basically a process that belongs to the setup of the syslog server

server through the stunnel instance listening on localhost tcp port 514. The permissions on the local log files where also modified to make then readable only by root.

To monitor the log files I run logcheck on the central syslog server. I will not go into details on the logcheck setup as it is a part of the setup on the syslog server.

3.11 Configure time synchronization

It is really important to keep the time on your servers synchronized if you want to be able to compare log entries from different servers in the network. I use a small shell script called timesync.sh shown in 7.14 to synchronize the server system time and hardware clock to the time on the corporate ntp server. I drop the script in `/usr/local/bin` and create a symlink to `/etc/cron.daily/timesync.sh` to make it run once a day.

3.12 Harden the openssh server

The ssh daemon is configured to disallow root logins. Password authentication for other users are disabled. The daemon is also configured to use version 2 of the ssh protocol only. The default port is changed to port 1803. This may seem like an attempt to enhance security by obscurity. It is not, as it will now require a port scan to detect the listening sshd. The port scans can be detected by ids software like snort or portsentry. The change to a non-default port number will then give me the possibility to detect a starting attack on the server. Note that I will not install portsentry or other portscan detection on this server. The portscan detection should be installed on a server that is not a part of the production environment. Then the data on the production servers will not be jeopardized in the case that the port scan detection is compromized. Privilege separation is turned on to run most of the daemon as an unprivileged user. Only users in the admins group are allowed to login via ssh. As this server is not a bastion host of any kind I disable tcp forwardning through ssh. A Banner file is created in `/etc/issue.net` and sshd is configured to show the banner before a user tries to log in. The banner is shown in 7.11. I create a private and public DSA ssh key with the command:

```
# ssh-keygen -t dsa -b 4096
```

This key is used to autheticate the sos user when it logs in via ssh. The private key is moved to the workstation or server used for remote administration of this server. The public key is move to `/home/sos/.ssh/authorized_keys` [19]. The user sos is then made the owner of the key by executing:

```
# chown -R sos.admins /home/sos/.ssh
```

The full sshd_config file is shown in 7.13.

[19]Note that /home/sos/.ssh has to be created first.

3.13 Secure the postfix MTA

I have installed the postfix MTA. The MTA will only be used to send mail from the server. For this reason I don't need the smtp daemon. I disable the daemon by commenting out the line starting with `smtp inet` in the `/etc/postfix/master.cfg` file. The postfix MTA now defaults to make the mail drop directory group writeable instead of the heavily discussed[9] world writeable mail drop directory. In other words I will not have to be worried about this vulnerability. I have already enabled the use of a relay host in the `/etc/postfix/main.cf` file during the initial configuration of the postfix package. The rest of the configuration is just the default coonfiguration installed by the debian package. I end up with an MTA that will not allow relaying, doesn't listen on a network socket and has no world writeable directories.

3.14 Harden the kernel

3.14.1 Where to compile the kernel

It is usually a good idea to keep your production servers clean of compilers and development tools[20]. These tools will just give an attacker better opportunities to compile and install exploits and rootkits. As a consequense I always compile software including the Linux kernel on a development server on the internal network. I can then copy the kernel and the system map file to the production server. To get most control of what goes in the kernel for the server I download the latest 2.4 version of the kernel source from www.kernel.org web site. The 2.6 kernel has a number of improvements over the 2.4 version. It is still not usable on debian production systems, as it requires many packages from the unstable debian release. These packages are not supported by the debian security team[10] and should not be used on production systems. I also need some security enhancements that are not in the Linux kernel and need to be applied from source code patches.

To compile the Linux kernel from kernel.org on Debian GNU/Linux I need the following packages: tar, bzip2, make, gcc, libc6-dev, bin-utils, libncurses5-dev and patch. The packages will also need their dependencies installed of course.

3.14.2 Features I want to include in the kernel

Most of the vulnerabilities discovered in Linux or Unix daemons are buffer overflows or format string vulnerabilities. Stack smashing protection will protect the server from many attempts to compromise the daemons running on it. There are many implementations that provide stack smashing protection for GNU/Linux based systems. Most of the implementations works by avoiding overwriting of the critical parts of the stack such as the frame pointer and the return address. If a program tries to overwrite these parts of the stack it will exit with a page fault if it has been compiled with stack smashing protection. This will effectively prevent exploitation of buffer overflows and format string vulnerabilities. The problem in this implementation is that it needs to be compiled into every program it should protect from exploitation. But it

[20]Programs like make

can be applied to the most critical parts of the operation system. It is difficult to protect an entire system with this implementation unless it has been implemented by the creators of the GNU/Linux distribution. Another way to protect a server from buffer overflows and format string vulnerabilities is by going for a kernel level implementation. This will protect the entire operating system and all I have to do is to patch and recompile the kernel.
These features are not in the kernel so I need to apply a patch to achieve my goal. I have chosen the stack smashing protection implementation from Grsecurity[4]. The kernel patch from Grsecurity will also give the server several other security enhancements - mainly chroot jail restrictions and additional randomness. Additional randomness provided by Grsecurity will randomize the location of executable images, library images, several heaps, user space stack and kernel space stack in the memory. It will also randomize the initial sequence numbers of tcp connections, tcp source ports, ip ids and process ids. These randomization features will make it harder for an attacker to locate executeable content placed in the memory by the attacker. It will also make the server harder to fingerprint as much of the tcp/ip data used by finger print software will be randomized by the Grsecurity patch. The Grsecurity patch also provides a role-based access controll system. Role-based access controll is good for servers that takes care of the same tasks for a long time without much modification and without too many configuration changes. I will not use the role-based access control features as it will make it far too complicated to make configuration changes or add new service to the server.
I will disable loadable module support in the kernel. This will make it impossible for an attacker to install an LKM rootkit that I would not be able to detect with aide.

3.14.3 How I build the kernel

To accomplish all this I download the latest 2.4 kernel[21] from www.kernel.org. I place it is `/usr/src` on my development server. I then unbzip2 and untar the kernel source:

```
foobar:/usr/src# tar jxf linux-2.4.26.tar.bz2
```

I also download the grsecurity patch for the 2.4.26 kernel from grsecurity.org. I place it in `/usr/src` on my development server and apply it to the kernel source:

```
foobar:/usr/src# patch -p0 <grsecurity-2.0-2.4.26.patch
```

This command will output the patched files. I now change directory to `/usr/src/linux-2.4.26` and execute make menuconfig to configure the kernel:

```
foobar:/usr/src# cd linux-2.4.26 && make menuconfig
```

I keep the default configuration except that I change the items on the following list:

- Loadable module support
  - Enable loadable module support - removed
- General setup
  - ISA bus support - removed

[21]Version 2.4.26 at the time of this writing

  - Support for hot-pluggable devices - removed
  - Kernel support for a.out binaries - removed
  - Kernel support for MISC binaries - removed

- Plug and Play configuration
  - ISA Plug and Play support - removed
- Networking options
  - Network packet filtering - added
  - IP: TCP syncookie support - added
  - IP: Netfilter Configuration
    * Connection tracking - added
    * FTP protocol support - added
    * IP tables support - added
    * limit match support - added
    * Packet type match support - added
    * Multiple port match support - added
    * Connection state match support - added
    * Connection tracking match support - added
    * Packet filtering - added
    * LOG target support - added
- SCSI support
  - SCSI low-level drivers
    * AMI MegaRAID2 support - added
- Network device support
  - Dummy net driver support - removed
- Character devices
  - Mice
    * Bus Mouse Support - removed
    * Mouse Support - removed
  - Direct Rendering Manager
    * Direct Rendering Manager - removed
- File systems

  - Kernel automounter version 4 support - removed
  - Ext3 journalling file system support - added
  - Network File Systems
    * NFS file system support - removed
    * NFS server support - removed

- Sound
  - Sound card support - removed
- USB support
  - Support for USB - removed
- Library routines
  - CRC32 functions - removed
- Grsecurity
  - Grsecurity - added
  - (Customized) Security level
  - Address Space Protection
    * Deny writing to `/dev/kmem`, `/dev/mem`, and `/dev/port` - added
    * Disable privileged I/O - added
    * Remove addresses from `/proc/pid/[maps|stat]` - added
    * Hide kernel symbols - added
  - Role Based Access Control Options
    * Hide kernel processes - added
  - Filesystem Protections
    * Proc restrictions - added
    * Restrict to user only - added
    * Additional restrictions - added
    * Linking restrictions - added
    * FIFO restrictions - added
    * Chroot jail restrictions - added
    * Deny mounts - added
    * Deny double-chroots - added
    * Deny pivot_root in chroot - added
    * Enforce chdir("/") on all chroots - added

    - Deny (f)chmod +s - added
    - Deny fchdir out of chroot - added
    - Deny mknod - added
    - Deny shmat() out of chroot - added
    - Protect outside processes - added
    - Restrict priority changes - added
    - Deny sysctl writes in chroot - added
    - Capability restrictions within chroot - added
  - Kernel Auditing
    - `/proc/<pid>/ipaddr` support - added
  - Executable Protections
    - Dmesg(8) restriction - added
    - Randomized PIDs
    - Trusted path execution - added
    - Partially restrict non-root users - added
    - GID for untrusted users: 1005
  - Network Protections
    - Larger entropy pools - added
    - Truly random TCP ISN selection
    - Randomized IP IDs - added
    - Randomized TCP source ports - added

The items are mostly changed to remove support for hardware or features that are not used. Support for the kernel based firewall netfilter is also added. Most of the Grsecurity options are enabled. Most of the Grsecurity kernel audit options are disabled as they create too much logging. The option Deny access to abstract AF_UNIX sockets out of chroot is not enabled as the powerdns server needs to access the mysql database through a UNIX socket bound outside the chroot jail that powerdns is running in.
The kernel is now build with the command:

```
# make dep clean bzImage
```

The files `System.map` and `arch/i386/boot/bzImage` are now copied from the development server to our production server. I place them in `/boot/System.map` and `/boot/vmlinuz-2.4.26` on the production server.

3.14.4 Configure and secure lilo

Lilo is now configured to boot the new kernel and the old kernel in backup. The option `password=""` is set and both kernels images are given the restricted option in `lilo.conf`. Setting an empty `password` option will make lilo aks for a password when it is installed. A local attacker now has to provide a password to pass boot arguments to the kernel. The entire `lilo.conf` is shown in 7.15. The boot loader is now installed on the boot device with the command:

```
# /sbin/lilo
```

3.15 Host based firewall

I will now show how to configure a host based firewall for the server. The firewall will implement the rules described in 2.4.1. I have used the linux kernel firewall netfilter to implement the firewall. The configuration is a simple shell script shown in 7.16. The script is placed in `/etc/init.d/firewall.sh`. I then use the command `# ln -s /etc/init.d/firewall.sh /etc/rcS.d/S38firewall.sh` to make the firewall start at boot. According to the files `/etc/rcS.d/README` the firewall will now be started at boot time before the network is initiated.

3.16 Hardening MySQL

The MySQL server provided with Debian GNU/Linux 3.0 is already installed. The default behaviour is that the server does not listen on any network ports. I will keep this configuration as I do not need access to the server from the network. The only way to access the server will now be trough the UNIX socket on the server. This socket should be reachable by the powerdns server, so I configure MySQL to place the socket in the directory structure I will use for the chroot jail that powerdns will run in. The socket is configured to be in `/var/chroot/pdns/run/mysql.sock`. This directory does not exist so I create it with the command:

```
# mkdir -p /var/chroot/pdns/run
```

I change the owner and group on these directories to mysql.root on `/var/chroot/pdns/run/`. The mysql user needs the write access to the directory to create the UNIX socket. The rest of the mysql installation will be placed outside the jail. If an attacker will have the luck to exploit the powerdns server, it will not give him access to change the database files, only to read from them through the MySQL socket. The Innodb table type is enabled as powerdns runs much better on these tables. The MySQL server also need some information on where to get the pdns database from. I have configured the slave replication in the lines between

```
#start slave settings
```

and

```
#start slave settings
```

in the mysql configuration file `/etc/mysql/my.cnf`. The password for the replicate user and the server id should be replaced in the example file provided in 7.17. The master-host parameter in the configuration file is set to 127.0.0.1. The slave server will then connect to the stunnel instance listening on port 3306 on 127.0.0.1 on the server. The connection will then pass through the SSL tunnel to a stunnel instance on the master server. The data passing through the connection will then be secured against sniffer attacks. To start the replication the slave server needs a copy of the database from the master [22]. To make this copy I open the mysql client on the master server and issue the command:

```
mysql> FLUSH TABLES WITH READ LOCK;
```

This will lock the tables so no writing will occur while I make the copy. I then quit the mysql client, enter the mysql data directory and create a tar backup file of the pdns database directory and the Innodb data files in the mysq data directory. I then enter the mysql client again and type the commands:

```
mysql> show master status;
+-------------+----------+-------------+-----------------+
| File | Position | Binlog_do_db | Binlog_ignore_db |
+-------------+----------+-------------+-----------------+
| ns1-bin.001 | 17015123 | | |
+-------------+----------+-------------+-----------------+
1 row in set (0.00 sec)
mysql> unlock tables;
```

The second command will unlock the tables again. The first command gives me some output I need to start the replication on the slave server. The first field in the output is the name of the currently used binary log file on the master server. The second field is the current position in the binary log file. I the copy the tar file with the database and the Innodb data files to the mysql data directory on the slave server with `scp`. I untar the tar file. I then stop the slave server. The slave server also needs some information about the master in a file called `master.info` in the mysql data directory. In this file I type some parameters one parameter per line. The lines contain the information:

```
name of the binary log file currently used on the master
position in the binary log file
ip address of the master
username of the replication user
password of the replication user
tcp port used for connection
slave connection retry in seconds
```

Each piece of information on a line by itself. I fill in the gathered information and save the file. The mysql server will then start the replication the next time it starts. All the options above have been implemented in the `/etc/mysql/my.cnf` configuration file. The file is shown in 7.17. The access to the mysql server through the UNIX socket can also be limited. The command `# chmod 0770 /var/chroot/pdns/run/mysql.sock && chown mysql.pdns /var/chroot/pdns/run/mysql.sock` will limit the access through the socket to the user mysql and the members of the pdns group. The mysql server recreates the socket on every startup so I put the command mentioned above in the mysql startup script `/etc/init.d/mysql` just after line 38. The mysql-server package comes with a test database called test. This database is not needed and is deleted with the command `# echo 'drop`

[22]More recent versions of mysql are able to start this replication with any file copying.

`database test;' | mysql`. The last thing I do to configure the mysql server is to set the mysql root password with the command `# mysqladmin password '<mynewpassword>'`

3.17 Configuring and Hardening PowerDNS

The powerdns server is configured in the file `/etc/powerdns/pdns.conf`. This file has many comments - some more usefull than others. I have created my own version of the file and written some more useful comments. The powerdns server is configured

- to run as an unprivileged user
- not to do recursive queries
- not to allow zone transfers
- allow tcp queries[23]
- not to use forwarders
- change root directory to `/var/chroot/pdns`
- connect to mysql through the socket in the chroot environment
- use the mysql user pdns to connect to the mysql server
- not to act as a master
- not to act as a slave
- not to start the webserver
- not to use URL and MBOXFW records

The configuration is shown with comments in 7.18. The unprivileged user pdns and the group pdns needs to be created. I use the command:

```
# groupadd pdns
```

to create the group and the command:

```
# useradd -s /bin/false -d /var/chroot/pdns pdns
```

to create the user with disabled password, disabled shell and home directory in `/var/chroot/pdns`.

[23]Since it is required to be an authoritive server by some registries

3.17.1 Create PowerDNS backend database

I need to load the structure of the powerdns database and add a zone to the database to show that the server is operational. The database is created on the master MySQL server before the initial copy of the database is moved to the server to initialize the replication. An sql script to create the database can be found on the powerdns documentation site[12]. The sql script is saved in the file pdns.sql and the lines

```
create database pdns;
use pdns;
```

are added to the top of the script. In the configuration I have chosen to run on the server - single instance master DNS server where Mysql takes care of the replication - there is no need for the pdns mysql user to have other rights that select on the tables in the PowerDNS database. I have removed the GRANT statements in the last three lines of the original script and added my own more restrictive GRANT statement. I have also set a password for the pdns mysql user. The full script is shown in 7.19. The word password in the last line should be changed with a strong password. The database is created with the command:

```
# mysql -p <pdns.sql
```

The three GRANT statement in the end of the `pdns.sql` script is also executed on the slave server. These access right are part of the database called mysql that is not replicated from the master server.
I also need some test data. On the site [12] I find a small sql script to create the zone test.com with some records. I save the script on the master server in the file test.com.sql shown in 7.20 and use the command:

```
# mysql pdns -p <test.com.sql
```

on the master server to import the zone data to the database on the master MySQL server. The test data is imported after the replication has been started.

In 5.1.1 I will show that the data has replicated to the slave server and that the PowerDNS server answers queries against the test.com zone.

3.18 Final changes

The server configuration is now finished and I now install the correct `/etc/fstab` shown in 7.1. The file systems on all partitions originally created with the ext2 file system are now changed to ext3 file system. For example the file system on the `/dev/sda6` partition is changed with the command: `# tune2fs -j /dev/sda8`. The partition `/dev/sda8` can now be remounted as an ext3 file system This enhancement will make the server less vulnerable to power outages as the ext3 file system is a journaled file system.

I then reboot the server to mount the partitions according to the new `fstab` file. After the reboot I the run the command:
`apt-get update && apt-get upgrade` to update the local package catalog and upgrade all packages installed on the server. This also shows that upgrade works even though the firewall doesn't normally permit downdloads and the partitions containing the binaries on the server are normally mounted read-only.

4 Design and Implement Ongoing Maintenance Procedures

4.1 Managing Security

I have only used pre-packaged software for this server. But if you need to use software that is not available in packages for you linux distribution I will seriously recommend that you compile the software and build a package for your distribution. It may be nearly impossible to manage security if the people who compiled the software leaves the company and there is no log of what packages are installed on a system. I have used the Debian packages interface to install software on this servers, so I can always issue a `# dpkg --list` command to see what software I have installed on the server.

4.2 Security announcements

There are some main sources where security issues in different products are announced. Many of them are discussion forums in the form of heavy load mailing lists. You need to pick the sources you follow carefully or you will drown in emails. Of course none of the sources will be of any use if you need to read so many emails a day that you don't read any of them. First of all I would recommend that the administrator for this server should follow the announcement and the security mailing lists for all the products installed on the server:

- Debian announce[13]
- Debian security announce[13]
- PowerDNS announce[14]
- Linux kernel announce[15]

These four mailing lists will cover all the software installed on the server. Security announcements for all the software installed from the debian distribution will be sent out on Debian security announce. Important issues in Debian linux[24] will be announced on the Debian announce list. Information about new kernels or kernel issues is announced on the Linux kernel announce list. New releases and important information about PowerDNS is announced on the PowerDNS announce list. The most dangerous thing about the configuration of powerdns is that some configuration parameters change their effect to the opposite with a new release of powerdns. So it is really important to read the release notes very carefully before you install a new version of PowerDNS.
The administrator also needs information about newly discovered vulnerabilities. New vulnerabilities might not be fixed by the security teams at the moment they are discovered. But some of them can be circumvented by making cofiguration changes in the software they concern or in other software [25]. The administrator also needs to be aware that some projects and

[24] I.e. the Debian server compromise in 2003.
[25] I.e change kernel configuration.

companies do not like full or partial disclosure of vulnerabilities in their products. They may try to forget the vulnerability or dismiss the disclusure as FUD[26]. Although this is not the typical behaviour of project owners in the open source community, it is still good to know of these security issues even before the security team has produced a patch to close the vulnerability. I would then recommend the administrator to follow the lists:

- Secunia advisories - unbiased advisories that covers almost any software product[16]
- Cert advisories - unbiased advisories from the cert organization[17]
- Bugtraq lists - mailing list where newly discovered vulnerabilities and general security issues are discussed[18]
- Full-Disclosure - mailing list where newly discovered vulnerabilities and general security issues are discussed[19]

If new vulnerabilities are discovered in the software that run on the server the administrator should follow the guidelines given in 4.3 to update the server. The administrator should also consult the different mailing lists mentioned above to check if there are any configuration change that can be made to migitate the impact of the vulnerabilities if a patch has not been released yet..

4.3 Security updates

If I keep my company package repository up to date[27] I can simply use the command:

```
# apt-get update
```

to update the list of available package versions and then the command:

```
# apt-get upgrade
```

to download the most recent version of all the packages installed on the server and install them. I need to configure apt[28] to do some things before I can use these commands to update the packages on the server. I have prohibited downloads in the local firewall so I need to open for new connections from the server to the debian security server security.debian.org and to the company package repository to tcp port 80 and established connections from security.debian.org and from the company package repository from tcp port 80 to the server. I also need to remount the partitions I have mounted read-only with the rw[29] option to update the binaries on those partitions. Some of the partitions mount with the noexec option also needs to be remounted with the exec option. This is required by the install and uninstall scripts in the debian packages. I do not want aide to react to my own system updates. So I configure apt to update the aide database after I have updated the packages. The last thing

[26] Fear, Uncertanity, and Doubt

[27] I.e. I need to download the new version of the pdns-static package whenever one becomes available and put it on the company package repository.

[28] The package that provides apt-get

[29] read-write

I configure apt to do is to show a list of packages apt is going to upgrade. It is not possible to configure apt to change the rules in the firewall before it tries to download the packages. Instead I have written a small script shown in 7.3 that will open the firewall, download available update, notify an administrator by email if there were any updates available and close the firewall again. This script called `apt-alert.sh` is configured to run every night through root's crontab. Then the administrator only needs to turn the command:

```
# apt-get upgrade
```

to install the downloaded packages. The whole apt configuration is shown in 7.2. The `apt.conf` file is saved in the directory `/etc/apt` on the server.

I also need to act on vulnerabilities in the linux kernel. Whenever a vulnerability is discovered in the Linux kernel I will install a newer kernel if the vulnerability fix is available in a newer version of the Linux kernel. Otherwise I will see if there are other ways to mitigate the impact of the discovered vulnerability and apply them to the server. The aide database should be copied with scp from the server to the development server every time the server has been updated. The aide database on the server can be verified against the copy on the development server to check that the database has not been replaced by an attacker.

4.4 Ongoing system audits

Once every three months the server be checked with this check list

- The server should be scanned for newly discovered vulnerabilities:
  - I use the Nessus[21] vulnerability scanner to perform the test
- The server should be checked for a system compromise:
  - Boot on a Knoppix-std CDROM
  - Checked the server with chkrootkit
  - Check for week passwords with John the ripper
  - Use scp to copy the aide database from `/var/lib/aide/aide.db` to an internal server. Check if the aide database matches the last backup copy.

4.4.1 Test the server for new vulnerabilities with Nessus

To run a security scan on the server with the Nessus vulnerability scanner I need a nessus server and a nessus client. These can be installed on the same machine. As I don't have any linux machines with graphical user interface in the network I install the nessus server on the development server and the client on my Windows XP workstation. From the Nessus download site [22] I download the NessusWX native Win32 client. I install it on my workstation with the default installation options. I also download the nessus 2.0.12 installer `nessus-installer.sh` from the Nessus ftp server[23] to the development server. As I download these packages directly from the Nessus projects own servers and not from a mirror I will not have to verify the MD5 sums of the packages. I install the nessus server on the development server with default installation options like this:

```
# sh nessus-installer.sh
```

The nessus installer needs the debian packages ”sharutils” , ”flex”, and ”bison” to run. To add a nessus user I run the command:

```
# /usr/local/sbin/nessus-adduser
Using /var/tmp as a temporary file holder

Add a new nessusd user
----------------------



Login :  sos
Authentication (pass/cert) [pass]:  cert
Please enter User Distinguished Name:
Country:  Denmark
STate:
Location:  CPH
Organization:  Giac
Organizational Unit:
Common Name:
e-Mail:

User rules
----------
nessusd has a rules system which allows you to restrict the hosts
that sos has the right to test.  For instance, you may want
him to be able to scan his own host only.

Please see the nessus-adduser(8) man page for the rules syntax

Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)



Login :  sos
*******
DN : /C=Denmark/L=CPH/O=Giac
Rules :



Is that ok ?  (y/n) [y]y
user added.
```

This will add the sos nessus user but it will not create a certificate for authenticating the user. To create the certificate I run the command:

```
# /usr/local/bin/nessus-mkcert-client
Do you want to register the users in the Nessus server
as soon as you create their certificates ?  (y/n):  y
This script will now ask you the relevant information to create the SSL
client certificates for Nessus.
Client certificates life time in days [365]:
Your country (two letter code) [FR]: DK
Your state or province name [none]:
Your location (e.g.  town) [Paris]:  CPH
Your organization [none]:  Giac
Your organizational unit [none]:
*********
We are going to ask you some question for each client certificate
If some question has a default answer, you can force an empty answer by
entering a single dot '.'
********
User #1 name (e.g.  Nessus username):  sos
Certificate, key or Nessus DN file(s) already exist.
Do you want to go on and overwite it/them?  y
Client certificates life time in days [365]:
Country (two letter code) [DK]:
State or province name []:
Location (e.g.  town) [CPH]:
Organization [Giac]:
Organization unit []:
e-mail []:
warning, not much extra random data, consider using the -rand option
Generating RSA private key, 1024 bit long modulus
...........++++++
..................++++++
e is 65537 (0x10001)
Using configuration from /tmp/nessus-mkcert.29522/stdC.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:State or Province Name (full name) [Some-State
]:Locality Name (eg, city) []:Organization Name (eg, company) [Internet Widgits
Pty Ltd]:Organizational Unit Name (eg, section) []:Common Name (eg, your name or
your server's hostname) []:Email Address []:Using configuration from /tmp/nessu
s-mkcert.29522/stdC.cnf
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName :PRINTABLE:'DK'
localityName :PRINTABLE:'CPH'
organizationName :PRINTABLE:'Giac'
commonName :PRINTABLE:'sos'
Certificate is to be certified until Aug 24 19:51:46 2005 GMT (365 days)

Write out database with 1 new entries
Data Base Updated

User rules
----------
nessusd has a rules system which allows you to restrict the hosts
that has the right to test.  For instance, you may want
him to be able to scan his own host only.

Please see the nessus-adduser(8) man page for the rules syntax

Enter the rules for this user, and hit ctrl-D once you are done:
(the user can have an empty rules set)
User added to Nessus.
Another client certificate?  n
Your client certificates are in /tmp/nessus-mkcert.29522
You will have to copy them by hand
```

I use scp to copy the client certificates and private key to my Windows XP workstation. I then delete the directory `/tmp/nessus-mkcert.29522` from the development server. The nessus server plugins needs to be updated before we can start the scan. Each plugin is a script used in a nessus test case. To get the best scanning and find the most recent discovered vulnerabilities I update the plugins every time I use the nessus server. On the development server I run the command below to update the nessus plugins.

```
# /usr/local/sbin/nessus-update-plugins
```

I now start the nessus server on the development server.

```
# /usr/local/sbin/nessusd -D
```

On the Windows XP workstation I open the NessusWX program. I press Alt-f-l to open the "Client Certificates" dialog. I click the install button and type sos in the "Certificate name" box, click the "..." button and browse for the client certificate I moved before with scp. I choose the certificate file `cert_nessuswx_sos.pem`. It is only one readable by the NessusWX client. I then click on the OK button to install the certificate and private key and click the close button to return to the NessusWX console. I can now press F4 to connect to the development server. I fill in the fully qualified domain name of the development server, the user name sos, click ”authenticate by certificate”, and choose the sos certificate in the drop down box. I then click the connect button to connect to the server. The NessusWX client now downloads the plugin information from the server. From the ”Session” menu I choose ”New...” to create a new scan session. I write PowerDNS server in the session name box and press enter to continue. On the ”Targets” tab I click ”add”, type the ip address of the PowerDNS server, and click ”ok”. On the ”Options” tab I add ”Do reverse DNS lookups” and ”Resolve unknown services”. On the ”Portscans” tab I choose ”specific port range” and type `1-65535` in the box to scan all ports. I enable the ”Nmap” scan and click configure to change the Nmap scan options. In the Configure ”Nmap” plugin dialog I change ”UDP port scan ” to yes, ”Identify the remote OS” to yes, ”Use hidden options to identify the remote OS” to yes, ”Fragment IP packets” to yes, and click ”ok”. I also disable ”Ping the remote host” on the ”Portscan” tab. On the ”Plugins” tab I choose ”Use sessions-specific plugin set”, click ”Select plugins”, click ”Enable Non-DoS”, click ”no” to keep my port scanner selection intact, and click ”close”. I then click ”ok” to save the session. I then right click on the session and select ”execute” from the menu. I click ”execute” to start the scan. The scan takes a while and should not be interrupted. When the scan has finished I click ”close”, select the session result which was just produced, and click ”View” to view the results. During an audit like this the administrator should pay special attention to any changes he or she finds in the results compared to the last audit.

4.4.2 Offline system check with Knoppix-std

To perform the offline system check I need a CDROM of the most recent version of Knoppix-std[30]. I download an ISO image of the CDROM from the closest Knoppix-std mirror found on the Knoppix-std download site[25]. I download the ISO image to the development server. I also find the MD5 checksum for the image on the download site. I put the line:

```
de03204ea5777d0e5fd6eb97b43034cb knoppix-std-0.1.iso
```

in a file called knoppix-std-0.1.iso.md5 in the same directory as the ISO image. I then run the command:

```
# md5sum -cv knoppix-std-0.1.iso.md5
knoppix-std-0.1.iso OK
```

The output shows that the MD5sum verifies. I don’t need to check gpg signature on the MD5 sum as I found that on the official Knoppix-std website and not on a mirrror. can then burn the ISO image on a CDROM and boot the server on the CDROM. When the server has booted and the graphical user interface is started I right click the mouse on the desktop. From the fluxbox menu I choose ”XShells” and then ”Root ATerm”. I then create the directory `/mnt/rootdir` and mount the hard drive partitions according to the original fstab7.1 except that I use the directory `/mnt/roodir/` as the root drive. In example I mount `/dev/sda5` on `/mnt/rootdir` and then `/dev/sda1` on `/mnt/rootdir/boot` and so on.
I can now run the first check. I use the command:

[30]Version 0.1 at the time of this writing.

```
chkrootkit -r /mnt/rootdir/
```

to check for rootkits installed on the server. some of the checks will not be done as they involve checking the running kernel which is not the server kernel while the system is running on the bootable CDROM. The output of the check will show if there are any rootkits installed. The second check I run on the server is performed with the password cracker ”John the ripper”. Knoppix-std has both John and a long wordlist usefull for cracking passwords. To run John on the shadow file I execute the following commands:

```
# cd /etc/john
# cat /usr/bin/pwd-tools/allwords2.dictionary \
| ./john -stdin /mnt/rootdir/etc/shadow
```

This will run the standard John the ripper passwords cracks and also check the passwords against all the words in the 27Mb large dictionary `allwords2.dictionary`. The output of the command will show the weak passwords on the server. These passwords should be changed after the check. The check takes some time so if the administrators are not able to schedule the downtime for the server I would recommend that the lesser standard check is run. To run the standard check I execute the commands:

```
# cd /etc/john
# ./john /mnt/rootdir/etc/shadow
```

The output of the command will show the weak passwords.

4.4.3 Verification of the aide database

The aide database needs to be verified every three months. A backup of the aide database has already been made during the last system update. The backup is the file aide.db.backup The current aide database is copied from the server to the development server with scp. I can now verify the database with the commands:

texttt# md5sum aide.db.backup — sed ’s/ḃaackup//’ ¿ aide.db.md5

texttt# md5sum -cv aide.db.md5

```
aide.db OK
```

The output shows that the md5sum of the currently installed database matches the md5sum of the backup. This shows that none of the monitored files on the server has been changed.

4.5 Backup requirements

The backup requirements for this server are not big since the data on the server is replicated from a master database server and will be backed up on the master server. All the install scripts and configuration files are standard installation files. They are stored on a file server on the internal network and backed up from the file server. The configuration files for the PowerDNS server and the MySQL server should be backed up after the installation and every time anyone changes the configuration of the PowerDNS server or the MySQL server. The files can be backup by using scp from the development server:

```
# scp -p 1803 sos@<powerdns server ip>:/etc/mysql/mycnf ./
# scp -p 1803 sos@<powerdns server ip>:/etc/powerdns/pdns.conf ./
```

5 Test and Verify the Setup

5.1 Test functionality of the server

5.1.1 PowerDNS server

To check the functionality of the PowerDNS server I execute a couple of test queries against the server. I have already loaded the test.com zone to the database on the master MySQL server. If the PowerDNS server answers queries to the test.com zone it shows that the replication works, the stunnel instance is up and running, and that the PowerDNS server is up and running First I execute a normal udp query against the server. From a server on the internal network I execute the command:

```
# dig @<powerdns server ip> test.com soa
; <<>> DiG 9.2.1 <<>> @<powerdns server ip> test.com soa
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode:  QUERY, status:  NOERROR, id:  5073
;; flags:  qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;test.com.  IN SOA

;; ANSWER SECTION:
test.com.  86400 IN SOA localhost.  ahu.ds9a.nl.  1 10800
3600 604800 40001

;; Query time:  11 msec
;; SERVER: <powerdns server ip>#53(<powerdns server ip>)
;; WHEN: Mon Jul 26 18:40:35 2004
;; MSG SIZE rcvd:  82
```

I get the expected answer. I the excute the same query over the tcp protocol. From the server on the internal network I execute the command:

```
# dig @<powerdns server ip> test.com soa
; <<>> DiG 9.2.1 <<>> +tcp @<powerdns server ip> test.com soa
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode:  QUERY, status:  NOERROR, id:  5073
;; flags:  qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;test.com.  IN SOA

;; ANSWER SECTION:
test.com.  86400 IN SOA localhost.  ahu.ds9a.nl.  1 10800
3600 604800 40001

;; Query time:  11 msec
;; SERVER: <powerdns server ip>#53(<powerdns server ip>)
;; WHEN: Mon Jul 26 18:42:26 2004
;; MSG SIZE rcvd:  82
```

The answer I get is the expected. The server is now functional. It answers queries via tcp or udp, stunnel instances work to make the server replicate the database over an SSL connection, and database replication is also up and running.

5.2 Verify the enhanced security implementations

5.2.1 Verify security in general

To verify that it is not possible to log in as root locally I turn to the console on the server and try to log in as root. I get a nice ”Permission denied” message. I also verify that the legal warning from 7.12 is displayed on the console. I need to create a test user on the server to verify that it is not possible to log in locally or via ssh if the user is not a member of the ”admins” group. To create this user I use the commands:

```
# useradd -s /bin/sh testuser
# passwd testuser
```

I set the password for the testuser account. I can then try to log in locally with the testuser account. I also get a nice ”Permission denied” messsage. I will verify that the user is not able to log in via ssh in 5.2.2. To verify that the firewall is in place and that no outgoing connections to other servers can be made I use telnet. The program can be used to create a tcp connection to another server. I use the command:

```
# telnet <ip of local apt repository> 80
Trying <ip of local apt repository>
telnet:  Unable to connect to remote host:  Connection timed out
```

to try to connect to the web server on the corporate apt repository server. The output shows that this is not possible.

I check the log and the central syslog server and verify that all logs from the system are sent via the stunnel instance to the central syslog server.

5.2.2 Verify security in sshd

To verify that it is not possible to log in via ssh with password authentication I try to log in from the development server from a user that doesn't have access to the private dsa key matching the public key on the server.

```
# ssh -p 1803 sos@<ip of powerdns server>
Unauthorized access to this machine is prohibited.
Use of this system is limited to authorized individuals only.
All activity is monitored.
Permission denied (publickey).
```

I get the legal warning banner followed by a nice "Permission denied".

I then try to log on the server via ssh as the root user:

```
# ssh -p 1803 root@<ip of powerdns server>
Unauthorized access to this machine is prohibited.
Use of this system is limited to authorized individuals only.
All activity is monitored.
Permission denied (publickey).
```

I get the legal warning banner followed by a nice "Permission denied".

To demonstrate that it is only possible to log on the server via ssh as a member of the admins group try to use the private key for authentication and log on as the testuser account. First I have to copy the public key used in the authetication process to the testuser home directory:

```
# mount /home/ -o remount,rw
# mkdir -p /home/testuser/.ssh
# cp /home/sos/.ssh/authorized_keys /home/testuser/.ssh/
# chown -R testuser /home/testuser/
# mount /home/ -o remount,ro
```

I then try to log on the server as the testuser:

```
# ssh -p 1803 testuser@<ip of powerdns server>
Unauthorized access to this machine is prohibited.
Use of this system is limited to authorized individuals only.
All activity is monitored.
Permission denied (publickey).
```

I get a nice "Permission denied" so everything seems to be in place. I now delete the user account "testuser" with the command:

```
# userdel testuser && mount /home -o remount,rw && rm -rf /home/testuser
&& mount /home -o remount,ro
```

5.2.3 Verify security in PowerDNS server

Two basic tests on DNS server setup is to verify that no recursion can be made, and that it is not possible to make zone transfers from the server.

If recusion is enabled for all clients it will be possible for clients to make the server perform lookups on other servers. If an exploit that utilized the payload of a query to exploit a DNS server was found, it would be possible for a client to use the recursive DNS server to send this exploit to other DNS servers. To verify that no recursion is possible I try to query the server against an existing zone that is not hosted on the server.

```
# dig @<powerdns server ip> example.com

; <<>> DiG 9.2.1 <<>> @<powerdns server ip> example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode:  QUERY, status:  SERVFAIL, id:  57508
;; flags:  qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.  IN A

;; Query time:  1 msec
;; SERVER: <powerdns server ip>#53(<powerdns server ip>)
;; WHEN: Tue Aug 24 16:11:08 2004
;; MSG SIZE rcvd:  29
```

The PowerDNS server answers with a `SERVFAIL` which is exactly the desired answer.
If zone transfers are enabled for all clients a client will be able to get a list of all records in a zone. This is not a direct security issue. But in the sense that all information is valuable information to an attacker it is definately usable information to an attacker. To verify that no zone transfers can be made I try to query the server for a zone transfer of a zone that I know is hosted on the server.

```
# dig @<powerdns server ip> test.com axfr

; <<>> DiG 9.2.1 <<>> @<powerdns server ip> test.com axfr
;; global options:  printcmd
; Transfer failed.
```

The zone transfer fails which is the desired result. I use the command:

```
# ps auxw | grep pdns
```

to show that the PowerDNS server runs as an unprivileged user pdns. It shows in the first collumn of the output. The output of the command also shows some PowerDNS processes running as root. These are the guardian processes that restart the other threads if they die. They are not listening on the network interface. Therefore it is ok that they run as root. The fact that the MySQL socket path in the PowerDNS configuration file7.18 works shows that the PowerDNS server runs in a chroot jail. If the server wasn't jailed it would be looking for the socket in `/run/mysql.sock` which is non-existing and the server would fail.

5.2.4 Verify security in MySQL server

I have already shown that the MySQL server is running and replicating. The replication shows that the encrypted tunnel to the master MySQL server is running. Otherwise the server would not be able to replicate as the server tries to connect to a master server on the address 127.0.0.1. There will be no connection to the master server here unless the stunnel instance is running. I verify the the local mysql root password is set by starting the mysql client.

```
# mysql
ERROR 1045:  Access denied for user:
'root@localhost' (Using password:  NO)
```

The output shows that it is not possible to connect to the MySQL server without the password. I also verify that the restrictive permissions set on the mysql UNIX socket works. With these permissions set I should not be able to connect through the socket if I am logged on as the sos UNIX user.

```
$ mysql
ERROR 2002:  Can't connect to local MySQL server
through socket '/var/chroot/pdns/run/mysql.sock' (13)
```

The output shows that this is not possible and the permissions work. The server is now ready for production and can be moved to the production network.

6 The Bibliography

References

[1] PowerDNS Web site: http://www.powerdns.com/

[2] PowerDNS download site: http://www.powerdns.com/downloads/index.php

[3] Debian Web site ISO image download: http://www.debian.org/CD/http-ftp/

[4] Grsecurity kernel patches and tools: http://grsecurity.org/

[5] Debian Web site latest stable release: http://www.debian.org/CD/#latest

[6] Debian gpg keyring: http://keyring.debian.org/

[7] Debian security manual: http://www.debian.org/doc/manuals/securing-debian-howto/

[8] Kurt Siefried's Linux Administrator's Security Guide: http://www.seifried.org/lasg/

[9] Daniel Bernstien's attack on the world writeable mail drop directory in postfix: http://cr.yp.to/maildisasters/postfix.html

[10] Debian security team FAQ: http://www.debian.org/security/faq

[11] Netfilter and IPTables how-to: http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO-7.html

[12] Powerdns database structure for mysql: http://doc.powerdns.com/configuring-db-connection.html#CONFIGURING-MYSQL

[13] Debian user mailing lists: http://lists.debian.org/users.html

[14] PowerDNS announce list: http://mailman.powerdns.com/mailman/listinfo/pdns-announce

[15] Linux kernel announce list: http://vger.kernel.org/vger-lists.html

[16] Secunia advisories: http://secunia.com/secunia_security_advisories/?menu=prod

[17] Cert advisories: http://www.us-cert.gov/cas/index.html

[18] Bugtraq discussion list: http://www.securityfocus.com/archive/1

[19] Full-Disclusore discussion list: http://lists.netsys.com/mailman/listinfo/full-disclosure

[20] SANS resource site Password policy: http://www.sans.org/resources/policies/Password_Policy.pdf

[21] Nessus security scanner website: http://www.nessus.org/

[22] Nessus download site: http://www.nessus.org/download.html

[23] Nessus ftp server: http://ftp.nessus.org/nessus/nessus-2.0.12/nessus-installer/

[24] Knoppix security tools distribution: http://www.knoppix-std.org/

[25] Knoppix download site: http://www.knoppix-std.org/download.html

7 Configuration files and scripts

7.1 fstab

```
# /etc/fstab:  static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/sda5 / ext3 errors=remount-ro 0 1
/dev/sda13 none swap sw 0 0
proc /proc proc defaults 0 0
/dev/fd0 /floppy auto user,noauto 0 0
/dev/cdrom /cdrom iso9660 ro,user,noauto 0 0
/dev/sda1 /boot ext3 ro,nodev,nosuid,noexec 0 2
/dev/sda6 /usr ext3 ro,nodev 0 2
```

```
/dev/sda7 /var ext3 noexec,nosuid,nodev 0 2
/dev/sda8 /home ext3 ro,noexec,nosuid,nodev 0 2
/dev/sda9 /tmp ext3 noexec,nosuid,nodev 0 2
/dev/sda10 /var/lock ext3 noexec,nosuid,nodev 0 2
/dev/sda11 /var/tmp ext3 noexec,nosuid,nodev 0 2
/dev/sda12 /opt ext3 ro,noexec,nosuid,nodev 0 2
```

7.2 apt.conf

```
//show a list of packages going to be upgraded
APT::Get::Show-Upgraded "True";
DPkg
{
Pre-Invoke {
//remount file systems for rw and exec
//many packages require these options to install
"/bin/mount -o remount,exec /var";
"/bin/mount -o remount,rw /usr";
"/bin/mount -o remount,exec /tmp";
"/bin/mount -o remount,exec /var/tmp";
};
Post-Invoke {
//remount file systems with enabled security
"/bin/mount -o remount,noexec /var";
"/bin/mount -o remount,ro /usr";
"/bin/mount -o remount,noexec /tmp";
"/bin/mount -o remount,noexec /var/tmp";
//update the aide database to reflect the installed/removed packages
"/usr/bin/aide --update && /bin/mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db";
};
};
```

7.3 apt-alert.sh

```
#!/bin/sh
APTGET=/usr/bin/apt-get
IPTABLES=/sbin/iptables
#ip of security.debian.org
DEBIANSECWEB="194.109.137.218"
ADMIN="sos@s0s.dk"
```

```
HOSTNAME=`/bin/hostname`

#temporarily open the firewall

$IPTABLES -A OUTPUT -d <ip of company apt repository> -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

$IPTABLES -A INPUT -s <ip of company apt repository> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

$IPTABLES -A OUTPUT -d $DEBIANSECWEB -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

$IPTABLES -A INPUT -s $DEBIANSECWEB -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

#perform the update

$APTGET update >/dev/null

$APTGET -d --yes upgrade | grep -E "[1-9]" >/dev/null

if [ $?  -eq 0 ] ;

then

#alert administrator if there were updates available

$APTGET -d --yes upgrade | mail $ADMIN -s "update are available for $HOSTNAME"

fi

#close the firewall again

$IPTABLES -D OUTPUT -d <ip of company apt repository> -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

$IPTABLES -D INPUT -s <ip of company apt repository> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

$IPTABLES -D OUTPUT -d $DEBIANSECWEB -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

$IPTABLES -D INPUT -s $DEBIANSECWEB -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

7.4 installpackage.sh

```
#!/bin/sh
#the settings
PACKAGE=$1
APTGET=/usr/bin/apt-get
#the action
#temporarily open the firewall
/sbin/iptables -A OUTPUT -d <ip of company apt repository> -p tcp --dport 80 -m state --state NEW,ESTABLISHED
-j ACCEPT
/sbin/iptables -A INPUT -s <ip of company apt repository> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -d <ip of local debian mirror> -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j
ACCEPT
/sbin/iptables -A INPUT -s <ip of local debian mirror> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
#install the package
$APTGET install $PACKAGE
#Close the firewall again
/sbin/iptables -D OUTPUT -d <ip of company apt repository> -p tcp --dport 80 -m state --state NEW,ESTABLISHED
-j ACCEPT
/sbin/iptables -D INPUT -s <ip of company apt repository> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
/sbin/iptables -D OUTPUT -d <ip of local debian mirror> -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j
ACCEPT
/sbin/iptables -D INPUT -s <ip of local debian mirror> -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

7.5 initial.sh

```
#!/bin/sh

#create administrative group

groupadd admins
```

```
#add administrative user and make the user a member of the admins group
#the "-g admins -G admins" is need to pass the pam su restrictions useradd -g admins -G admins sos
echo "Set password for sos"
passwd sos
mkdir /home/sos
chown sos.admins /home/sos
#disable inetd daemon
update-rc.d -f inetd remove
update-rc.d inetd stop 20 0 1 2 3 4 5 6 .
#disable shells for all system users
cat /etc/passwd | sed 's/sh$/false/' >/etc/passwd.tmp
mv /etc/passwd.tmp /etc/passwd
#apply patch to /etc
cd /
patch -p0 <etc.patch
```

7.6 etc.patch

```
diff -rc /etc/inetd.conf etc/inetd.conf
** /etc/inetd.conf Wed Aug 18 20:41:15 2004
--- etc/inetd.conf Wed Aug 18 19:36:47 2004
**************
** 18,28 ****
#echo dgram udp wait root internal
#chargen stream tcp nowait root internal
#chargen 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
!  time stream tcp nowait root internal
#time dgram udp wait root internal

#:STANDARD: These are standard services.
--- 18,28 ----
#echo dgram udp wait root internal
#chargen stream tcp nowait root internal
#chargen 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
!  #time stream tcp nowait root internal
#time dgram udp wait root internal

#:STANDARD: These are standard services.
**************
** 30,36 ****
#:BSD: Shell, login, exec and talk are BSD protocols.

#:MAIL: Mail, news and uucp services.
!  #disabled#smtp stream tcp nowait mail /usr/sbin/exim exim -bs

#:INFO: Info services

--- 30,36 ----
#:BSD: Shell, login, exec and talk are BSD protocols.

#:MAIL: Mail, news and uucp services.
!  #smtp stream tcp nowait mail /usr/sbin/exim exim -bs

#:INFO: Info services

diff -rc /etc/inittab etc/inittab
** /etc/inittab Fri Jan 25 12:53:58 2002
--- etc/inittab Wed Aug 18 19:24:41 2004
**************
** 30,36 ****
z6:6:respawn:/sbin/sulogin

# What to do when CTRL-ALT-DEL is pressed.
!  ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let this work."
--- 30,36 ----
z6:6:respawn:/sbin/sulogin

# What to do when CTRL-ALT-DEL is pressed.
!  #ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let this work."
```

```
diff -rc /etc/issue etc/issue
** /etc/issue Wed Nov 28 11:18:48 2001
--- issue.net Mon Aug 23 21:17:35 2004
**************
** 1,2 ****
!  Debian GNU/\s 3.0 \n \l
!
--- 1,3 ----
!  Unauthorized access to this machine is prohibited.
!  Use of this system is limited to authorized individuals only.
!  All activity is monitored.
diff -rc /etc/login.defs etc/login.defs
** /etc/login.defs Sun Apr 7 15:59:14 2002
--- etc/login.defs Wed Aug 18 19:24:41 2004
**************
** 51,62 ****
#
# Enable display of unknown usernames when login failures are recorded.
#
!  LOG_UNKFAIL_ENAB no

#
# Enable logging of successful logins
#
!  LOG_OK_LOGINS no

#
# Enable setting of ulimit, umask, and niceness from passwd gecos field.
--- 51,64 ----
#
# Enable display of unknown usernames when login failures are recorded.
#
!  # default no
!  LOG_UNKFAIL_ENAB yes

#
# Enable logging of successful logins
#
!  # default no
!  LOG_OK_LOGINS yes

#
```

```
# Enable setting of ulimit, umask, and niceness from passwd gecos field.
**************
** 174,181 ****
# PASS_MIN_DAYS Minimum number of days allowed between password changes.
# PASS_WARN_AGE Number of days warning given before a password expires.
#
!  PASS_MAX_DAYS 99999
!  PASS_MIN_DAYS 0
PASS_WARN_AGE 7

#
--- 176,188 ----
# PASS_MIN_DAYS Minimum number of days allowed between password changes.
# PASS_WARN_AGE Number of days warning given before a password expires.
#
!  # default
!  #PASS_MAX_DAYS 99999
!  #PASS_MIN_DAYS 0
!  #PASS_WARN_AGE 7
!
!  PASS_MAX_DAYS 180
!  PASS_MIN_DAYS 7
PASS_WARN_AGE 7

#
diff -rc /etc/pam.d/login etc/pam.d/login
** /etc/pam.d/login Tue Aug 17 20:12:20 2004
--- etc/pam.d/login Wed Aug 18 19:24:41 2004
**************
** 46,52 ****
# Uncomment and edit /etc/security/access.conf if you need to
# set access limits.
# (Replaces /etc/login.access file)
!  # account required pam_access.so

# Standard Un*x account and session
account required pam_unix.so
--- 46,52 ----
# Uncomment and edit /etc/security/access.conf if you need to
# set access limits.
# (Replaces /etc/login.access file)
!  account required pam_access.so
```

```
# Standard Un*x account and session
account required pam_unix.so
diff -rc /etc/pam.d/su etc/pam.d/su
** /etc/pam.d/su Tue Aug 17 20:24:17 2004
--- etc/pam.d/su Wed Aug 18 19:24:41 2004
**************
** 7,13 ****
# to the end of this line if you want to use a group other
# than the default "root".
# (Replaces the ‘SU_WHEEL_ONLY’ option from login.defs)
!  # auth required pam_wheel.so

# Uncomment this if you want wheel members to be able to
# su without a password.
--- 7,13 ----
# to the end of this line if you want to use a group other
# than the default "root".
# (Replaces the ‘SU_WHEEL_ONLY’ option from login.defs)
!  auth requisite pam_wheel.so group=admins debug

# Uncomment this if you want wheel members to be able to
# su without a password.
diff -rc /etc/security/access.conf etc/security/access.conf
** /etc/security/access.conf Mon Jan 21 20:25:02 2002
--- etc/security/access.conf Wed Aug 18 19:24:41 2004
**************
** 39,51 ****
#
##############################################################################
#
!  # Disallow non-root logins on tty1
#
!  #-:ALL EXCEPT root:tty1
#
# Disallow console logins to all but a few accounts.
#
!  #-:ALL EXCEPT wheel shutdown sync:console
#
# Disallow non-local logins to privileged accounts (group wheel).
#
--- 39,51 ----
#
##############################################################################
```

```
#
!  # Disallow root logins
#
!  -:root:ALL
#
# Disallow console logins to all but a few accounts.
#
!  -:ALL EXCEPT admins:ALL
#
# Disallow non-local logins to privileged accounts (group wheel).
#
```

7.7 install-apps.sh

```
#!/bin/sh
APTGET=/usr/bin/apt-get
APPS="ntpdate mysql-server ssh postfix aide syslog-ng stunnel pdns-static"
$APTGET update
for app in $APPS;
do
$APTGET --yes install $app;
done
```

7.8 aide.conf

```
# AIDE conf
database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new
# Change this to "no" or remove it to not gzip output
# (only useful on systems with few CPU cycles to spare)
gzip_dbout=yes
# Here are all the things we can check - these are the default rules
#
#p:  permissions
#i:  inode
#n:  number of links
#u:  user
#g:  group
#s:  size
#b:  block count
#m:  mtime
h #a:  atime
#c:  ctime
#S: check for growing size
#md5:  md5 checksum
#sha1:  sha1 checksum
#rmd160:  rmd160 checksum
#tiger:  tiger checksum
#R: p+i+n+u+g+s+m+c+md5
#L: p+i+n+u+g
#E: Empty group
#>:  Growing logfile p+u+g+i+n+S
```

```
# This is the email address reports get mailed to
# It’s only used by the cron script and at the moment only the first address
# specified in this manner will be used.
@@define MAILTO root
@@define LINES 1000
# Custom rules
Binlib = p+i+n+u+g+s+b+m+c+md5+sha1
ConfFiles = p+i+n+u+g+s+b+m+c+md5+sha1
Logs = p+i+n+u+g+S
Devices = p+i+n+u+g+s+b+c+md5+sha1
Databases = p+n+u+g
StaticDir = p+i+n+u+g
ManPages = p+i+n+u+g+s+b+m+c+md5+sha1
# Next decide what directories/files you want in the database
# Kernel, system map, etc.
=/boot$ Binlib
# Binaries
/bin Binlib
/sbin Binlib
/usr/bin Binlib
/usr/sbin Binlib
/usr/local/bin Binlib
/usr/local/sbin Binlib
/usr/games Binlib
# Libraries
/lib Binlib
/usr/lib Binlib
/usr/local/lib Binlib
# Log files
/var/log$ StaticDir
/var/log/aide/aide.log(.[0-9])?(.gz)?  Databases
/var/log/aide/error.log(.[0-9])?(.gz)?  Databases
/var/log/setuid.changes(.[0-9])?(.gz)?  Databases
/var/log Logs
# Devices
!/dev/pts
/dev Devices
# Other miscellaneous files
/var/run$ StaticDir
!/var/run
# Test only the directory when dealing with /proc
/proc$ StaticDir
!/proc
# You can look through these examples to get further ideas
# MD5 sum files - especially useful with debsums -g
#/var/lib/dpkg/info/([^
.]+).md5sums
# Check crontabs
#/var/spool/anacron/cron.daily Databases
#/var/spool/anacron/cron.monthly Databases
#/var/spool/anacron/cron.weekly Databases
/var/spool/cron Databases
/var/spool/cron/crontabs Databases
# manpages can be trojaned, especially depending on *roff implementation
/usr/man ManPages
/usr/share/man ManPages
/usr/local/man ManPages
# docs
/usr/doc ManPages
/usr/share/doc ManPages
# check users’ home directories
/home Binlib
# check sources for modifications
#no sources on this server
#/usr/src L
```

```
#/usr/local/src L
# Check headers for same
/usr/include L
/usr/local/include L
# Check powerdns chroot environment
/var/chroot/pdns/ Binlib
```

7.9 stunnel.sh

```
#!/bin/sh

case "1”in
start)
#CreateSSLtunnel forsyslog
/usr/sbin/stunnel − c − d127.0.0.1 : 515 − rsyslog.giac.net : 515 − sstunnel − gstunnel − p/etc/syslog − ng/syslog.pem
#CreateSSLtunnel forMySQL
/usr/sbin/stunnel − c − d127.0.0.1 : 3306 − rmysql − master.giac.net : 3306 − sstunnel − gstunnel − p/etc/mysql/mysql.pem
;;
stop)
forpidin‘psauxw|grep − E”[0 − 9]/usr/sbin/stunnel − c − d127.0.0.1 : 514”|cut − d”” − f3‘;
do
kill − TERMpid;
done
;;
esac
```

7.10 syslog-ng.conf

```
#turn on long hostnames, so logentries can be matched to a host on the central syslog server
options { long_hostnames(on); sync(0); };
#allow logging to syslog-ng through local unix socket, but not through network sockets
source src { unix-dgram("/dev/log"); internal(); };

#configure some standard log destination ( i.e log file, network, device, etc.)
destination authlog { tcp("localhost" port(514)); file("/var/log/auth.log" owner("root") group("adm") perm(0600));
};
destination syslog { tcp("localhost" port(514)); file("/var/log/syslog" owner("root") group("adm") perm(0600));};
destination cron { tcp("localhost" port(514)); file("/var/log/cron.log" owner("root") group("adm") perm(0600));
};
destination daemon { tcp("localhost" port(514)); file("/var/log/daemon.log" owner("root") group("adm") perm(0600));
};
destination kern { tcp("localhost" port(514)); file("/var/log/kern.log" owner("root") group("adm") perm(0600));
};
```

```
destination lpr { tcp("localhost" port(514)); file("/var/log/lpr.log" owner("root") group("adm") perm(0600));
};
destination mail { tcp("localhost" port(514)); file("/var/log/mail.log" owner("root") group("adm") perm(0600));
};
destination user { tcp("localhost" port(514)); file("/var/log/user.log" owner("root") group("adm") perm(0600));
};
destination uucp { tcp("localhost" port(514)); file("/var/log/uucp.log" owner("root") group("adm") perm(0600));
};
destination debug { tcp("localhost" port(514)); file("/var/log/debug" owner("root") group("adm") perm(0600));
};
destination messages { tcp("localhost" port(514)); file("/var/log/messages" owner("root") group("adm") perm(0600));
};
destination console { usertty("root"); };
destination console_all { file("/dev/tty8"); };

#create filters to match incomming log traffic
filter f_authpriv { facility(auth, authpriv); };
filter f_syslog { not facility(auth, authpriv, mail, cron, daemon, kern, lpr, user, uucp); };
filter f_cron { facility(cron); };
filter f_daemon { facility(daemon); };
filter f_kern { facility(kern); };
filter f_lpr { facility(lpr); };
filter f_mail { facility(mail); };
filter f_user { facility(user); };
filter f_uucp { facility(uucp); };
filter f_messages { level(info .. warn)
and not facility(auth, authpriv, cron, daemon, mail); };
filter f_emergency { level(emerg); };
filter f_info { level(info); };
filter f_notice { level(notice); };
filter f_warn { level(warn); };
filter f_crit { level(crit); };
filter f_err { level(err); };

#match the filters to a destination( i.e. log file, network, etc.)
log { source(src); filter(f_authpriv); destination(authlog); };
log { source(src); filter(f_syslog); destination(syslog); };
log { source(src); filter(f_daemon); destination(daemon); };
log { source(src); filter(f_kern); destination(kern); };
log { source(src); filter(f_lpr); destination(lpr); };
log { source(src); filter(f_mail); destination(mail); };
log { source(src); filter(f_user); destination(user); };
log { source(src); filter(f_uucp); destination(uucp); };
log { source(src); filter(f_messages); destination(messages); };
log { source(src); filter(f_emergency); destination(console); };
```

7.11 issue.net

```
Unauthorized access to this machine is prohibited.
Use of this system is limited to authorized individuals only.
All activity is monitored.
```

7.12 issue

```
Unauthorized access to this machine is prohibited.
Use of this system is limited to authorized individuals only.
All activity is monitored.
```

7.13 sshd_config

```
# What ports, IPs and protocols we listen for
Port 1803
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
#privilege separation is turned on to keep most processes running as an unprivileged user
UsePrivilegeSeparation yes
# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768
# Logging
SyslogFacility AUTH
LogLevel INFO
# Authentication:
LoginGraceTime 600
PermitRootLogin no
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile # rhosts authentication should not be used
RhostsAuthentication no
# Don't read the user's  /.rhosts and  /.shosts files
```

```
IgnoreRhosts yes
# Disable host based RSA authetication
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no
# disable s/key passwords
ChallengeResponseAuthentication no
# disable tunneled clear text passwords
PasswordAuthentication no
# Kerberos authetication is disabled by default so we really don't need this
#KerberosAuthentication no
#disable X11 forwardning
X11Forwarding no
X11DisplayOffset 10
PrintMotd no
#disable print of last logon user and time
PrintLastLog no
#send keppalive messages to clients in order to discover lost connections
KeepAlive yes
# display legal banner before login
Banner /etc/issue.net # only allow logins from users in the admins group
AllowGroups admins
```

7.14 timesync.sh

```
#!/bin/sh
/usr/sbin/ntpdate ntp.giac.net >/dev/null
/sbin/hwclock --systohc
```

7.15 lilo.conf

```
#create password on installation or reuser existing password
password=""
# Support LBA for large hard disks.
lba32
# Specifies the boot device.
boot=/dev/sda
# Specifies the device that should be mounted as root.  ('/')
```

```
root=/dev/sda5
# Installs the specified file as the new boot sector
# You have the choice between:  bmp, compat, menu and text
# Look in /boot/ and in lilo.conf(5) manpage for details
#
install=/boot/boot-menu.b
# Specifies the location of the map file
map=/boot/map
# Specifies the number of deciseconds (0.1 seconds) LILO should
# wait before booting the first image.
delay=5
# Specifies the VGA text mode at boot time.  (normal, extended, ask, <mode>)
vga=normal
# Boot up Linux by default.
default=Linux
image=/boot/vmlinuz-2.4.26
label=Linux
read-only
restricted
image=/vmlinuz
label=LinuxOLD
read-only
restricted
```

7.16 firewall.sh

```
#!/bin/sh
#set usefull variables
LAN=eth0
IPTABLES=/sbin/iptables
LOOP=127.0.0.1
DNS_SERVERS="<space delimited ip adresses of corporate dns servers>"
SMTP_SERVERS="<space delimited ip adresses of corporate smtp relay servers>"
SSHIP="<space delimited ip adresses of servers used for remote administration via ssh>"
MYSQL_SERVERS="<space delimited ip adresses of corporate mysql master servers>"
SYSLOG_SERVERS="<space delimited ip adresses of corporate corporate syslog servers>"
NTP_SERVERS="<space delimited ip adresses of corporate ntp servers>"
TCP_PORTS="53"
UDP_PORTS="53"
#enable syncookies to avoid syn flood dos attacks
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
# DELETE old rules
$IPTABLES -F OUTPUT
$IPTABLES -F INPUT
$IPTABLES -F FORWARD
# Set default policies to DROP
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP
# Prevent external packets from using loopback addr
```

```
$IPTABLES -A INPUT -i $LAN -s $LOOP -j DROP
# Drop invalid packets
$IPTABLES -A INPUT -m state --state INVALID -j DROP;
$IPTABLES -A OUTPUT -m state --state INVALID -j DROP;
$IPTABLES -A FORWARD -m state --state INVALID -j DROP;
# Allow service on local computer
for tcpport in $TCP_PORTS;
do
$IPTABLES -A INPUT -p tcp --dport $tcpport -m state --state NEW,ESTABLISHED -j ACCEPT;
$IPTABLES -A OUTPUT -p tcp --sport $tcpport -m state --state ESTABLISHED -j ACCEPT;
done
for udpport in $UDP_PORTS;
do
$IPTABLES -A INPUT -p udp --dport $udpport -m state --state NEW,ESTABLISHED -j ACCEPT;
$IPTABLES -A OUTPUT -p udp --sport $udpport -m state --state ESTABLISHED -j ACCEPT;
done
#Allow ssh from admin server
for sship in $SSHIP;
do
$IPTABLES -A INPUT -s $sship -p tcp --dport 1803 -m state --state NEW,ESTABLISHED -j ACCEPT;
$IPTABLES -A OUTPUT -d $sship -p tcp --sport 1803 -m state --state ESTABLISHED -j ACCEPT;
done
#Allow Replication from mysql master server
for mysqlip in $MYSQL_SERVERS;
do
$IPTABLES -A INPUT -d $mysqlip -p tcp --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT;
$IPTABLES -A OUTPUT -s $mysqlip -p tcp --sport 3306 -m state --state ESTABLISHED -j ACCEPT;
done
#Allow syslog to central syslog servers
for syslogip in $SYSLOG_SERVERS;
do
$IPTABLES -A INPUT -d $syslogip -p tcp --dport 514 -m state --state NEW,ESTABLISHED -j ACCEPT;
$IPTABLES -A OUTPUT -s $syslogip -p tcp --sport 514 -m state --state ESTABLISHED -j ACCEPT;
done
#Allow dns lookups
for dnsip in $DNS_SERVERS;
do
$IPTABLES -A OUTPUT -d $dnsip -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT;
$IPTABLES -A INPUT -s $dnsip -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT;
done
#Allow smtp connections to smtp relay server
for smtpip in $SMTP_SERVERS;
do
$IPTABLES -A OUTPUT -d $smtpip -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -s $smtpip -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT
done
#Allow time synchronization through ntp
for ntpip in $NTP_SERVERS;
do
$IPTABLES -A OUTPUT -d $ntpip -p udp --dport 123 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -s $ntpip -p udp --sport 123 -m state --state ESTABLISHED -j ACCEPT
done
#log all packet not dropped or accepted at this stage to syslog
$IPTABLES -A INPUT -p all -j LOG --log-prefix "FIREWALL: "
```

7.17 my.cnf

```
# You can copy this to one of:
# /etc/mysql/my.cnf to set global options,
# mysql-data-dir/my.cnf to set server-specific options (in this
# installation this directory is /var/lib/mysql) or
#  /.my.cnf to set user-specific options.
# One can use all long options that the program supports.
# Run the program with --help to get a list of available options
# This will be passed to all mysql clients
[client]
#password = my_password
port = 3306
socket = /var/chroot/pdns/run/mysql.sock
# Here is entries for some specific programs
# The following values assume you have at least 32M ram
[safe_mysqld]
err-log = /var/log/mysql/mysql.err
[mysqld]
user = mysql
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/chroot/pdns/run/mysql.sock
port = 3306
#
# You can also put it into /var/log/mysql/mysql.log but I leave it in /var/log
# for backward compatibility.  Both location gets rotated by the cronjob.
#log = /var/log/mysql/mysql.log
log = /var/log/mysql.log
basedir = /usr
datadir = /var/lib/mysql
tmpdir = /tmp
language = /usr/share/mysql/english
skip-locking
# The skip-networkin option will no longer be set via debconf menu.
# You have to manually change it if you want networking i.e.  the server
# listening on port 3306.  The default is "disable" - for security reasons.
skip-networking
set-variable = key_buffer=16M
set-variable = max_allowed_packet=1M
set-variable = thread_stack=128K
# Here you can see queries with especially long duration
#log-slow-queries = /var/log/mysql/mysql-slow.log
# The following can be used as easy to replay backup logs or for replication
#server-id = 1
#log-bin = /var/log/mysql/mysql-bin.log
#binlog-do-db = include_database_name
#binlog-ignore-db = include_database_name
# Read the manual if you want to enable InnoDB!
# skip-innodb
innodb_data_file_path = ibdata1:30M
#start slave settings
master-host=127.0.0.1
master-user=replicate
master-password=<replace with unique strong password>
master-port=3306
server-id=<replace with unique id>
replicate-wild-do-table=pdns.#end slave settings
[mysqldump]
quick
set-variable = max_allowed_packet=1M
[mysql]
#no-auto-rehash # faster start of mysql but no tab completition
[isamchk]
set-variable = key_buffer=16M
```

7.18 pdns.conf

```
#set the user- and group id running the server
setuid=pdns
setgid=pdns
#select wich backend to launch
launch=gmysql
#configure access to backend
#the path to the mysql socket is relative to the chroot environment
gmysql-socket=/run/mysql.sock
gmysql-user=pdns
gmysql-dbname=pdns
gmysql-password=<passowrd for pdns mysql user>
#no recursion is allowed
# allow-recursion=
# cache-ttl Seconds to store packets in the PacketCache
cache-ttl=30
# chroot If set, chroot to this directory for more security
chroot=/var/chroot/pdns
# config-dir Location of configuration directory (pdns.conf)
config-dir=/etc/powerdns
#not needed.  config-name Name of this virtual configuration - will rename the binary image
# config-name=
#not needed.  control-console Debugging switch - don't use
# control-console=no
```

```
#This is also specified in the startup script.  The value below will have no effect
#unless the startup parameter is removed from the script.
#daemon Operate as a daemon
# daemon=no
# default-soa-name name to insert in the SOA record if none set in the backend
# default-soa-name=a.misconfigured.powerdns.server
# disable-axfr Disable zonetransfers but do allow TCP queries
disable-axfr=yes
# disable-tcp Listen to TCP queries
disable-tcp=no
# distributor-threads Default number of Distributor (backend) threads to start
distributor-threads=15
#not needed.  fancy-records Process URL and MBOXFW records
# fancy-records
#This is also specified in the startup script.  The value below will have no effect
#unless the startup parameter is removed from the script.
# guardian Run within a guardian process
# guardian=no
# lazy-recursion Only recurse if question cannot be answered locally
lazy-recursion=no
#not needed.  load-modules Load this module - supply absolute or relative path
# load-modules=
#not needed.  local-address Local IP address to which we bind
# local-address=
#not needed.  local-ipv6 Local IP address to which we bind
# local-ipv6=
# local-port The port on which we listen
local-port=53
#not needed.  log-dns-details If PDNS should log dns details
log-dns-details=off
# log-failed-updates If PDNS should log failed update requests
log-failed-updates=off
#not needed.  I use syslog instead.  logfile Logfile to use
# logfile=pdns.log
# logging-facility Log under a specific facility
logging-facility=daemon
# loglevel Amount of logging.  Higher is more.  Do not set below 3
loglevel=10
# master do not act as a master
master=no
# max-queue-length Maximum queuelength before considering situation lost
max-queue-length=5000
# max-tcp-connections Maximum number of TCP connections
```

```
max-tcp-connections=10
# module-dir Default directory for modules
module-dir=/no/where
# negquery-cache-ttl Seconds to store packets in the PacketCache
negquery-cache-ttl=60
# out-of-zone-additional-processing Do out of zone additional processing
out-of-zone-additional-processing=no
# query-cache-ttl Seconds to store packets in the PacketCache
query-cache-ttl=20
# query-logging Hint backends that queries should be logged
query-logging=no
# queue-limit Maximum number of milliseconds to queue a query
queue-limit=1500
# receiver-threads Number of receiver threads to launch
# receiver-threads=1
# recursive-cache-ttl Seconds to store packets in the PacketCache
recursive-cache-ttl=10
# recursor If recursion is desired, IP address of a recursing nameserver
recursor=no
# skip-cname Do not perform CNAME indirection for each query
skip-cname=no
# slave Act as a slave
slave=no
#not needed.  slave-cycle-interval Reschedule failed SOA serial checks once every ..  seconds
# slave-cycle-interval=60
#not needed as I do not use MBOXFW records.  smtpredirector Our smtpredir MX host
# smtpredirector=
# soa-minimum-ttl Default SOA mininum ttl
soa-minimum-ttl=40001
# soa-serial-offset Make sure that no SOA serial is less than this number
# REMEMBER TO CHANGE
# soa-serial-offset=0
# socket-dir Where the controlsocket will live
# socket-dir=/opt/pdns/var
# strict-rfc-axfrs Perform strictly rfc compliant axfrs (very slow)
strict-rfc-axfrs=no
#not needed as I do not use URL records.  urlredirector Where we send hosts to that need to be url redirected
# urlredirector=
# use-logfile Use a log file
use-logfile=no
#do not launch webserver to show logs and queries.
webserver=no
#not needed.  webserver-address IP Address of webserver to listen on
```

```
# webserver-address=
#not needed.  webserver-password Password required for accessing the webserver
# webserver-password=
#not needed.  webserver-port Port of webserver to listen on
# webserver-port=8081
#not needed.  webserver-print-arguments If the webserver should print arguments
# webserver-print-arguments=no
#not needed.  wildcard-url only used in URL and MBOXFW records
# wildcard-url=yes
# wildcards Honor wildcards in the database
wildcards
```

7.19 pdns.sql

```
create database pdns;
use pdns;
create table domains (
id INT auto_increment,
name VARCHAR(255) NOT NULL,
master VARCHAR(20) DEFAULT NULL,
last_check INT DEFAULT NULL,
type VARCHAR(6) NOT NULL,
notified_serial INT DEFAULT NULL,
account VARCHAR(40) DEFAULT NULL,
primary key (id)
);
CREATE UNIQUE INDEX name_index ON domains(name);
CREATE TABLE records (
id INT auto_increment,
domain_id INT DEFAULT NULL,
name VARCHAR(255) DEFAULT NULL,
type VARCHAR(6) DEFAULT NULL,
content VARCHAR(255) DEFAULT NULL,
ttl INT DEFAULT NULL,
prio INT DEFAULT NULL,
change_date INT DEFAULT NULL,
primary key(id)
);
CREATE INDEX rec_name_index ON records(name);
CREATE INDEX nametype_index ON records(name,type);
CREATE INDEX domain_id ON records(domain_id);
create table supermasters (
ip VARCHAR(25) NOT NULL,
nameserver VARCHAR(255) NOT NULL,
account VARCHAR(40) DEFAULT NULL
);
GRANT SELECT ON supermasters TO pdns;
GRANT ALL ON domains TO pdns;
GRANT ALL ON records TO pdns;
```

7.20 test.com.sql

```
INSERT INTO domains (name, type)
VALUES ('test.com', 'NATIVE');
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','localhost ahu@ds9a.nl 1','SOA',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','dns-us1.powerdns.net','NS',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','dns-eu1.powerdns.net','NS',86400,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'www.test.com','199.198.197.196','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'mail.test.com','195.194.193.192','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'localhost.test.com','127.0.0.1','A',120,NULL);
INSERT INTO records (domain_id, name, content, type,ttl,prio)
VALUES (1,'test.com','mail.test.com','MX',120,25);
```