Global Information Assurance Certification Paper

Hardening Debian 4.0 – Creating a simple and solid foundation for your applications

GSEC Gold Certification

Author: Alexandre Déry, alexandre/dot/dery/at/gmail/dot/com

Advisor: Richard Genova, rgenova/at/securewindows/dot/biz

Accepted: August 2nd 2007

Updated: February 3rd, 2009

Outline

1 Introduction

Any operating system is vulnerable to attacks if it's not properly configured. People get really emotional about the security of their preferred operating system: every mildly technical forum is bound to be a battle ground for flame wars between OS lovers. But the bottom line is: company politics and policies aside, whatever the operating system is, its security depends mainly on the knowledge of its administrator. Debate all you want, but even an OpenBSD server will be hacked if its administrator has no clue!

GNU/Linux servers are really popular these days, because they are free, often touted as “much more secure”, and they boast an enthusiast community willing to help out. The problem with this is that it's possible to set up a Linux server with practically no knowledge! The story is typical. New kid at the company is asked by his boss: “Hey Eric, you know Linux right? Could you go ahead and set up a PHP server on the internet for our new website we just had developed? Thanks!” Eric reads a few “howtos” on the net, and after a few hours, manages to have a Linux server with Apache and PHP ready to go! “Job done boss!” he says, going back to his VB code, his real assignment. I do not need to tell you what happens next...

Many of these “howtos” found on the Internet aren't general enough, too often focused on the application to be hosted. I believe that the key to securing servers is to have *a secure foundation* that

you can trust to host all your other applications. That foundation is of course the operating system, be it Windows, GNU/Linux or BSD. In this paper, I will be describing how to install a secure and simple Debian 4.0 system that will happily host whatever you want to throw at it: DNS, DHCP, Web, Database, etc... I choose to use the Debian distribution because of its good reputation, great package management system and rock hard stability which makes it an excellent choice for servers.

We will learn how to install a minimal Debian GNU/Linux 4.0 operating system (codenamed “Etch”, currently the stable branch), remove unnecessary services, replace software with secure alternatives, secure SSH, address time synchronization, keep up with patches, use “sudo” for granular access, protect the boot loader and install a firewall. All these tasks will be done using software provided by Debian (no compilation needed), and all modifications to the system will be done “the Debian way”.

The target audience for this paper are mildly Unix-savvy persons, all the Erics of this world, who are looking for a recipe to lock-down a Debian server, but do not have the time, nor the need, for hardcore kernel settings and custom application patches. It is aimed at the general less-than-ten-servers shop, not the three-hundred-nodes-web-farm business.

2 Requirements

Here is a list of things you will need to successfully follow this cookbook:

- ✓ A fast connection to the internet to download the Debian 4.0 ISO and to download subsequent updates and software;
- ✓ A CD burner and an empty CD-R to burn the ISO image;
- ✓ A SSH identity (SSH key) because password based login will not be accepted. Use this command to generate one and follow the instructions (use a strong pass **phrase**!!!). *(If you've done this on a Debian/Ubuntu based system before May 13th 2008, please discontinue using that key, as it may be vulnerable. See : http://www.debian.org/security/2008/dsa-1571 for details)*.

```
$ ssh-keygen -b 2048 -t rsa -C "Your Name <your.email@example.domain>"
```

- ✓ Your server which should have a network card, one hard disk, video card, monitor and keyboard;

3 Information gathering

The following tables contain some information that you need to have *before* you begin installing the system. The values that are used here *must be replaced* by valid values for your network. For instance, the server name "serveur" and the desktop name "client" must be changed to match your environment. Same thing goes for IP addresses.

Networking settings

Item	Value
IP Address	192.168.2.10
Subnet mask	255.255.255.0
Gateway	192.168.2.1
DNS Server	192.168.2.5
Server name	serveur
Domain name	domain.example

Table 1 – Networking

Disk partitions

A good partition scheme is key to the performance and the security of a system. The subject could be the basis for a paper of its own, but we'll try to get the basics right while leaving room for additional improvements. The main idea is to separate the file system into small task-oriented chunks, giving us the power to secure them in different ways, because the data they'll contain requires different approaches. The following table depicts a sample configuration for a server with a single 30gig disk. Please adjust these values to suit *your needs*:

	Disk	Size	Type	Location	Use as	Mount point	Bootable flag
/ (root)	sda	1 GB	Primary	Beginning	Ext3	/	On
swap	sda	1 GB	Primary	Beginning	swap	n. a.	Off
/usr	sda	2 GB	Logical	Beginning	Ext3	/usr	Off
/tmp	sda	1 GB	Logical	Beginning	Ext3	/tmp	Off
/var	sda	10 GB	Logical	Beginning	Ext3	/var	Off
/srv	sda	10 GB	Logical	Beginning	Ext3	/srv	Off
/home	sda	5 GB	Logical	Beginning	Ext3	/home	Off

Table 2 - Partitions

We need to separate the server's data from the operating system. Why? If an application misbehaves and creates a lot of data or some hacker fills up your logs with garbage, your disk will clog up, and the operating system will crawl down to a halt! By separating the logs (/var) and the data (/srv, /home) from the rest of the OS (/, /usr/, etc…), you are making your system more resilient against such problems. You can find more information about this on the internet, in documents such as the Filesystem Hierarchy Standard [2].

Mail server

Your new server will send its outgoing messages through another SMTP server, which can be yours or your ISP's:

Server	DNS name or IP address
my SMTP server	smtphost.example.domain

Table 3 – Mail

Accounts

Who will need access to the server? You need to find out who *really* needs it, and if they do, what they are allowed to do. Here I have 3 sample accounts. Alex is the administrator and as such is allowed to do everything as root using the sudo command. Joe is a simple DBA and doesn't need any special UNIX privileges to do his work. Bob is the coder and he needs tcpdump to troubleshoot his network application. Using sudo is a great way to give users the rights to execute one particular piece of software as root, without having full root access on the server. Be careful not to give root access to a script that the user can edit, or to a program that can provide a shell to the user (like the VI editor).

Login	Name	Groups	Sudo
alex	Alexandre Dery	adm	full
joe	Joe Bine	none	no

bob	Bob Inno	none	/usr/sbin/tcpdump

Table 4 – Account

4 Installation

Hardware configuration

Physically prepare the server for installation. If it's a brand name server (like HP, Dell, IBM or others), you might need to run a preparation CD that configures the system for your operating system before proceeding with the installation of the OS. Please refer to your manuals for directions.

The installation of some packages will require access to the internet. Since the server will be vulnerable before all patches are installed, it is recommended that you attach the server to a network which is already protected by a firewall. This way, hackers won't get to your server before you finish installing it! When the installation is over, you can then move the server to its real network.

From your desktop, you need to download and burn the Debian "netinstall" CD image. Visit http://www.debian.org/CD/netinst/ and download the appropriate image for your hardware (i386 is the most common).

Beginning of installation

Unplug the network cable;

Insert the CD you just burned in the drive and boot it;

The Debian logo appears with the following prompt :

```
Press F1 for help, or ENTER to boot: [PRESS ENTER]
```

- The kernel is loaded and the installer is started...
- Choose language - Choose a language: **English - English**
  (I choose to install my servers in English by default, because it makes searching for error messages much easier. Later in the installation, I'll show how to manually install language packages for your language.)
- Choose language - Select a country, territory or area: **Canada** (select your own country)
- Select a keyboard layout - Keymap to use: **American English** (or choose whichever you prefer)
- The installer detects your hardware...

Network configuration

- If you have more than one network interface, the installer will ask this question, and you will need to choose which one to use:
  - Configure the network - Primary network interface: **(choose the right card)**
- Configure the network - Network autoconfiguration failed (because the network cable isn't plugged in): **Continue**
- Configure the network - Network configuration method: **Configure**

network manually

- Now you may plug a cable in the network interface;
- Configure the network - IP address: [Networking:**IP Address**]
- Configure the network - Netmask: [Networking:**Subnet mask**]
- Configure the network - Gateway: [Networking:**Gateway**]
- Configure the network - Name server addresses: [Networking:**DNS Server**]
- Configure the network - Is this information correct?: **Yes**
- Configure the network - Hostname: [Networking:**Server Name**]
- Configure the network - Domain name: [Networking:**Domain Name**]

Disk configuration

- Partition disks - Partitioning method : select **Manual**
- Partition disks: Your hard disks are listed. Their names will differ depending on the types of controller you have (ide, scsi, raid, etc...).
- For every disk that doesn't have a **FREE SPACE** tag underneath:
  - Select the disk and press [ENTER]
  - Partition disks - Create new empty partition table on this device? : **Yes**

Repeat these steps for every line in the [Partition] table:

- Under [Partition:**Disk**], select **FREE SPACE**
- Partition disks - How to use this free space: **Create a new partition**
- Partition disks - New partition size: [Partition:**Size**]
- Partition disks - Type for the new partition : [Partition:**Type**]
- Partition disks - Location for the new partition : [Partition:**Location**]

- Partition disks - Partition settings :
  - Use as : Choose [Partition:**Use as**]
  - Mount point : [Partition:**Mount point**]
  - Bootable flag : set to [Partition:**Bootable flag**]
  - Select **Done setting up the partition**
- When all partitions are created, select **Finish partitioning and write changes to disk**;
- Partition disks - Write the changes to disk? **Yes**
- Partitions formatting: **wait...**
- Configure time zone - Select your time zone : **Eastern** (select yours)
- Set up users and passwords - Root password : **enter a secure password**
- Set up users and passwords - Re-enter password to verify : **confirm the password**
- Set up users and passwords - Full name for the new user : **System Operator** (or you could use something more obscure)
- Set up users and passwords - Username for your account : **sysop** (ditto)
- Set up users and passwords - Choose a password for the new user : **enter another secure password**
- Set up users and passwords - Re-enter password to verify : **confirm the other secure password**
- Installing the base system...
- Configure the package manager - Use a network mirror? **Yes**
- Configure the package manager - Debian archive mirror country : **Select your country, mine is Canada**

- Configure the package manager - Debian archive mirror : **Select a mirror close to you, for me it's gulus.usherbrooke.ca**
- Configure the package manager - HTTP proxy information : **enter your proxy server here if you have one or press [ENTER]**
- Configuring apt - Scanning the mirror...
  Here, the installer is downloading the database of software available on the mirror (basically apt-get update).
- Select and install software...
- Configuring popularity-contest - Participate in the package usage survey : **No**

  This is where you choose your minimal system. There is no "Core/Minimal system" choice (that would be too obvious I guess), so what you need to do is uncheck every option: this will result in the most basic system the interactive installer is able to provide.
- Software selection - Choose software to install : **UNSELECT ALL CHOICES** and then **Continue**
- Install the GRUB boot loader on a hard disk - Install the GRUB boot loader to the master boot record? **Yes**
- Finish the installation - Installation complete : **Continue**
- The server restarts and Debian boots for the first time...

First connection to the server

- Let's connect to our new server with the root password we specified earlier :

```
Debian GNU/Linux 4.0 serveur tty1
```

```
serveur login: root
Password: [root password specified earlier]
Linux serveur 2.6.18...
[...]
serveur:~#
```

Configuring the APT system

The APT system is the collection of utilities that manages the ".deb" packages that make up the operating system. By default, our package "source" is the CDROM, which is no good since it's a minimal CD. We want our package source to be the Debian internet repository we selected earlier.

Let's deactivate the CDROM as a package source:

```
serveur:~# vi /etc/apt/sources.list
```

There are two "*deb cdrom*" lines: remove the second one (the first one is already commented out. The file should end up looking like this (but your http mirrors will be different):

```
#
# deb cdrom:[Debian GNU/Linux 4.0 r1 _Etch_ - Official i386 NETINST Binary-1
20070820-20:21]/ etch contrib main


deb http://gulus.usherbrooke.ca/debian/ etch main
deb-src http://gulus.usherbrooke.ca/debian/ etch main


deb http://security.debian.org/ etch/updates main contrib
deb-src http://security.debian.org/ etch/updates main contrib
```

Installing the latest patches

Since there are security updates that have been published after the creation of the installation CD, we need to update our server

before going any further. If we don't do so, we could end up installing old versions of packages, because the APT system wouldn't be aware of the newer versions that are available. After the update, we will reboot the server, because of kernel upgrades.

The commands to update a Debian system are “apt-get update”, which updates the APT database of packages and security updates, followed by “apt-get upgrade” which installs the available updates. The “apt-get dist-upgrade” command is very useful when more complex upgrades are needed. For instance, if a package-A update needs the installation of package-Z for dependency reasons, “apt-get update” won't be able to proceed because package-Z isn't already installed, and will say that the package-A update has been “held back”. When this happens, you need to use the “apt-get dist-upgrade” command, which has the ability to deal with package dependency problems, and will install package-Z before updating package-A. The “apt-get dist-upgrade” command can also be used to upgrade your distribution (thus the name) for example, from “etch” (4.0) to “lenny” (5.0, still unreleased as of this update!).

```
serveur:~# apt-get update
[...]
serveur:~# apt-get dist-upgrade
Reading package lists... Done
Building dependency tree... Done
The following packages will be upgraded:
 libssl0.9.8 linux-image-2.6.686 linux-image-2.6.18-5-686 perl-base
 vim-common vim-tiny
9 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 20.5MB of archives.
After unpacking 221kB disk space will be freed.
Do you want to continue [Y/n]? y
```

```
      [...]
      The installer might ask you some questions about the update, so read
carefully and answer the best you can! The default choices are often the correct
ones. For instance, after a kernel update, you are very strongly suggested to
reboot immediately.
      [...]
      serveur:~# reboot
      [...]
```

5 Configuring OpenSSH

There will be only two ways to access our server: remotely with SSH keys, and locally at the physical console with a simple UNIX password. We will display a warning banner in both cases. The message is short and simple; otherwise nobody would read/understand it. This message is the one suggested in the UNIX book of the GSEC curriculum. If English is not your native language, I recommend displaying the warning in both your language and in English, so it's understandable by everybody (including attackers). I went a step further and chose to write the French version without extended characters (plain ASCII 7 bit chars) to be sure the text isn't littered with garbage characters and whatnot. This might not be possible for some languages so the choice is yours.

Installing the ssh server and client

```
      serveur:~# apt-get install openssh-client openssh-server
      Reading package lists...
      Building dependency tree...
      The following extra packages will be installed:
        libedit2 libkrb53
      Suggested packages:
```

```
  krb5-doc krb5-user ssh-askpass xbase-clients rssh molly-guard
The following NEW packages will be installed:
  libedit2 libkrb53 openssh-client openssh-server
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 1301kB of archives.
After unpacking 3301kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Preconfiguring packages ...
[...]
Setting up openssh-client (4.3p2-9) ...

Setting up openssh-server (4.3p2-9) ...
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
NET: Registered protocol family 10
lo: Disabled  PrivacySSH server configuration Extensions
IPv6 over IPv4 tunneling driver
Restarting OpenBSD Secure Shell server: sshd.
serveur:~#
```

Let's display the hash of our server's ssh public key:

```
serveur:~# ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
2048 44:c3:7c:1e:0c:f6:24:82:2f:b7:f8:83:93:1f:08:13
/etc/ssh/ssh_host_rsa_key.pub
```

First SSH connection to the server

We'll connect using SSH and the “sysop” user we created earlier. Make sure that the hash shown upon connection is identical to the one we displayed in the previous step!

```
alex@client:~$ ssh sysop@serveur (or use the IP address of the server if
“serveur” isn't included in your /etc/hosts file or your DNS server)
The authenticity of host 'serveur (192.168.2.10)' can't be established.
```

```
RSA key fingerprint is 44:c3:7c:1e:0c:f6:24:82:2f:b7:f8:83:93:1f:08:13.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.10' (RSA) to the list of known hosts.
sysop@serveur's password: [enter sysop's password]
Linux serveur 2.6.18-4-686 #1 SMP Wed May 9 23:03:12 UTC 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
sysop@serveur:~$
```

From here on, you can complete the installation via SSH, with the “sysop” user, using “su” to get to root. I strongly recommended this, if only because pasting commands is so much faster than typing them manually... and the server room is cold!

Copy your SSH public key to the server

Since we will configure SSH to accept only key-based authentication (more on that later), we need to copy our public key to the “sysop” account now, to prevent being locked out of remote access.

```
sysop@serveur:~$ mkdir .ssh
sysop@serveur:~$ chown sysop:sysop .ssh
sysop@serveur:~$ chmod 700 .ssh
sysop@serveur:~$ exit
logout
Connection to serveur closed.
Substitute “.ssh/id_rsa.pub” with the path to your ssh public key file.
alex@client:~$ scp .ssh/id_rsa.pub sysop@serveur:.ssh/authorized_keys
```

```
sysop@serveur's password:
id_rsa.pub                                    100%  431     0.4KB/s   00:00
```

Connection test with the public key:

```
alex@client:~$ ssh sysop@serveur
Enter passphrase for key '/home/alex/.ssh/id_rsa': [enter passphrase]
Linux serveur 2.6.18-4-686 #1 SMP Wed May 9 23:03:12 UTC 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu May 17 11:44:15 2007 from client.example.domain
sysop@serveur:~$
```

Saving the server's SSH fingerprint

It's good practice to sign and store the hashes of the public keys of your servers somewhere. This way when you or your users connect to the server for the first time, they can verify the hash against the “trusted list” instead of blindly answering “yes”. For this example, I'll save *the DSA and* RSA hashes to a local file, sign that file using GnuPG and copy it to a file share located on another server, but remember there are many ways to build this “trusted list”.

```
alex@client:~$ ssh sysop@serveur ssh-keygen -l -f
/etc/ssh/ssh_host_rsa_key.pub > serveur_ssh_fingerprints.txt
alex@client:~$ ssh sysop@serveur ssh-keygen -l -f
/etc/ssh/ssh_host_dsa_key.pub >> serveur_ssh_fingerprints.txt
alex@client:~$ gpg -bs serveur_ssh_fingerprints.txt
```

```
      [...GnuPG details removed...]
      alex@client:~$ scp serveur_ssh_fingerprints.txt*
alex@otherserver:/srv/fileshare/keys/ssh/
      serveur_ssh_fingerprints.txt      100%  166     0.2KB/s   00:00
      serveur_ssh_fingerprints.txt.sig  100%   65     0.1KB/s   00:00
      alex@client:~$
```

Warning banner configuration

Log to the server, become the “root” user and edit “/etc/issue” to replace its content with this:

```
      sysop@serveur:~$ su -
      Password:
      serveur:~# vi /etc/issue
```

```
      **********Warning**********

      Authorized uses only.
      All activity may be monitored and reported.

      ************************************************
```

and here's a French-English version:

```
      ************Avertissement / Warning************

      Utilisations autorisees seulement.
      Toute activite peut etre surveillee et signalee.

      Authorized uses only.
      All activity may be monitored and reported.

      ************************************************
```

Edit *"/etc/pam.d/ssh"* and turn off the "message of the day (motd)" feature. We do this to make sure only our warning banner is displayed, and nothing else.

```
# Print the message of the day upon successful login.
#session    optional     pam_motd.so # [1]
```

Edit *"/etc/pam.d/login"* and turn off the "motd":

```
# Prints the motd upon successful login
# (Replaces the `MOTD_FILE' option in login.defs)
#session    optional   pam_motd.so
```

SSH server configuration

We will now tighten the SSH server's security. First we'll force it to listen only on *one* specific ipv4 address, instead of every address we (may) have on the server. We refuse direct root logins, because we want people to log in to their own account, and then use sudo or "su" to get the access they need.

We also disable password authentication, which means that the only way to authenticate to the SSH server will be with an SSH identity (public key), thus yielding two benefits. First, if your users put their SSH private keys on a USB key chain, you end up with a cheap (as in non-expensive) 3-factor authentication system! Second, it blocks all the automated SSH password guessing attacks, since password authentication simply isn't allowed. We then disable both X11 and TCP port forwarding, and activate the warning banner.

Edit the ssh server configuration file "*/etc/ssh/sshd_config*" and do the following modifications:

```
# Package generated configuration file
# See the sshd(8) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
ListenAddress 192.168.2.10
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
#PermitRootLogin yes
PermitRootLogin no
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys
```

```
      # Don't read the user's ~/.rhosts and ~/.shosts files
      IgnoreRhosts yes
      # For this to work you will also need host keys in /etc/ssh_known_hosts
      RhostsRSAAuthentication no
      # similar for protocol version 2
      HostbasedAuthentication no
      # Uncomment if you don't trust ~/.ssh/known_hosts for
RhostsRSAAuthentication
      #IgnoreUserKnownHosts yes

      # To enable empty passwords, change to yes (NOT RECOMMENDED)
      PermitEmptyPasswords no

      # Change to yes to enable challenge-response passwords (beware issues with
      # some PAM modules and threads)
      ChallengeResponseAuthentication no

      # Change to no to disable tunnelled clear text passwords
      #PasswordAuthentication yes
      PasswordAuthentication no

      # Kerberos options
      #KerberosAuthentication no
      #KerberosGetAFSToken no
      #KerberosOrLocalPasswd yes
      #KerberosTicketCleanup yes

      # GSSAPI options
      #GSSAPIAuthentication no
      #GSSAPICleanupCredentials yes

      # Deactivate port forwarding
      AllowTcpForwarding no

      #X11Forwarding yes
```

```
X11Forwarding no
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net
Banner /etc/issue

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

UsePAM yes
```

Restart the SSH server:

```
serveur:~# /etc/init.d/ssh restart
Restarting OpenBSD Secure Shell server: sshd.
serveur:~#
```

We logout and connect back. The new warning banner should appear. If you already have one, please empty the cache of your SSH agent.

```
serveur:~# exit
logout
sysop@serveur:~$ exit
logout
Connection to serveur closed.
alex@client:~$ ssh sysop@serveur
**********Warning**********
```

```
Authorized uses only.
All activity may be monitored and reported.

************************************************

Enter passphrase for key '/home/alex/.ssh/id_rsa': [enter your passphrase]
Last login: Thu May 10 13:50:22 2007 from client.example.domain
sysop@serveur:~$ exit
logout
Connection to serveur closed.
```

Let's make sure that password authentication is disabled (again, empty your SSH agent's cache if you have one):

```
alex@client:~$ ssh sysop@serveur
*********************Warning*********************

Authorized uses only.
All activity may be monitored and reported.

************************************************

Enter passphrase for key '/home/alex/.ssh/id_rsa': [enter]
Permission denied (publickey).
alex@client:~$
```

The authentication process didn't fall back to “password” authentication, as expected.

6 IP Configuration

Ethernet interfaces on servers are in no way “hot-pluggable” so we do the following modification in the network interfaces configuration file “/etc/network/interfaces”:

```
# The primary network interface
#allow-hotplug eth0
auto eth0
iface eth0 inet static
...
```

7 Removing unnecessary software

Since we have installed a pretty bare system, there is not much to uninstall. Currently we can't remove “openbsd-inetd” or “tcpd” because the package “netbase” (wrongly) depends on them, so we'll simply deactivate “inetd”. Sysklogd and klogd are removed and replaced by Syslog-NG, which offers a more flexible configuration. Here are the packages we'll remove:

- acpid: Power saving daemon
- dhcp3-common : Common files for DHCP client
- dhcp3-client : DHCP client
- sysklogd : Default syslog daemon
- klogd : Kernel message logger

Let's remove these packages, using the “--purge” argument, which forces all files (even configuration files) to be removed:

```
serveur:~# apt-get remove --purge acpid dhcp3-common dhcp3-client klogd
sysklogd
Reading package lists... Done
Building dependency tree... Done
The following packages will be REMOVED:
  acpid* dhcp3-client* dhcp3-common* klogd* sysklogd*
0 upgraded, 0 newly installed, 5 to remove and 0 not upgraded.
Need to get 0B of archives.
After unpacking 1778kB disk space will be freed.
Do you want to continue [Y/n]? y
```

```
(Reading database ... 13162 files and directories currently installed.)
Removing acpid ...
Stopping Advanced Configuration and Power Interface daemon: acpid.
Purging configuration files for acpid ...
Removing dhcp3-client ...
Purging configuration files for dhcp3-client ...
Removing dhcp3-common ...
Removing klogd ...
Stopping kernel log daemon: klogd.
Purging configuration files for klogd ...
Removing sysklogd ...
Stopping system log daemon: syslogd.
Purging configuration files for sysklogd ...
```

Leftover file...

```
serveur:~# rm /var/log/acpid
```

Let's stop and deactivate “openbsd-inetd” by removing any startup links pointing to it. While this could be done manually, Debian provides the command “update-rc.d” to do just that:

```
serveur:~# /etc/init.d/openbsd-inetd stop
Stopping internet superserver: inetd.
serveur:~# update-rc.d -f openbsd-inetd remove
 Removing any system startup links for /etc/init.d/openbsd-inetd ...
   /etc/rc0.d/K20openbsd-inetd
   /etc/rc1.d/K20openbsd-inetd
   /etc/rc2.d/S20openbsd-inetd
   /etc/rc3.d/S20openbsd-inetd
   /etc/rc4.d/S20openbsd-inetd
   /etc/rc5.d/S20openbsd-inetd
   /etc/rc6.d/K20openbsd-inetd
serveur:~#
```

8 Installing some tools

Here is a list of tools that I find handy to have on a server on a day to day basis. You may want to alter this list to suit your needs, but for every tool you add, ask yourself this question: “Do I really need this tool on ALL my servers?” If the answer is “Yes”, then it goes on the list. Remember that everything on your server could be used against you (by a rogue user for instance), so the less junk on the server the better.

- apt-show-versions : Lists what packages can be upgraded
- dnsutils : DNS client tools such as *dig* and *nslookup*
- ethtool : Configure speed and duplex of an Ethernet card
- file : Helps to determine the contents of a file
- less : Because less is more :)
- mailx : Simple local mail reader
- nullmailer : Lightweight outgoing mail daemon
- ntpdate : Local clock synchronization
- perl : Ubiquitous script language
- sudo : Implements granular “root” access
- syslog-ng : Modern replacement for sysklogd and klogd
- tcpdump : Really useful to troubleshoot network problems
- unzip : Decompress ZIP archives
- zip : Creates ZIP archives

```
serveur:~# apt-get install apt-show-versions dnsutils ethtool file less mailx nullmailer ntpdate perl sudo syslog-ng tcpdump unzip zip
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  bind9-host libapt-pkg-perl libbind9-0 libdns22 libisc11 libiscc0 libisccfg1
```

```
        liblockfile1 liblwres9 libmagic1 libpcap0.8 libpcre3 perl-modules
      Suggested packages:
        rblcheck libterm-readline-gnu-perl libterm-readline-perl-perl
      Recommended packages:
        sysklogd system-log-daemon perl-doc
      The following NEW packages will be installed:
        apt-show-versions bind9-host dnsutils ethtool file less libapt-pkg-perl
libbind9-0
        libdns22 libisc11 libisccc0 libisccfg1 liblockfile1 liblwres9 libmagic1
libpcap0.8
        libpcre3 mailx ntpdate nullmailer perl perl-modules sudo syslog-ng tcpdump
unzip
        zip
      0 upgraded, 27 newly installed, 0 to remove and 0 not upgraded.
      Need to get 9261kB of archives.
      After unpacking 35.4MB of additional disk space will be used.
      Do you want to continue [Y/n]? y
      [...]
```

Configuration of Nullmailer

A common misconception in UNIX-land is that you need a full-fledged mail transport agent (Sendmail, Postfix...) to enable your server to send outgoing mail (warnings and such). Not only is this false, but it's also a big security risk. Mail servers are an easy target because they need root privileges just to listen on port 25, and they commonly boast an impressive history of security flaws. For an attacker, a vulnerable SMTP daemon is like a key underneath a welcome doormat.

Nullmailer is a small daemon that is tailored to send outgoing mail to a central SMTP server (also called a smart host). It's a tiny piece of software that doesn't even need to listen on port 25 (this is better than Exim4, the default Debian mail handler, which needs to

listen on port 25 of the loopback interface at minimum). To complete its installation, you will be asked for the fully qualified name of your server, and the hostnames or IP addresses of mail servers that will accept mail from your server (you've defined this at the start of the document, right?):

- Configuring nullmailer - Mailname of your system: **serveur.domain.example** (complete name of the server).
- Configuring nullmailer - Smarthosts : **smtphost.domain.example**

9 Configuring file system restrictions

Now is the time to apply some additional security restrictions to some of our partitions. There are many combinations of security flags that we can set on any partition (noexec, nosuid, read-only, nodev), but it can get pretty specific depending on the use of the server, so we'll configure a basic one as an example. “Set-UID” binaries are executables that run with the privileges of their owner. If a binary file has the “setuid bit” set and it's owned by root, it will run with root's privileges. If a rogue user manages to install a “rogue setuid root binary” in its home folder, he has effectively become root! Here's what such a binary could look like:

```
-rwsrwxrwx 1 root rogue 54 2007-12-13 14:30 /home/rogue/evil
```

To prevent that, let's add the “nosuid” option to the /home and /tmp partitions, to prevent the execution of binaries with high privileges. As root, edit the file “/etc/fstab”, and add the “,nosuid” option to the /home and /tmp file systems:

```
# /etc/fstab: static file system information.
#
```

```
[...]
/dev/ida/c1d1p3 /home          ext3    defaults,nosuid       0       2
/dev/ida/c1d1p1 /srv           ext3    defaults       0       2
/dev/ida/c1d1p2 /tmp           ext3    defaults,nosuid       0       2
[...]
```

Now let's “remount” those file systems to activate the changes:

```
root@serveur:~# mount -o remount /tmp
root@serveur:~# mount -o remount /home
```

Let's verify our changes:

```
root@serveur:~# mount
[...]
/dev/ida/c1d1p3 on /home type ext3 (rw,nosuid)
/dev/ida/c1d1p2 on /tmp type ext3 (rw,nosuid)
[...]
```

10 Dealing with multiple languages

Debian is translated in many languages, and yours is probably included. Even though the French translation of Debian is complete and well done, I choose to install my *servers* in English by default. Why? When you're facing an error message that you don't know how to solve, you'll have much more results in your favorite search engine when searching for the English message than the translated one.

Now, this is *my opinion*, but other users and administrators may not care about that and still want the system translated. Your employer (this was my case) may also force you to install the system in your local language for reasons they consider valid. How do you solve this problem? Simple, just install the system in English, and

then add the libraries for your local language. This way, the system will default to English, but can be switched to your language, on a per-user basis, with only one line in a user's shell profile.

For instance, here are the packages for the French libraries:

- doc-debian-fr
- doc-linux-fr-text
- manpages-fr
- manpages-fr-dev
- manpages-fr-extra
- language-env

Now you may ask yourself, how do I find out which libraries I need for my particular language? Simple! Perform a basic English install of Debian on a spare machine (or using a tool such as VmWare), and then run the following command on it:

```
# dpkg --get-selections > english.txt
```

Save the newly created file. Then, perform another basic installation but select your language (ex: Korean), and also list the installed packages:

```
# dpkg --get-selections > korean.txt
```

And then compare those two files using *diff* or some other file comparison tool to find out what are the packages needed for your particular language. Voilà!

Installation of language libraries

```
serveur:~# apt-get install doc-debian-fr doc-linux-fr-text manpages-fr
manpages-fr-dev manpages-fr-extra language-env
Reading package lists... Done
Building dependency tree... Done
Suggested packages:
  doc-linux-fr-html
Recommended packages:
  developers-reference-fr maint-guide-fr apt-howto-fr ncurses-term wish
The following NEW packages will be installed:
  doc-debian-fr doc-linux-fr-text language-env manpages-fr manpages-fr-dev
manpages-fr-extra
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 8082kB of archives.
After unpacking 13.4MB of additional disk space will be used.
[...]
Setting up manpages-fr (2.39.1-5) ...
```

We need to activate these libraries:

```
serveur:~# dpkg-reconfigure locales
```

- A menu will appear :
  - Configuring locales - Locales to be generated: *Select those two for English/French* system (for a language other than French, choose accordingly) : **en_CA.UTF-8 UTF-8** and **fr_CA.UTF-8 UTF-8** an then **OK**
  - Configuring locales - Default locale for the system environment: select **en_CA.UTF-8** and then **OK**

Back to the console:

```
Generating locales (this might take a while)...
  en_CA.UTF-8... done
```

```
  fr_CA.UTF-8... done
Generation complete.
```

Testing the language libraries

Let's test the French libraries:

```
serveur:~# man women
No manual entry for women
serveur:~# LANG=fr_CA.UTF-8 man les_femmes
Aucune entrée de manuel pour les_femmes
serveur:~#
```

The system can't find any manual entry for women, either in French or in English, so we know everything is working! (My apologies to the ladies, I couldn't resist!).

Sample configuration for a non-English user

All that is needed to switch a user to another language is to add two lines to that user's *".bash_profile", as presented bellow:*

```
# ~/.bash_profile: executed by bash(1) for login shells.
[...snip...]

#Je veux mon systeme en Francais, sacrebleu!
LANG=fr_CA.UTF-8
export LANG
```

11 Specifying network card speed

Mismatched network speed or duplex can be a real performance killer. Sometimes, the network card may have trouble negotiating the right speed and duplex settings with its peer (switch, router, other

server, etc...). Some people advise *always* to force those settings, while others prefer to rely on negotiation. I fall in the latter category, and I recommend not forcing settings unless really necessary. So if the negotiated values are wrong, you should first try to see why it is so: there may be an old static configuration for your port in the switch, or your Ethernet cable might be busted, or something else. Let's use the “mii-tool” command to check our interface's settings:

```
root@serveur:~# mii-tool eth0
eth0: negotiated 100baseTx-FD, link ok
root@serveur:~#
```

Here you see the result of a working negotiation that ended up with a 100Mbps speed (100baseTx) and full duplex (FD). If the values aren’t the ones you expect, and you're out of troubleshooting options, you must force the right settings. Here's how you would force the interface “eth0” to 100Mbps full duplex:

Edit “*/etc/network/interfaces*” and add the following line in “eth0”'s configuration section:

```
iface eth0 inet static
     [...]
     up ethtool -s eth0 speed 100 duplex full autoneg off
```

The “up” keyword means that the following command will be executed when the interface comes up. We use the “ethtool” command (that we installed earlier) to force the settings. The “down” keyword also exists, but it’s not needed in this situation. Don't forget to configure the peer with the same settings!

12 Configuring the default editor

If the default editor, “nano”, doesn't suit you, here's how to modify it globally, *the Debian way*:

```
serveur:~# update-alternatives --set editor /usr/bin/vim.tiny
Using `/usr/bin/vim.tiny' to provide `editor'.
```

13 Time Synchronization with NTP

It's really important that the clock(s) of your server(s) be synchronized, to ease the process of comparing logs in case of a break-in, or simply troubleshooting a problem. Some protocols like Kerberos rely heavily on time, so it’s very important that your servers (and clients too) be synchronized. To achieve this goal, we will use the client program “ntpdate”, and schedule it to run every 2 hours. We will use the “Debian-ized” version of “ntpdate” that gets its configuration from the “/etc/default/ntpdate” by default.

Configuring ntpdate

We change the defaults to use the “/etc/default/ntpdate” configuration file and we make sure everything is logged to Syslog. If you have an NTP server in your network, just put its address in the “NTPSERVERS” variable, as shown below. Edit “*/etc/default/ntpdate*” change the following:

```
# The settings in this file are used by the program ntpdate-debian, but not
# by the upstream program ntpdate.

# Set to "yes" to take the server list from /etc/ntp.conf, from package ntp,
# so you only have to keep it in one place.
```

```
      #NTPDATE_USE_NTP_CONF=yes
      NTPDATE_USE_NTP_CONF=no


      # List of NTP servers to use  (Separate multiple servers with spaces.)
      # Not used if NTPDATE_USE_NTP_CONF is yes.
      NTPSERVERS="0.debian.pool.ntp.org 1.debian.pool.ntp.org
2.debian.pool.ntp.org 3.debian.pool.ntp.org"
      # OR IF YOU HAVE YOUR OWN NTP SERVER
      #NTPSERVERS="ntpserver.domain.example 0.debian.pool.ntp.org
1.debian.pool.ntp.org 2.debian.pool.ntp.org 3.debian.pool.ntp.org"


      # Additional options to pass to ntpdate
      #NTPOPTIONS=""
      #The -s means “silent operations”, i.e., no console output, write to syslog.
      NTPOPTIONS=" -s "
```

Scheduling with CRON

Add the following lines to root’s crontab. The first line is for time synchronization with NTP, and the second saves the time to the hardware clock.

```
      serveur:~# crontab -e

      # m h  dom mon dow   command
      # Time synchronization
      11 */2 * * * /usr/sbin/ntpdate-debian > /dev/null 2>&1
      15 */2 * * * /sbin/hwclock --systohc >/dev/null 2>&1
```

First manual time synchronization

Let's force a manual synchronization to make sure everything works:

```
      serveur:~# date
```

```
      Tue Aug  7 06:59:59 EDT 2007
      serveur:~# /usr/sbin/ntpdate-debian
      serveur:~# date
      Tue Aug  7 11:00:09 EDT 2007
      serveur:~#
```

14 Creating user accounts

Let's create users for people that *really* need access to the server. This'll be easy since you've already made that list!

For every person in the *Accounts* table, do these steps:

```
      serveur:~# adduser [Accounts:Login]
      Adding user [Accounts:Login] ...
      Adding new group [Accounts:Login] (some id > 1000) ...
      Adding new user [Accounts:Login] (some id > 1000) with group
[Accounts:Login] ...
      Creating home directory `/home/[Accounts:Login]' ...
      Copying files from `/etc/skel' ...
      Enter new UNIX password: [enter a secure password for this user]
      Retype new UNIX password: [confirm]
      passwd: password updated successfully
      Changing the user information for [Accounts:Login]
      Enter the new value, or press ENTER for the default
              Full Name []: [Accounts:Name]
              Room Number []: [ENTER]
              Work Phone []: [ENTER]
              Home Phone []: [ENTER]
              Other []: [ENTER]
      Is the information correct? [y/N] y
      serveur:~#
```

Add the new user to its groups with the following command (run once per group):

```
serveur:~# adduser [Accounts:Login] [Accounts:Group]
Adding user [Accounts:Login] to group [Accounts:Group] ...
Done.
```

Configuring SUDO

SUDO is a program that brings granular access delegation to UNIX systems. So instead of the *root-or-nothing* model, SUDO enables the administrator to give a user the right to run “this particular command” as root, without knowing root's password! The file that contains the settings is “/etc/sudoers”, but it MUST be edited through the “visudo” command, which will prevent you from breaking the configuration, thus rendering SUDO unusable. Since SUDO is a really important piece of software, I'll describe three different usage scenarios:

Full access

For each user in the “Accounts” table that has “Yes” in the “Sudo” field, add a line like this in “/etc/sudoers”. This line gives “root” access to the user, so be careful who gets it!

```
root@serveur# visudo

# /etc/sudoers
# User privilege specification
root    ALL=(ALL) ALL
alex   ALL=(ALL) PASSWD: ALL
```

Single command with password

Bob needs to be able to run “tcpdump” (as seen in the “Accounts” table), so let's give him that permission. Note that Bob will have to

type that command “as-is” or else it won't run. Bob will be asked to enter his own password before the command is executed:

```
bob   ALL=(ALL) PASSWD: /usr/sbin/tcpdump -ni eth0
```

Single command without a password

Let's suppose we want the “sysop” user to be able to install system updates, without being prompted for a password (for scripting purposes):

```
sysop   ALL=(ALL) NOPASSWD: /usr/bin/apt-get update
sysop   ALL=(ALL) NOPASSWD: /usr/bin/apt-get upgrade
```

Test

Now let's verify that “sysop” can update the system. Again, please note that the command *must be typed exactly as entered in /etc/sudoers* or else it won't work.

```
serveur:~# su – sysop
sysop@serveur:~$ sudo apt-get update
[update stuff...]
```

15 Disabling reboot on CTRL+ALT+DEL

By default, Linux servers reboot when they receive a CTRL+ALT+DELETE on the console (MS-DOS nostalgia I guess...). I know at least one junior administrator that rebooted a major mail server, thinking he was login on his Windows NT machine... (Okay that was me :)... To prevent surprises, we deactivate this feature and log a message to Syslog and also to the console. Edit *“/etc/inittab”* and modify the following line:

```
      # What to do when CTRL-ALT-DEL is pressed.
      #ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
      ca:12345:ctrlaltdel:/usr/bin/logger -s -p auth.notice -t [INIT]
"CTRL+ALT+DEL caught but ignored! This is not a Windows(r) machine."
```

Force “init” to reload its configuration:

```
      serveur:~# init q
```

You can try the CTRL+ALT+DEL on the physical server console to make sure it doesn't reboot.

16 Protecting GRUB

We'll protect the GRUB boot loader with a password, to prevent people from adding boot parameters that could yield full access. This doesn't offer total protection, but it helps “keeping people honest”. You may also want to modify the boot order on your system (in the BIOS) so that it boots straight to the hard disk, and nothing else. You should also protect the BIOS with a password, or this is a moot point. And please, lock you server room!

Hashing a password for GRUB

For more protection, the password we put in the GRUB configuration is hashed with md5. Here's how to do that step:

```
      serveur:/boot/grub# grub-md5-crypt
      Password: [password to protect GRUB]
      Retype password: [confirm password]
      $1$sqO7z1$abxxxU49wVmFTPaVn/tUt1
      serveur:/boot/grub#
```

Adding a password to the Grub configuration

Edit *"/boot/grub/menu.lst"* and add the following line, using the password hash YOU generated*:*

```
      ## password ['--md5'] passwd
      # If used in the first section of a menu file, disable all interactive
editing
      # control (menu entry editor and command-line)  and entries protected by the
      # command 'lock'
      # e.g. password topsecret
      #      password --md5 $1$gLhU0/X9dhV3P2b2znUoe/
      # password topsecret
      password --md5 $1$sqOj--your-hash-here--fn/tUt1
```

17 Configuring a firewall

Even if your perimeter defenses are top notch, each server should still protect itself. This is called "defense in depth": your security architecture should have more than one layer. Why? If another of your servers is compromised, it can now launch attacks against your other servers which aren't protected anymore. If every server has a firewall that restricts inbound and outbound traffic, it will be more resilient against internal attacks, and may also prevent it from becoming a launch pad for other attacks. Here is the basic traffic we allow:

- Inbound :
  - SSH (restricted to IP address/subnet if possible)
  - PING (echo-request/reply, basic troubleshooting)
- Outbound:
  - DNS towards your DNS server

- NTP towards a ntp server
- SYSLOG towards your syslog server
- SMTP towards your email gateway (smart host)
- HTTP towards your preferred Debian mirror
- HTTP towards the security.debian.org mirrors

How to deal with multiple update servers

The fully qualified domain name for the Debian security update repository is "*security.debian.org*". Of course, many servers are available to provide load-balancing and redundancy. So every time you connect to "security.debian.org", you're possibly connecting to a different server on a different IP address. This causes a problem for our firewall rules because we want to restrict our outbound HTTP connections to specific IP addresses. This leaves us with two possible solutions: a lazy one, and a complete one.

The lazy one is quite simple: we shortcut the resolving process by adding this line in our /etc/hosts file:

```
194.109.137.218 security.debian.org     klecker.debian.org
```

This way, security.debian.org will always resolve to 194.109.137.218 (klecker.debian.org), and thus we only need one line in our firewall rules for this HTTP connection. There is a possibility for problems if "klecker" goes down for an extended period of time, because you will be without updates for your server(s), unless you change the update server manually when the problem arises. Although I haven't seen that yet, we should probably be more proactive and go for solution #2:

The complete solution is to put all the Debian security updates

servers in our firewall rules, so we have redundancy in case of problems with one of the server. Here's how you can get a list of the update servers:

```
alex@client:~$ dig security.debian.org

; <<>> DiG 9.3.4 <<>> security.debian.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24809
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;security.debian.org.           IN      A

;; ANSWER SECTION:
security.debian.org.    164     IN      A       212.211.132.32
security.debian.org.    164     IN      A       212.211.132.250
security.debian.org.    164     IN      A       128.31.0.36

;; AUTHORITY SECTION:
debian.org.             3464    IN      NS      klecker.debian.org.
debian.org.             3464    IN      NS      raff.debian.org.
debian.org.             3464    IN      NS      rietz.debian.org.

;; ADDITIONAL SECTION:
raff.debian.org.        3504    IN      A       192.25.206.59
rietz.debian.org.       3504    IN      A       140.211.166.43
klecker.debian.org.     3504    IN      A       194.109.137.218

;; Query time: 91 msec
;; SERVER: 192.168.2.66#53(192.168.2.66)
;; WHEN: Tue Oct  2 09:50:31 2007
;; MSG SIZE  rcvd: 194
```

With this list in hand, you need to add a line for each IP in our firewall rules: this is what we will do soon.

Creating the firewall configuration file

Let's create the firewall script: */etc/init.d/firewall* and configure it to start and stop automatically:

```
serveur:~# touch /etc/init.d/firewall
serveur:~# chown root:root /etc/init.d/firewall
serveur:~# chmod 755 /etc/init.d/firewall
serveur:~# update-rc.d firewall start 41 S . stop 89 0 6 .
 Adding system startup for /etc/init.d/firewall ...
   /etc/rc0.d/K89firewall -> ../init.d/firewall
   /etc/rc6.d/K89firewall -> ../init.d/firewall
   /etc/rcS.d/S41firewall -> ../init.d/firewall
serveur:~#
```

Edit the file and paste the following script into it. You need to change the variables of the IP Addresses section with the IPs of the servers in your network. Some rules may be of no use to you. For instance, if you don't have a Syslog server, you should comment out that rule in the “outbound” section. If your have one or two NTP servers, you should specify their IP addresses in the NTP rules instead of opening port 123 outbound to everything. I recommend that you read the “INBOUND” and “OUTBOUND” sections to familiarize yourself with the format of Netfilter rules.

```
#!/bin/sh
#------------------------------------------------------------------------
# /etc/init.d/firewall
#
```

```
# IPTables (netfilter) firewall manager script
#
# Server : serveur
#
# History of modifications
# When          Who                     What
# ----          ---                     ----------
# 2007-05-14    Harden Debian 4.0       Original version
#
#--------------------------------------------------------------------------



#--------------------------------------------------------------------------
# Global variables
#

IPTABLES='/sbin/iptables'       # Full path to “iptables” binary
MODPROBE='/sbin/modprobe'       # Full path to “modprobe” binary
DEPMOD='/sbin/depmod'           # Full path to “depmod” binary

FLAGS='URG,ACK,PSH,RST,SYN,FIN' # All flags but ECN
LOG_LEVEL="debug"

#---------------------------------------------------------------
# IP Addresses
#
SRV_LOG="192.168.2.2"    # syslog server
SRV_NTP="192.168.2.2"    # ntp (time) server
SRV_SMTP="192.168.2.30"   # smtp (mail gateway)
SRV_DNS="192.168.100.2"    # dns server

ADMIN_RANGE="192.168.42.0/24"  # Only this subnet will be allowed to SSH in

SRV_DEBIAN_MIRROR="206.167.141.10"          # gulus.usherbrooke.ca
```

```
      SRV_DEBIAN_SECURITY_1="212.211.132.32"   # villa.debian.org
      SRV_DEBIAN_SECURITY_2="212.211.132.250"Creating the firewall configuration
file   # lobos.debian.org
      SRV_DEBIAN_SECURITY_3="128.31.0.36 "   # steffani.debian.org


      #-------------------------------------------------------------------------
      # Function: Usage
      #    Shows a reminder
      #-------------------------------------------------------------------------


      Usage() {
          echo "Usage: $0 start|stop|restart"
          exit 1
      }


      #-------------------------------------------------------------------------
      # Function: StartFirewall
      #    Loads the rules in memory
      #-------------------------------------------------------------------------


      StartFirewall() {


      #-------------------------------------------------------------------------
      # Loading of kernel modules for filtering (some modules work better if
loaded first)
      #
      $DEPMOD -a


      $MODPROBE ip_tables
      $MODPROBE ip_conntrack
      $MODPROBE iptable_filter
      $MODPROBE ipt_LOG
      $MODPROBE ipt_limit
      $MODPROBE ipt_state
      $MODPROBE ip_conntrack_ftp
```

```
#------------------------------------------------------------------------
# Empty the “filter” table
#
$IPTABLES -t filter -F
$IPTABLES -t filter -X

#------------------------------------------------------------------------
# Default policy for all tables : drop everything
#
$IPTABLES -t filter -P INPUT DROP
$IPTABLES -t filter -P OUTPUT DROP
$IPTABLES -t filter -P FORWARD DROP

#------------------------------------------------------------------------
# Log entries definitions
#
# Every log “line” will be prefixed with "[FW:" (for firewall), to
# make log filtering easier down the road.

# Log DROPs
$IPTABLES -N LOG_DROP
$IPTABLES -A LOG_DROP -j LOG --log-prefix '[FW:DROP] ' --log-level
$LOG_LEVEL
$IPTABLES -A LOG_DROP -j DROP

# Log ACCEPTs
$IPTABLES -N LOG_ACCEPT
$IPTABLES -A LOG_ACCEPT -j LOG --log-prefix '[FW:ACCEPT] ' --log-level
$LOG_LEVEL
$IPTABLES -A LOG_ACCEPT -j ACCEPT

# Log REJECTs
$IPTABLES -N LOG_REJECT
$IPTABLES -A LOG_REJECT -j LOG --log-prefix '[FW:REJECT] ' --log-level
```

```
$LOG_LEVEL
      $IPTABLES -A LOG_REJECT -j REJECT


      # Drop weird packets


      # A packet can't have SYN+ACK and also be new! (state NEW)
      $IPTABLES -A INPUT -p tcp --tcp-flags SYN,ACK SYN,ACK -m state --state NEW -
j LOG_REJECT


      # No legal packet can have all flags on or off : doesn't make sense
      $IPTABLES -A INPUT -p tcp --tcp-flags ALL NONE -j LOG_DROP
      $IPTABLES -A INPUT -p tcp --tcp-flags ALL ALL  -j LOG_DROP


      #-------------------------------------------------------------
      # Loopback interface (lo : 127.0.0.1) must be open to itself


      $IPTABLES -A INPUT -i lo -j ACCEPT
      $IPTABLES -A OUTPUT -o lo -j ACCEPT


      # Anti-spoofing : traffic from 127.0.0.0/8 must originate from the loopback
interface
      $IPTABLES -A INPUT -s 127.0.0.0/255.0.0.0 -i ! lo -j LOG_DROP


      #-------------------------------------------------------------
      # Logging of start and end of connections (but not the “middle” packets)
      $IPTABLES -t filter -A OUTPUT -p tcp --tcp-flags $FLAGS SYN,ACK -m state --
state ESTABLISHED,RELATED -j LOG_ACCEPT
      $IPTABLES -t filter -A OUTPUT -p tcp --tcp-flags $FLAGS FIN,ACK -m state --
state ESTABLISHED,RELATED -j LOG_ACCEPT
      $IPTABLES -t filter -A OUTPUT -p tcp --tcp-flags $FLAGS RST,ACK -m state --
state ESTABLISHED,RELATED -j LOG_ACCEPT


      $IPTABLES -t filter -A INPUT  -p tcp --tcp-flags $FLAGS SYN,ACK -m state --
state ESTABLISHED,RELATED -j LOG_ACCEPT
      $IPTABLES -t filter -A INPUT  -p tcp --tcp-flags $FLAGS FIN,ACK -m state --
state ESTABLISHED,RELATED -j LOG_ACCEPT
      $IPTABLES -t filter -A INPUT  -p tcp --tcp-flags $FLAGS RST,ACK -m state --
```

```
state ESTABLISHED,RELATED -j LOG_ACCEPT


      # We accept without logging the packets in the “middle” of the connections
      $IPTABLES -t filter -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
      $IPTABLES -t filter -A INPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT


      #------------------------------------------------------------------------
      # INBOUND traffic (INPUT table)
      # Traffic addressed explicitly for this server (ie : not forwarded traffic,
      # if the server is used as router/firewall).


      # SSH
      $IPTABLES -t filter -A INPUT -p tcp --dport 22 -s $ADMIN_RANGE --tcp-flags
$FLAGS SYN -m state --state NEW -j LOG_ACCEPT


      # PING
      $IPTABLES -t filter -A INPUT -p icmp --icmp-type echo-request -j LOG_ACCEPT




      #------------------------------------------------------------------------
      # OUTBOUND traffic (OUTPUT table)
      # Traffic that this server sends (not forwarded traffic)


      # SMTP : Outgoing emails
      $IPTABLES -t filter -A OUTPUT -p tcp --dport 25  -d $SRV_SMTP --tcp-flags
$FLAGS SYN -m state --state NEW -j LOG_ACCEPT


      # DNS : Name resolution
      $IPTABLES -t filter -A OUTPUT -p udp --dport 53  -d $SRV_DNS -j LOG_ACCEPT
      $IPTABLES -t filter -A OUTPUT -p tcp --dport 53  -d $SRV_DNS --tcp-flags
$FLAGS SYN -m state --state NEW -j LOG_ACCEPT


      # HTTP : Debian mirror for software installation
      $IPTABLES -t filter -A OUTPUT -p tcp --dport 80  -d $SRV_DEBIAN_MIRROR --
tcp-flags $FLAGS SYN -m state --state NEW -j LOG_ACCEPT
```

```
      # HTTP : Debian security updates
      $IPTABLES -t filter -A OUTPUT -p tcp --dport 80  -d $SRV_DEBIAN_SECURITY_1 --tcp-flags $FLAGS SYN -m state --state NEW -j LOG_ACCEPT
      $IPTABLES -t filter -A OUTPUT -p tcp --dport 80  -d $SRV_DEBIAN_SECURITY_2 --tcp-flags $FLAGS SYN -m state --state NEW -j LOG_ACCEPT
      $IPTABLES -t filter -A OUTPUT -p tcp --dport 80  -d $SRV_DEBIAN_SECURITY_3 --tcp-flags $FLAGS SYN -m state --state NEW -j LOG_ACCEPT

      # SYSLOG : Centralized logging (disable if you don't have a syslog server)
      $IPTABLES -t filter -A OUTPUT -p udp --dport 514 -d $SRV_LOG -j ACCEPT

      # NTP : Time synchronization to a particular server
      # $IPTABLES -t filter -A OUTPUT -p udp --dport 123 -d $SRV_NTP -j LOG_ACCEPT
      # OR
      # Time synchronization to any NTP server on the network
      $IPTABLES -t filter -A OUTPUT -p udp --dport 123 -j LOG_ACCEPT

      # PING : Ultra basic troubleshooting
      $IPTABLES -t filter -A OUTPUT -p icmp -j ACCEPT



      #-----------------------------------------------------------------------
      # Log all packets before they are dropped
      # (default policy)

      $IPTABLES -t filter -A INPUT  -j LOG_DROP
      $IPTABLES -t filter -A OUTPUT -j LOG_DROP
      $IPTABLES -t filter -A FORWARD -j LOG_DROP
      }

      #-----------------------------------------------------------------------
      # Function: StopFirewall
      #    Stop the firewall and ACCEPT ALL TRAFFIC
      #-----------------------------------------------------------------------
```

```
StopFirewall() {

#--------------------------------------------------------
# Empty all filter tables

$IPTABLES -t filter -F
$IPTABLES -t filter -X

#--------------------------------------------------------
# Default policy : Accept everything
#

$IPTABLES -P INPUT ACCEPT
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
}

#---------------------------------------------------------------------------
# Function: RestartFirewall
#    Empty and reload firewall rules
#---------------------------------------------------------------------------

RestartFirewall() {

#--------------------------------------------------------
# Empty all filter tables
#

$IPTABLES -t filter -F
$IPTABLES -t filter -X

StartFirewall
}
```

```
#-----------------------------------------------------------------------
# Main program [ main() ]
#    Check first argument and launch appropriate function
#-----------------------------------------------------------------------

case "$1" in
        'start')
                echo -n "Loading firewall rules..."
                StartFirewall
                echo "OK"
                ;;
        'stop')
                echo -n "Removing firewall rules..."
                StopFirewall
                echo "OK"
                ;;
        'restart')
                echo -n "Removing and reloading firewall rules..."
                RestartFirewall
                echo "OK"
                ;;
        *)
                Usage
                ;;
esac

exit 0
```

Start the firewall. You might be disconnected while doing this, but you should be able to reconnect back.

```
serveur:~# /etc/init.d/firewall start
Loading firewall rules...OK
serveur:~#
```

18 Configuring the logging system

We've replaced the “sysklogd+klogd” logging combo with “syslog-ng”. This will enable us to do log filtering based on strings. The configuration file, while really longer than that of “Classic Syslog”, is actually readable by a human being, and really flexible. That configuration file is *“/etc/syslog-ng/syslog-ng.conf”.*

Redirect firewall logs to dedicated file

Since the Netfilter firewall is part of the kernel (either compiled-in or as a module), all the logs it generates (DROPs, ACCEPTS, FORWARDs, etc...) are from the “kernel” facility (in Syslog parlance, a facility is a source or origin of a message). The firewall will generate a *lot* of messages, and thus makes it hard to find “real” kernel messages when they are all saved to the “kern.log” file. Since we've already configured our logging rules to prefix all messages with “[FW:” (aren't we clever!), we only need to do some basic string matching to find them, and redirect them appropriately.

Add this to the “destinations” section:

```
# Firewall logs : specify a dedicated file for those
destination df_firewall { file("/var/log/firewall.log"); };
```

Add these filters to the “filters” sections:

```
filter f_only_debug { level(debug); };
filter f_firewall { match("\\[FW:"); };
filter f_not_firewall { not match("\\[FW:"); };
```

Modify these “log” commands so that we don't pollute those files with firewall logs:

```
# *.*;auth,authpriv.none          -/var/log/syslog
log {
        source(s_all);
        filter(f_syslog);
        filter(f_not_firewall);
        destination(df_syslog);
};

# kern.*                          -/var/log/kern.log
log {
        source(s_all);
        filter(f_kern);
        filter(f_not_firewall);
        destination(df_kern);
};Redirect firewall logs to dedicated file

# *.=debug;\
#         auth,authpriv.none;\
#         news.none;mail.none     -/var/log/debug
log {
        source(s_all);
        filter(f_debug);
        filter(f_not_firewall);
        destination(df_debug);
};
```

Add this “log” command at the end of the file:

```
# firewall                          -/var/log/firewall.log
log {
        source(s_all);
        filter(f_kern);
        filter(f_only_debug);
        filter(f_firewall);
        destination(df_firewall);
```

```
};
```

Logging to a remote Syslog server

If you have a working Syslog server (I'll call it “loghost”), here's how send a copy of every message from this server to your loghost. If you don't have/want one, then go ahead and skip this section.

Add this to the “destinations”:

```
# Loghost server : centralized logging
destination ds_loghost { udp("192.168.2.2" port(514)); };
```

Add this at the end of the file:

```
# *.*                           @loghost
log {
        source(s_all);
        destination(ds_loghost);
};
```

Reloading the configuration

```
serveur:~# /etc/init.d/syslog-ng restart
```

Rotating log files

Log files can grow up quite big if left unattended for a while. Rotation is the act of renaming an active log file, compressing it and creating a new one at regular intervals. Automatic weekly rotation of log files with 4 weeks of archive is the default on a Debian system. We only need to add our log file

(/var/log/firewall.log) to the configuration so it gets rotated at the same time.

Create */etc/logrotate.d/firewall* and add this to it:

```
serveur:~# vi /etc/logrotate.d/firewall

/var/log/firewall.log {
   rotate 4
   weekly
   missingok
   notifempty
   compress
   postrotate
      /etc/init.d/syslog-ng reload >/dev/null
   endscript
}
```

Let's force a rotation cycle and check everything went well:

```
serveur:~# cd /var/log
serveur:/var/log# ls -l firewall*
-rw-r----- 1 root adm  174 2007-05-15 09:56 firewall.log
serveur:/var/log# logrotate -f /etc/logrotate.conf
serveur:/var/log# ls -l firewall*
-rw-r----- 1 root adm  174 2007-05-15 09:56 firewall.log
-rw-r----- 1 root adm 1042 2007-05-15 09:55 firewall.log.1.gz
```

19 Configuring semi-automatic updates

To ease the process of updating your server(s), we'll automate part of the work. I do not recommend full automation (update + upgrade) because some updates require human input, and working around that is icky, so let's automate the boring stuff, and do the thinking

ourselves (that is what we are paid for, right?).

The automated part: every morning at 5:30AM, the server(s) will fetch the list of updated packages from Debian (apt-get update). Afterwards, a script will login to the server(s), verify what updates are needed (apt-show-versions -u) and mail a report to you.

The manual part: each morning, you will read your emails, and see what servers need updates. Now you have to think carefully about the impact of these updates: Can you try them on a test server? Do you have to reboot (kernel update)? Have you had your first caffeinated beverage yet? Once you've answered all these, you can go ahead and install the updates manually.

Automating the update

Add this to root's crontab:

```
serveur:~# crontab -e

#### Update the APT database every morning (apt-get update) ####
30 5 * * * apt-get update > /dev/null 2>&1
```

Automatic checking for available updates

Put this script on a server that can SSH (with a key) into all your servers:

```
#!/bin/bash
#
# update_check.sh
#
# Look for servers needing updates. We trust that apt-get update has already been done.
```

```
#
# When                  Who                          What
# 2007-02-12           Alex                         Original version
#

SERVEURS="serveur server-1 server-2 server-3"
for SERVEUR in ${SERVEURS}
  do echo ===Available updates for ${SERVEUR}===
     ssh ${SERVEUR} apt-show-versions -u 2> /dev/null
  done
```

Here's a sample crontab entry to run it and mail the report:

```
#### Checking for available updates ####
0 7 * * * /bin/bash /home/sysop/update_check.sh | /usr/bin/mail -s "Debian Updates Available (`/bin/date -R`)" your.name@domain.example
```

20 The end

Congratulations! You've reached the end! Here are some pointers about what to do next:

- Install any remaining stuff;
- DOCUMENT. YOUR. SERVER. ***IT'S IMPORTANT!***
- Store the passwords (root, sysop, etc...) at your designated place (if you have nothing, a PGP/GPG encrypted file is a good start);
- Add the server to your backup routine;
- Notify users of the changes;
- 0x3a28213a [3].

21 References

[1] Free Standards Group, (2004, January 29th). Filesystem Hierarchy Standard. Retrieved November 19, 2007, from Free Standards Group Web site: http://www.pathname.com/fhs/

[2] Krafft, Martin F. (2005). The Debian System: Concepts and Techniques. San Francisco, CA: No Starch Press.

[3] Munroe, Randall (2006, 08, 07). Pointers. XKCD, Retrieved November 19, 2007, from XKCD web site: http://xkcd.com/138/

[4] Fernández-Sanguino Peña, Javier (2007). Securing Debian Manual. Retrieved November 19, 2007, from Securing Debian Manual Web site: http://www.us.debian.org/doc/manuals/securing-debian-howto/

[5] Timme, Falko (2007, April 9th). The Perfect Setup - Debian Etch (Debian 4.0). Retrieved November 19, 2007, from HowtoForge Web site: http://www.howtoforge.com/perfect_setup_debian_etch