



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

# **Staged and tested by Open Source**

**By Guillaume Tamboise  
for SANS GCUX Certification  
Version 1.9, Option 1  
Original submission  
June 4, 2003**

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Conventions</b>	<b>1</b>
<b>3</b>	<b>Description</b>	<b>2</b>
3.1	Choice of Open Source	3
<b>4</b>	<b>Risk analysis</b>	<b>4</b>
4.1	Assets	4
4.1.1	Keyring	4
4.1.2	Distributed software	5
4.2	Access	5
4.3	Risks	5
4.4	Services	6
<b>5</b>	<b>Step by Step Guide</b>	<b>6</b>
5.1	Assumptions	6
5.2	Web of Trust	7
5.3	Physical Installation	8
5.4	CD Set	8
5.5	First Boot	10
5.6	Partitioning	11
5.7	Formating	13
5.8	Initial Kernel	15
5.9	Network Configuration	15
5.10	Base System	16
5.11	Dselect	17
5.12	Configuration	18
5.13	Recompiling the Kernel	18
5.14	Locking Down	20
5.14.1	Denial of Service	20
5.14.2	Shutdown	21
5.14.3	IP Stack	21
5.14.4	Firewall Rules	22
5.15	TCPWrapper	24
5.16	Inet	24
5.17	Apt	25
5.18	PGP key	26
5.19	Upgrade	30
5.20	Debian Master Keys	31

---

5.21 Release File . . . . .	32
5.22 Software Updates . . . . .	36
5.23 Additional Software . . . . .	37
5.23.1 Sudo . . . . .	38
5.23.2 Ippl . . . . .	38
5.23.3 Network Time Protocol . . . . .	39
5.23.4 Ssh . . . . .	40
5.23.5 Console Access . . . . .	41
5.23.6 Cracklib . . . . .	41
5.23.7 Apache . . . . .	42
5.23.8 Nessus . . . . .	44
5.23.9 Getting Newer Versions . . . . .	46
<b>6 Ongoing Maintenance . . . . .</b>	<b>51</b>
6.1 Backups . . . . .	51
6.2 Updates . . . . .	52
6.2.1 Key Updates . . . . .	52
6.2.2 Software Updates . . . . .	52
6.2.3 Kernel Updates . . . . .	55
6.3 Checking Logs with <i>Logcheck</i> . . . . .	55
6.4 Security Audits with <i>Tiger</i> . . . . .	56
<b>7 Check your Configuration . . . . .</b>	<b>56</b>
7.1 Remaining Services . . . . .	56
7.2 Firewall and TCPWrapper Configuration . . . . .	57
7.3 Verify Network Time Protocol (NTP) functionality . . . . .	59
7.4 Verify Software Download functionality . . . . .	60
7.5 Verify Software Distribution functionality . . . . .	62
<b>A Kernel Configuration . . . . .</b>	<b>64</b>
<b>B Firewall Configuration . . . . .</b>	<b>82</b>
<b>C Linux Kernel . . . . .</b>	<b>87</b>
<b>D Apt-release-check . . . . .</b>	<b>90</b>

---

## List of Figures

1	Network Topology	3
2	Initial web of trust	27
3	Web Browser	62

## List of Tables

1	Hardware and Software	2
2	Partitioning Schema	13
3	<i>dselect</i> main keys	17
4	Initial Configuration	18
5	<i>apt-get</i>	25

---

# 1 Introduction

Downloading software from the Internet brings new functionalities as much as security concerns . When updating software, at what point do we stop improving the service delivered by a server, and start installing poorly tested software that should never go to a production system? At what point do we stop patching a software for a high risk vulnerability and start adding a Trojan horse?

The GNU/Linux platform addresses this concern with a target, the Strong Distribution model [3]. This model uses cryptography to ensure the authenticity of the distributed software throughout its development cycle - from the upstream developer to the end user, going through quality assurance groups.

We will install Debian GNU/Linux, a pure Open Source Linux distribution and make it a software distribution and remote vulnerability audit server. The purpose of the server will be to provide a lab environment with a trusted source of software, along with tools to perform network security audits. This server will be able to download software that will be authenticated using PGP. It will make it available for staging by lab systems. The server will also be able to perform testing on those systems to ensure that the software (or software patch) has been properly installed. To achieve this, the server users will have acquired some knowledge of various webs of trust and will have initiated a web of trust within the company.

This approach privileges some initial investment. Not much hardware or software investment, considering the fact that we will run a free Operating System server on an outdated Sun Ultra 10. The investment is about the effort we are going to make to get, trust and preserve the PGP keys involved in software distribution. Instead of having to take the risk of trusting a checksum each time we download a software, we invest once and trust the key that will sign series of checksums.

## 2 Conventions

Text in `typewriter` characters is printed on a screen.

Text to be typed is in **bold characters**.

Characters in ***bold italic*** are interesting areas of a screenshot.

Backslash (\) is the Unix Shell character used to get a shell command spanned on multiple lines. When performing copy-paste from the PDF document, make sure that the \ is followed directly by a carriage return and not a space.

☞ means that the line of text is from a screenshot and has been artificially split to a new line to fit in this document.

We are using the standard Debian bash prompt in most of the document. The user ID typing the command is either *user* or *root*. In the former case, the user prompt finishes with a dollar sign (\$), in the latter it finishes with a number sign (#). The path where the commands are launched is also indicated in the prompt.

---

In the rest of this document, the following definitions will apply:

- *Open Source* follows the definition of the Open Source Initiative [10]. It happens to have a common ground with Debian's *Free Software Guidelines* [6]. It will be taken as a synonym of Free, as opposed to non-Free in Debian's distribution model.
- *Freeware* is software from the public domain.
- *Strong Distribution* model [3]:

A strong distribution model is a model of software distribution which is cryptographically strong. In such a distribution model software archives and source code are protected against alteration, damage and replacement through the science of cryptography. Specifically, a strong distribution model is a distribution model which makes use of public key cryptographic technology to make attack, fraudulent presentation, compromise and alteration theoretically hard problems<sup>1</sup>.

- *PGP* and *GPG* will both refer to *GnuPG*, the Open Source counterpart of the commercial software *Pretty Good Privacy*.

### 3 Description

Component	Details
Platform	Sun Sparc Ultra10
CPU	TI UltraSparc Ili
Memory	256 MB
Network Card	Internal and Quad FastEthernet card <sup>2</sup>
Hard Drive	IDE 36 GB
Tape drive	SCSI
SCSI Controller	LSI Logic / Symbios Logic 53c875 (rev 14)
Debian	3.0 r0

Table 1: Hardware and Software

Table 1 gives a quick overview of the software and hardware involved. The version of Debian implies the versions for all software distributed with it (including Apache).

The networking environment is described in figure 1. The Intranet is not a trusted environment. The lab network, however, is fully trusted. Please note that the cloud

---

<sup>1</sup>To keep it simple, a hard problem in cryptography is a problem for which there exists no systematic and efficient way to find a solution. See [9] for some more details.

<sup>2</sup>Both cards are recognized as Happy Meal (rev 01) by Linux kernel 2.4.x

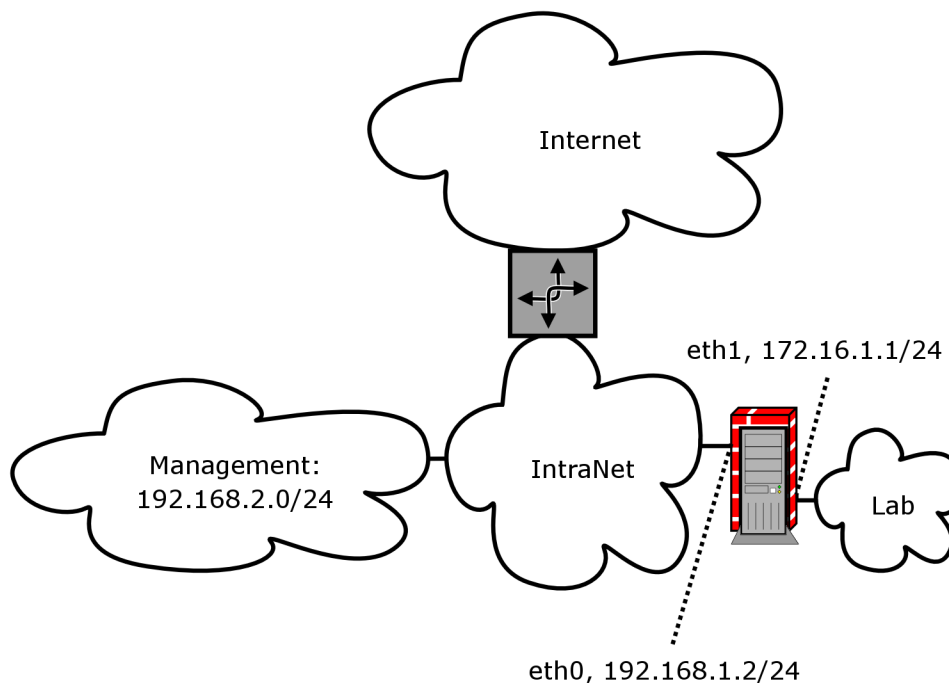


Figure 1: Network Topology

representing the lab network can contain a single switch, or even cross-over cables leading to the Quad FastEthernet Network Interface Card on the Sun server.

### 3.1 Choice of Open Source

John Viega and Gary McGraw wrote a highly instructive chapter on some of the security aspects of Open Source and Closed Source Software [11]. The last sentence of the chapter gives the general message:

Don't use the source availability, or lack thereof, as a crutch to convince yourself that you've been duly diligent when it comes to security.

As stated in its copyright, Debian GNU/Linux comes with absolutely no (legal) warranty. There is no commercial company to sue in case of catastrophic failure. It is definitely one of the major changes compared to any closed source (e.g. from Sun, Microsoft) or Open Source commercial vendor (e.g. from RedHat, Suse).

Lack of legal warranty does not mean anarchy. Debian as a project is built around a set of documents [6] worthwhile reading. Processes are as open as the distribution. Some examples:

- The *Debian constitution* gives the organizational structure for formal decision-making in the Project



- 
- The *Debian policy* describes the requirements of the distribution.
  - The *Debian Developer's Reference* provides an overview of the recommended procedures and the available resources for Debian maintainers.

Debian maintainers organize Open Source software in a complete Operating System formed of packages. Packages can contain software, documentation or even the Linux kernel. There are many benefits to this packaging process (common coherent ground, yet another review of the software as part of the “Many-Eyeballs Phenomenon” [11], testing performed by an entity different than the upstream developer, more secure default configuration...). However, it does not mean that the choice of the Debian maintainer will best fit the system administrator's needs. Or even that the maintainer has not overlooked some aspects of the configuration. This process can induce security issues instead of reducing them, as it has been reportedly the case with Apache in the past [12]. Trusting Debian means trusting this community of developers (or maintainers) to have the expertise to add value to the Open Source software, and to have the correct processes in place to enforce their policies.

Simultaneously, Debian developers themselves are aware that they still have a long way to go before achieving a Strong Distribution [3]. So far, a good deal of the required software tools have already been developed. Some existing processes already use the benefits of digital signature. Are included the very process that gets a developer to enter the Debian project itself<sup>3</sup>, security advisories and distribution releases. However, some processes were still missing at the time of the release, which excluded the corresponding tools from the Debian 3.0 release (e.g. individual digital signature of the packages).

## 4 Risk analysis

### 4.1 Assets

The system has two main valuable assets:

#### 4.1.1 Keyring

It takes some considerable time and effort to enter (be it in a unidirectional way) in the webs of trusts the various vendors belong to. The most valuable asset of this server is definitely its trusted keyrings..

---

<sup>3</sup>The members of the Debian project are defined by their belonging to a PGP web of trust that will be explained in 5.2. The process to enter the web of trust is defined in the Debian Developer's Reference [6].

---

### 4.1.2 Distributed software

The distribution server is the main way for the lab systems to be staged with trusted software, before they are sent to the production network. This staging phase impacts potentially all the devices installed on the Company's network: workstations, servers, routers, switches...

## 4.2 Access

This server needs to be accessed by its system administrator and by a number of users. Users need to have the permissions to make available the software that they have authenticated. They also need to be able to perform network tests on lab devices to check the result of their staging work.

## 4.3 Risks

The main risk associated with the assets listed in section 4.1 boils down to the company devices being compromised at staging time.

- **Physical Access:** As for any sensitive server, physical access should be protected. Tamper proof hardware is far from being widely spread, and servers certainly do not fall in this category. An attacker with physical access can also easily launch a Denial of Service attack. Using physical access to compromise the server<sup>4</sup> would cause a major hit on our assets.
- **Network Access:** By its very nature, this system needs access to both an untrusted network (the company's Intranet) and a trusted network (the lab). There are two security issues induced by the connectivity:
  - The Intranet can compromise this server, its keyring or its distributed software using its network connectivity.
  - The lab devices are in a test environment and are expected to behave as in a test environment. All these exotic routing protocols, conflicting IP addresses or other network-related anomalies that may be tested should stay within the lab environment and not affect the production network.
- **Operating System:** One of the purpose of this system is to make sure that the software it distributes has not been altered from the moment it left its original location. The Operating System itself might get compromised by a Trojan or other

---

<sup>4</sup>*Start of down time* - shutting the server down, taking the hard drive out, remounting it on another server, writing down the encrypted version of the *root* password, wiping it out, remounting the hard drive back to the original server, rebooting - *End of down time*, installing whatever rootkit, clean logs, put the encrypted version of the password back in place.

---

form of undesirable (untested or unmaintained) software. Matt Power [13] gives a very interesting threat model regarding the patching process in particular.

- Administration: Each Debian package is supported by a package maintainer, and in case of security risks the patch process can be accelerated by the Debian security team. This process is regulated by the social contract [6]. Debian has proved very reactive when vulnerabilities are discovered. The security limiting factor will remain with the system administrator. The administrator has tools (*apt-get*, web pages with recent security alerts [7], the debian-security-announce mailing list<sup>5</sup>). Not using them to keep the system up to date makes the system vulnerable.

## 4.4 Services

This server needs to run two services on a permanent basis:

- An Ssh server for its remote administration and to trigger the software upload from the Internet;
- An Apache server per its role of distribution server.

On an as-needed basis, the *Nessus* daemon needs also to run.

## 5 Step by Step Guide

### 5.1 Assumptions

This paper assumes that anyone reading this paper has some knowledge of

- Console access;
- Packaged Operating Systems in general;
- Unix command line.

Moreover, in the rest of this paper the user must trust its administrator. Meaning trust *root* and all users with administrator privileges. *PGP* will not protect the user from a malicious system administrator, no matter what<sup>6</sup>.

---

<sup>5</sup>See <http://lists.debian.org/>.

<sup>6</sup>System administrators have the Unix permissions required to tamper with PGP's binaries. Or simpler, write a wrapper in Bourne shell.

---

## 5.2 Web of Trust

A Web of Trust is a network of PGP keys. A node of this network is a pair of PGP keys: A private key that the node is the only one to know, a public key that can be freely sent. Only the public key can decrypt what the private key encrypts (which provides digital signature), and only the private key can decrypt what the public key encrypts (which provides secrecy).

Two nodes are linked if they have digitally signed each other's public key. This link can be unidirectional if only one of the two nodes signs the other's node key. Signing a key means recognizing as true the association between the key and the owner of the key (whether a human being or an administrative role). The signature is then part of the key and is made available to the community. So is the revocation of the key. The owner itself is defined by a name and an email address<sup>7</sup>, which builds a unique identification.

A node's keyring represents the vision of the Web of Trust from the node's perspective. First, each key in the keyring is assigned a *trust*, in the sense "trust to correctly sign other keys". The level of trust can have these values:

- (n) no trusted
- (m) marginally trusted
- (f) fully trusted
- (u) ultimately trusted

On an individual basis, the software used by each node needs to decide the *validity* of a newly inserted key. It uses the existing keys in the keyring to calculate a level of trust [14][8] in the sense "trust to correctly make the link between the public key and the physical person (or administrative role)".

A Key  $K$  is considered valid if it meets two conditions:

1. it is signed by enough valid keys, meaning
  - we have signed it personally,
  - it has been signed by one fully trusted key, or
  - it has been signed by three marginally trusted keys; and
2. the path of signed keys leading from  $K$  back to our own key is five steps or shorter.

This default behavior can be tuned by `--completes-needed` (*number of completely trusted users to introduce a new key signer (defaults to 1)*), `--marginals-needed`

---

<sup>7</sup>In fact, could be many names and many email addresses

---

(*Number of marginally trusted users to introduce a new key signer (defaults to 3)*) and `--max-cert-depth` (*Maximum depth of a certification chain (default is 5)*).

It is definitely interesting that the default maximum depth of a certification chain is 5. Studies [4] have indeed considered another kind of web, where a link between the human beings  $X$  and  $Y$  exists if they know each other. Using this metric, two people in the world are distant of at most 6 hops. This rule would not apply only to human networks.

In this document, we will show the steps that the system administrator would perform to install the Operating System. On a multi-user system, it is definitely possible to imagine that each user owns a public/private key pair. Users would sign each other's key and trust each other to make the right key signing decisions. This way, the work load required to evaluate the level of trust the company can put in relevant keys can be distributed.

### 5.3 Physical Installation

If the lab is only used internally, we may well decide to leave the server running just like another lab device - racked nearby one of these very pricey routers. The lab security policy should take care of physical access. The connectivity of the lab has to be visually straightforward: One connection and only one is coming from the Intranet to the lab, and this connection is going to be patched later to the distribution server using a designated interface.

Visually, the server should be connected to two cables at this point:

1. One power cable;
2. One console cable going to a trusted host that has no network connectivity.

### 5.4 CD Set

By one way or another, we have to get an ultimate trust in the set of Debian CD.

There are various ways to get such a set of CD, like buying it or downloading the ISO image from the Internet. Buying it could be a way to offload the responsibility of carrying out the trust checks onto the reseller. We would be making a series of assumptions:

- The reseller does a good job in insuring the authenticity of the CD set;
- If we physically go and buy the CDs, it is indeed the reseller that we have in front of us;
- If we decide to get the CD set mailed to us, we trust the carrier not to alter the CDs and we trust the person that gives us the CDs to be the carrier.

---

Once we have a CD set, we download the checksum file from one of the mirrors<sup>8</sup> and check its signature. MD5 checksums are fine, but SHA1 tend to be more reliable because this algorithm is less sensitive to Birthday attacks [1].

We can check the MD5 checksum on an existing and trusted Unix/Linux system using *md5sum*:

```
# md5sum /dev/cdrom
0c42db56a8fe72d1e9f22f8c1cf74c62 /dev/cdrom
```

We can also use the *openssl* tool:

```
# openssl md5 /dev/cdrom
MD5(/dev/cdrom)= 0c42db56a8fe72d1e9f22f8c1cf74c62
```

Note that *openssl* can also provide us with the SHA1 checksum:

```
# openssl sha1 /dev/cdrom
SHA1(/dev/cdrom)= af3618f539f4f81060c65db3a753dde71b39232d
```

One step is still missing: Checking the validity of the MD5 or SHA1 checksum file that we have previously downloaded.

```
$ gpg --verify MD5SUMS
gpg: Signature made Sat Jan 11 14:18:19 2003 CST using DSA key ID DD9B9910
gpg: Good signature from "Philip Hands <phil@hands.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 8DF8 CE53 24F3 177C 2417 6C65 6203 8A4B DD9B 9910
```

Even if we have the key that signed the checksum, we still need to trust it... It is a good occasion to enter the web of trust! We can download Debian's keyring [6] (Philip Hands's key is in it), but we need to have a real life additional check to ensure the authenticity of the keyring. Checking someone's key can typically happen in a key signing party<sup>9</sup> [2].

To improve our trust in the CDs, we can also purchase them from multiple vendors in different countries<sup>10</sup>, and compare the checksums.

Anyway, at this point, we should have a full set of Debian CDs, or at least the very first one and the *non-US* one. And we should fully trust all of them to be the direct work of the Debian developers.

---

<sup>8</sup>Using <http://gd.tuwien.ac.at/opsys/linux/debian-cd/images/> for instance.

<sup>9</sup>*Linuxtag 2003* will host such a party, <http://www.palfrader.org/ksp-1t2k3.html>. See also *Debconf3*, for Debian Conference, on <http://www.debconf.org/debconf3/keysigning.php>

<sup>10</sup>See <http://www.debian.org/CD/vendors/> to have a list of Debian CD vendors.

---

## 5.5 First Boot

We need to go to the `ok` prompt to start the installation. To do this, we send a break to the console<sup>11</sup>. It is the best moment to make sure that we have an OpenBoot password:

```
ok setenv security-mode command
security-mode =      command
ok setenv security-password password
security-password =
```

This password makes the life of an attacker with physical access a little bit more difficult: She has to physically remove a drive to compromise the system.

In order to boot using the cdrom, we type `boot cdrom`:

```
Boot device: /pci@1f,0/pci@1,1/ide@d/cdrom@0,0:f  File and args:
SILO
```

```
Welcome to Debian GNU/Linux 3.0!
```

This is the Debian Install CD. Keep it once you have installed your system, as you can boot from it to repair the system on your hard disk if that ever becomes necessary.

WARNING: You should completely back up all of your hard disks before proceeding. The installation procedure can completely and irreversibly erase them! If you haven't made backups yet, remove the rescue CD from the drive and press L1-A to get back to the OpenBoot prompt.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
[ ENTER - Boot install ]  [ Type "rescue" - Boot into rescue mode ]
boot:
```

Our Ultra 10 has a 64 bit CPU. Because of a bug in the installation CD, it is necessary to specify both the Linux kernel to use and the location of the initial ramdisk:

```
boot: /boot/sparc64 
initrd=/dists/stable/main/disks-sparc/current/images-1.44/root.bin
```

On a 32 bit system, we would just have hit `<Enter>`.

The first screen gives a short description of Debian, we just hit `<Continue>`. Browsing through the options is performed using the `<Tab>`ulation key, we accept a choice by hitting the `<Enter>` key. Then comes a description of the partitioning process, we pick `<Next>`. `/dev/hda` is the IDE disk we are going to use for the installation<sup>12</sup>. SCSI systems have their disk drivers under `/dev/sd*`.

---

<sup>11</sup>There are various ways to achieve this, depending on the software used to get the console access.

<sup>12</sup>Beware on *sun4m* (Sparc 32 bits) systems with multiple disks: What Linux calls `/dev/hda` for first disk, `/dev/hdb` for second disk is not necessarily what OpenBoot calls first and second disk...

---

## 5.6 Partitioning

First thing first, let us create a Sun disklabel:

Command (m for help): **s**

Building a new sun disklabel. Changes will remain in memory only, until you decide to write them. After that, of course, the previous content won't be recoverable.

Drive type

? auto configure  
0 custom (with hardware detected defaults)  
a Quantum ProDrive 80S  
b Quantum ProDrive 105S  
c CDC Wren IV 94171-344  
d IBM DPES-31080  
e IBM DORS-32160  
f IBM DNES-318350  
g SEAGATE ST34371  
h SUN0104  
i SUN0207  
j SUN0327  
k SUN0340  
l SUN0424  
m SUN0535  
n SUN0669  
o SUN1.0G  
p SUN1.05  
q SUN1.3G  
r SUN2.1G  
s IOMEGA Jaz

Select type (? for auto, 0 for custom):

The safest choice seems to be 0. Default values come from an autodetect, they are usually good. We press <Enter> for each question. The partitioning scheme should now look like:

Command (m for help): **p**

Disk /dev/hda (Sun disk label): 64 heads, 32 sectors, 34730 cylinders  
Units = cylinders of 2048 \* 512 bytes

Device	Flag	Start	End	Blocks	Id	System
/dev/hda1		0	34680	35512320	83	Linux native
/dev/hda2	u	34680	34730	51200	82	Linux swap
/dev/hda3		0	34730	35563520	5	Whole disk

We remove the first two partitions but keep "whole disk":

Command (m for help): **d**



---

Partition number (1-8): **1**

Command (m for help): **d**

Partition number (1-8): **2**

Command (m for help): **p**

Disk /dev/hda (Sun disk label): 64 heads, 32 sectors, 34730 cylinders  
Units = cylinders of 2048 \* 512 bytes

Device	Flag	Start	End	Blocks	Id	System
/dev/hda3		0	34730	35563520	5	Whole disk

To add a Linux native partition (like a 1 GB *root* partition), we just type

Command (m for help): **n**

Partition number (1-8): **1**

First cylinder (0-24620): **0**

Last cylinder or +size or +sizeM or +sizeK (0-24620, default 24620): **+1024M**

To add a 512 MB swap (twice the size of the physical memory, even though only once the size of the physical memory might be enough: We do not want to be short in swap space),

Command (m for help): **n**

Partition number (1-8): **2**

First cylinder (725-24620): **725**

Last cylinder or +size or +sizeM or +sizeK (725-24620, default 24620): **+512M**

Command (m for help): **t**

Partition number (1-8): **2**

Hex code (type L to list codes): **L**

0	Empty	4	SunOS usr	7	SunOS var	83	Linux native
1	Boot	5	Whole disk	8	SunOS home	8e	Linux LVM
2	SunOS root	6	SunOS stand	82	Linux swap	fd	Linux raid auto
3	SunOS swap						

Hex code (type L to list codes): **82**

Changed system type of partition 2 to 82 (Linux swap)

After creating the partitions listed in table 2, the final partitioning scheme should look like:

Command (m for help): **p**

Disk /dev/hda (Sun disk label): 64 heads, 32 sectors, 34730 cylinders  
Units = cylinders of 2048 \* 512 bytes

Device	Flag	Start	End	Blocks	Id	System
--------	------	-------	-----	--------	----	--------

---

```

/dev/hda1          0      1024    1048576    83  Linux native
/dev/hda2  u      1024     1536     524288    82  Linux swap
/dev/hda3          0    34730   35563520     5  Whole disk
/dev/hda4        1536     5632    4194304    83  Linux native
/dev/hda5        5632    13824    8388608    83  Linux native
/dev/hda6       13824    15872    2097152    83  Linux native
/dev/hda7       15872    17920    2097152    83  Linux native
/dev/hda8       17920    34730   17213440    83  Linux native

```

Partition	Size	System	Mounting Point
/dev/hda1	1024 MB	Linux native	/
/dev/hda2	512 MB	Linux swap	swap
/dev/hda4	2048 MB	Linux native	/usr
/dev/hda5	8192 MB	Linux native	/var
/dev/hda6	1024 MB	Linux native	/opt
/dev/hda7	1024 MB	Linux native	/tmp
/dev/hda8	16 GB	Linux native	/home

Table 2: Partitioning Schema

Partitions 1 to 7 are just given more than what they really need. /home takes everything that remains, with the idea that distributed software will end up in this partition.

It is time to write the changes and go to the next step:

```

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

```

## 5.7 Formating

Current step is Next: Initialize and Activate a Swap Partition. Checking for bad blocks is always a good idea and does not take so long: We choose <Yes>, and <Yes>, we are sure.

Now, Next: Initialize a Linux Partition. As for the Filesystem Type, Ext3 is definitely the way to go. It makes it far less likely to get a corrupted file system.

If a file system gets corrupted (by a power cut, a faulty UPS, a hard reboot caused by a crash...), Debian automatically tries to repair it using *fsck* during the boot process. If the automatic way does not work, the console gives the choice between hitting <CTRL>+D for a normal startup and entering the *root* password for maintenance. Mounting a damaged file system is not advised. It is time to either find the one who knows the *root* password

---

or unseal the envelope containing it. The next step would then be to change the *root* password, and put in back in its safe place - this is just a short summary of what could be our *root* password handling policy. So much for not having a journalized filesystem.

For the *root* partition, we pick `/dev/hda1` (again, checking the bad blocks using this read-only test does not take so long).

We now need to repeat this step for all partitions:

Alternate : Initialize a Linux Partition,  
Ext3 Next Generation of Ext2, a journaling filesystem  
successively on `/dev/hda4`, `/dev/hda5`, `/dev/hda6`, `/dev/hda7` and `/dev/hda8`.

Two more points about the process of formatting. It should give us something similar to

```
Creating Ext3 filesystem (for 2.2 and newer kernels only)...
mke2fs 1.27 (8-Mar-2002)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
1048576 inodes, 2097152 blocks
104857 blocks (5.00%) reserved for the super user
First data block=0
64 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Checking for bad blocks (read-only test): done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

- 5 % of each partition is reserved by default to the *root* user. The idea is that even if a partition gets full for user *lambda*, the critical processes owned by *root* still have some room available and keep the server running. `tune2fs -m 0 /dev/hda8` would scale this number down to 0 % for the `/home` partition. Reserving 800 MB (5 % of 16 GB) of a partition where *root* is hardly supposed to write should sound odd. This tuning is usually performed on mounted filesystems without any issue.
- the filesystem is automatically checked every 36 mounts or 180 days. This setting dates back to the days of `ext2`, the non-journalized file system. If our server is reasonably stable, it should mean that all filesystems are checked at every reboot (we do not intend to have to reboot our system more often than once a quarter). On

---

our hardware configuration (36 GB hard drive), it does not take long, we will not bother changing the setting. It might be worth the trouble on larger or slower hard drives.

## 5.8 Initial Kernel

The step `Next: Install Kernel and Driver Modules` will be quick. We will not spend much time on adding fancy modules to the kernel, because the first thing we will do once the system is installed is to recompile a brand new one. `yes`, we want to use the CD for this installation, then `Next: Configure Device Driver Modules` and straight away `Exit Finished`. Return to previous menu.

## 5.9 Network Configuration

As for the network configuration, here are the questions we are facing:

- Name of the system: **gcux**
- Interface: **eth0**
- Automatic Network Configuration: **No**. DHCP is very handy in a mobile environment (typically users with laptops). However, it also makes it easier to impersonate a gateway or a DNS server. Once a cracker manages to make us use her as our gateway, she sees just all the traffic we send without having to bother with more sophisticated attacks like ARP spoofing.
- IP address: Whatever our network administrator configured. **192.168.1.2** in our case.
- Netmask: Whichever way our network administrator chose to subnet our network, **255.255.255.0**.
- Gateway: Whichever gateway our network administrator installed. Usually, the first IP address of the network (right after the network address). **192.168.1.1** in our case.
- Domain name: Whichever domain name our network administrator assigned us. **mydomain.com** in our case.
- DNS Server Addresses: Coming again from our network administrator. More than one DNS is generally a good idea, to use the automatic fail-over mechanisms built in the protocol.

Remember that at this point, the server is still not connected to the network.

---

## 5.10 Base System

The current step is Install the Base System. The medium we are using to install the system will be the CD-ROM drive. The Archive Path is the one proposed in `/instmnt`. We then Make System Bootable and Reboot the System.

At the time of the reboot, if you have multiple hard drives it is the good time to remember that the kernel is often in `1/vmlinuz` and that our root filesystem is `/dev/hda1` (see section 12).

If your screen looks like

```
Your imagename `` and arguments `` have either wrong syntax,
or describe a label which is not present in silo.conf
Type 'help' at the boot: prompt if you need it and then try again.
boot:
```

then

```
boot: 1/vmlinuz root=/dev/hda1
```

might well get you out of the woods.

Here comes the time of the Debian System Configuration:

- `<Yes>`, our hardware clock is set to GMT.
- `<Yes>`, we want to enable md5 passwords. This system will not interact much with other systems, there is no point to fall back to the weaker DES56 scheme. Do an `openssl speed` one day to have a clue about the additional complexity the attacker has to face.
- `<Yes>`, we want shadow passwords. There is no reason to give anyone access to the encrypted version of the password file.
- We pick a *root* password. There are tons of free software here and there that provide random passwords. The root password of this system is not supposed to be used except during the installation phase and in case of emergency. It does not have to be easy to remember. We should pick a password with at least 8 characters, with mixed case, digits and non alphanumeric characters.
- `<Yes>`, we want to create a normal user account now. Let us call it *user*, full name *Test User*. Each user with a UID greater or equal than 1000 should be a unique physical person<sup>13</sup>, whose full name is populated correctly in the `/etc/passwd` file.

---

<sup>13</sup>Unix users shared by multiple employees defy the purpose of accountability. This situation must be avoided.

- <No>, we do not want to use a PPP connection to install the system. In fact, we will never use a PPP connection.
- <Yes>, we want to scan another CD. We really want the `non-US` CD to be scanned because it is where we are going to find the `debian-keyring` package.
- <No>, we do not want to Add another `apt` source at this point. Most of the `apt` sources will come from the Internet, and we are not ready to connect to it.
- <No>, we do not want to Use security updates from `security.debian.org`, because we have no Internet connectivity at this point and no way to trust the content coming from *what we think is* this web site yet.
- <No>, we do not want to Run `tasksel`, because we want to keep a tight control on the list of packages we are going to install.
- <Yes>, we want to Run `dselect`. We read the next screen, and press the space bar.

## 5.11 Dselect

`apt` front-ends like *aptitude* have come along and made the long lived *dselect* kind of obsolete. However, and it is usually the approach preferred by Debian, *dselect* stays in the loop because it has proved robust and does the job. It is still the tool of choice for the initial packages installation of Debian 3.0.

Action	Key
Remove package	<code>-</code>
Install package	<code>+</code>
Search for package	<code>/</code>

Table 3: *dselect* main keys

The basic navigation is explained in table 3. We use these keystrokes to find and remove *nfs-common*, *pidentd*, *portmap* (*netbase* suggests *portmap*, but we press <ENTER> to confirm that we do not want it), *ssh*, *ipchains*, *ppp* (*dselect* proposes to remove *pppoeconf*, *pppconfig* and *pppoe*, we accept with <ENTER>), *dhcp-client*, *biff*, *vacation* (*bsdmainutils* suggests it but we can safely press <ENTER>), *lpr*, *ispell*, *iamerican*, *ibritishwenglish*. We can now use the same keystrokes to install *kernel-source-2.4.18* (along with all the packages that *dselect* suggests to install) and *debian-keyring*.

We have explicitly chosen not to install *Apache* or *ssh* because those packages have more recent version on `security.debian.org`.

We also have to be aware of the fact that we are including a development environment in the installation. The choice we are making is to compile whatever really needs compiling on the server itself, versus involving another server (that we would have to trust) to compile packages for us. Between making the life of a potential intruder easier because a compilation environment is ready for her and having to trust another server to compile for us, we choose the former evil.

The rest of the dialog looks like `<ENTER>`, `<SPACE>` followed by a proposition to add or remove packages, `<ENTER>`. And `Y`, we want to continue.

## 5.12 Configuration

Package	Desired Configuration
<i>Setserial</i>	AutoSave Once, <code>&lt;Ok&gt;</code>
<i>Binutils</i>	<code>&lt;Ok&gt;</code>
<i>Exim</i>	4 for all mails to stay local, user, <code>Y</code>
<i>less</i>	<code>&lt;No&gt;</code> mime handler for "application/*" <sup>14</sup>
<i>Locales</i>	en_US(?), C by default

Table 4: Initial Configuration

Table 4 gives the desired configuration for the (few) packages that are going to be installed. It is time to log in as *root*, and do ***ifdown eth0*** to stop the error messages about *eth0*. It is still not time to be connected to the network.

At this point, the number of packages installed should be more or less

```
gcux:~# dpkg --get-selections | wc -l
168
```

for a total size of

```
gcux:~# du -sh /
375M    /
```

## 5.13 Recompiling the Kernel

Compile work is usually performed under `/usr/local/src`. My problem with this location is that it is in the same partition as `/usr`. We definitely do not want `/usr` to become full, and we do know that compiling often takes a growing and not so predictable amount of space. Let us separate these two partitions, using the `/opt` that we created in section 5.6:

```
gcux:~# mv /usr/local/* /opt && rmdir /usr/local && ln -s /opt /usr/local
```

---

The only compile work that we will allow ourselves in `/usr` will be the kernel itself.

Recompiling the kernel is the best way to make it suite exactly both our hardware and functionality requirements. Assuming that we put the kernel configuration (provided in appendix A) in `/root/config-gcux`,

```
gcux:~# cd /usr/src
gcux:/usr/src# ls
kernel-source-2.4.18.tar.bz2
gcux:/usr/src# tar -xjf kernel-source-2.4.18.tar.bz2
gcux:/usr/src# ln -s kernel-source-2.4.18 linux
gcux:/usr/src# cd linux
gcux:/usr/src/linux# make mrproper
gcux:/usr/src/linux# cp /root/config-gcux ./config
gcux:/usr/src/linux# make-kpkg --revision gcux.1 kernel_image
```

The options chosen for this kernel do work with a Sun Ultra 10. I made little efforts to configure this kernel to be portable to other Sparc64 hardware. In particular,

**Lines 10-15** We want a kernel with modules enabled, even though modules might represent a vulnerability. Modules are pieces of code that can be inserted on demand in the kernel<sup>15</sup>. If an attacker manages to alter a module or insert a specially crafted module, she can virtually control the entire Operating System.

**Lines 23-24** This kernel is compiled for a 64 bit uniprocessor architecture. All software are not SMP safe, we avoid it because we do not need it.

**Lines 61-82** Printers can be used for example to print logs or public keys. We compile these options as module in case we need them in the future.

**Lines 139-200** We compile as modules most of the firewall abilities, even though this document will one use a tiny subset of them.

**Lines 310-344** We need SCSI for our backups on tape drives.

**Lines 369-445** We compile as modules most of the network interface card drivers. We do not want to have to recompile the kernel if we happen to change a NIC.

**Lines 511-559** We put the support for the `ext2` and `ext3` filesystems in the kernel, because the support for the root filesystem has to be built in the kernel (and because we might have to boot either in `ext2` or `ext3` in the future).

Some features built in the kernel could be safely removed, some could be added, but this configuration does the job it has been designed for.

---

<sup>15</sup>In our case, we use `kmod` that is configured in the `/etc/modutils` directory. Changes are committed using the command `update-modules`.



---

The configuration of a Sparc 32 kernel offers different and more limited options. It is beyond the scope of this document.

Compiling the source archive as *root* is not the cleanest choice ever. However, if we do not trust the archive of our own kernel, we have quite some other worries coming along.

We may now install the freshly compiled kernel using

```
gcux:/usr/src# dpkg -i kernel-image-2.4.20_gcux.1_sparc.deb
```

`/etc/silo.conf` should now have two kernels. We boot by default on the 2.4.18:

```
partition=1
root=/dev/hda1
timeout=100
default=linux
password=putyourpasswordhere

image=1/boot/linuz-2.2.20-sun4uvm
    label=oldlinux
    read-only

image=1/boot/vmlinuz-2.4.18
    label=linux
    read-only
```

We also added a boot password. It should be different to the OpenBoot password, but it can be more disseminated that the *root* password. There is no real need to launch **silo**<sup>16</sup>, but we prefer to do it just in case we made a syntax error in `silo.conf`.

## 5.14 Locking Down

The following subsections aim at locking down the server before being able to safely update it using the Internet.

### 5.14.1 Denial of Service

Let us configure the some parts of Pluggable Authentication Modules (PAM) in the file `/etc/security/limits.conf`:

```
* soft nproc 64
* hard nproc 128
* soft nofiles 256
* hard nofiles 1024
```

As documented in the file itself, these lines limit the number of processes and the maximum number of open files per user.

---

<sup>16</sup>It would be mandatory with *lilo*, the version of the Linux loader for the Intel platform.

---

### 5.14.2 Shutdown

Nobody should be allowed to reboot the server just because she has access to the keyboard. The `shutdown.allow` file, when it exists, contains a list of user logins. One of these users need to be logged on a system console when `<CTRL>+<ALT>+<DEL>` is pressed for the reboot to be allowed. If neither `root` or one of those users are logged, the system does not reboot. At least to enable this feature, let us run as root:

```
gcux:~# touch /etc/shutdown.allow
gcux:~# chmod 600 /etc/shutdown.allow
```

### 5.14.3 IP Stack

The Firewall script in section 5.14.4 will further secure the behavior of the stack. This section comes as a baseline for these settings. In `/etc/sysctl.conf`, we set

```
# This is not a router
net.ipv4.ip_forward=0
# Some protection against SYN attacks
net.ipv4.tcp_max_syn_backlog=4096
# Martians packets are logged
net.ipv4.conf.all.log_martians=1
# What Cisco calls RPF,
# Reverse Path Forwarding.
# Accept a packet on an interface only if
# the same interface would be used to send
# traffic to its source.
# OK here because our routing is symmetric
net.ipv4.conf.all.rp_filter=1
# No source routing
net.ipv4.conf.all.accept_source_route=0
net.ipv4.conf.all.send_redirects=0
net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.all.secure_redirects=0
# Same tuning set as default
net.ipv4.conf.default.log_martians=1
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.default.accept_source_route=0
net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.default.accept_redirects=0
net.ipv4.conf.default.secure_redirects=0
```

For further documentation and tuning, we can refer to the documentation provided with the kernel sources in:

```
/usr/src/linux/Documentation/networking/ip-sysctl.txt
```

---

#### 5.14.4 Firewall Rules

Now that we have an brand new kernel 2.4.18, we can setup its firewalling capabilities. The *netfilter* code that implements the 2.4.x,  $x \leq 20$  Firewall on Sparc 64 has a bug that prevents the `limit` rule from working correctly<sup>17</sup>. This issue breaks the log rate limiting functionality. A Kernel patch already exists, but it is simpler and safer to just wait for the next kernel release.

We copy the script provided in section B in the directory `/etc/network/if-pre-up.d` and name it `fw.sh`. The script should be owned by *root* and with permissions set to 700:

```
gcux:/etc/network/if-pre-up.d# chown root:root fw.sh ; chmod 700 fw.sh
```

In the network configuration file `/etc/network/interfaces`, we make sure to start this script as a `pre-up` condition:

```
iface eth0 inet static
    address 192.168.1.2
    netmask 255.255.255.0
    gateway 192.168.1.1
    pre-up /etc/network/if-pre-up.d/fw.sh
```

The script `fw.sh` is written in such a way that it is safe to run it even if the interface is already up. The `interfaces` file makes sure that right before the interface turns up, the firewall rules are applied. In case this script `fw.sh` that configures the firewall fails, the interface is not configured and will not forward packets to the IP stack. The `/etc/default/iptables` script has quite a humorous way of strongly inviting us to use this method.

It is also in the `interfaces` files, `iface` section that we would add our static routes if we needed to. We already specified our default gateway in section 5.9, we find it indicated as `gateway`. The kernel is also aware of the directly connected networks, we do not need to add any static route in our simple configuration. If the lab evolves in a network with routers, we will have to add statements under the `iface` section like

```
up route add -net 172.16.2.0 netmask 255.255.255.0 gw 172.16.1.128
down route del -net 172.16.2.0 netmask 255.255.255.0 \
    gw 172.16.1.128 || true
```

We added the `|| true` to make sure that a failure in the route removal process would not prevent the interface from deconfiguring.

Additional interfaces can be added in a similar fashion. It is to be noted that the interfaces are numbered in the same order as their hardware (MAC) addresses. This default behavior can be changed if desired (either as kernel parameter in `/etc/silo.conf` or module parameter in the `/etc/modutils` directory). We do not mind as long as we are aware of this feature.

---

<sup>17</sup>See <http://lists.netfilter.org/pipermail/netfilter-devel/2003-January/010209.html>.

---

It is also to be noted that all Quad FastEthernet interfaces will have the same hardware address by default. This default setting will become a problem if two ports are on the same LAN (or VLAN). We can override this settings in the OpenProm<sup>18</sup> using:

```
# eeprom local-mac-address\ ?=true
```

A reboot is required for this change to take effect.

We can also add the interface facing the lab network:

```
iface eth1 inet static
    address 172.16.1.1
    netmask 255.255.255.0
```

The interfaces that are to be brought up at boot time are specified by the `auto` keyword:

```
auto lo eth0 eth1
```

The firewall rules implemented by `fw.sh` are quite tight and implement the general policy of *allowing only what is really needed*:

- DNS queries are only allowed when destined to predefined name servers. Debian was at some point vulnerable to a nasty buffer overflow [5] back in 2002:

Buffer overflow in DNS resolver functions that perform lookup of network names and addresses, as used in BIND 4.9.8 and ported to glibc 2.2.5 and earlier, allows remote malicious DNS servers to execute arbitrary code through a subroutine used by functions such as `getnetbyname` and `getnetbyaddr`.

Our distribution server could be compromised during its very first connection to its very first server: its name server.

- Only http, https, ftp and basic network troubleshooting traffic is allowed outbound.
- Nothing but ssh from a management LAN (defined by `MGT_LAN`) is allowed inbound on the external interface.
- On the interface facing the lab, the firewall is wide open. This requirement comes from the security tests that the distribution server is going to perform. Let us imagine that we have just installed a patch on a network device to make sure that the malformed packet *X* does not crash it anymore. Because *X* is malformed, a firewall is likely to reject either the packet or its answer. It is typically what we can observe with the Operating System fingerprint option of `nmap` (`-O`):

---

<sup>18</sup>We can also use the `eeprom` utility in the `sparc-utils` package to configure this variable from Debian itself.

---

```
gcux:~# nmap -O -PS -p 80 192.168.1.1
```

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Warning: OS detection will be MUCH less reliable
because we did not find at least 1 open and 1 closed TCP port
sendto in send_tcp_raw: sendto(3, packet, 60, 0, 192.168.1.1, 16)
=> Operation not permitted
sendto in send_tcp_raw: sendto(3, packet, 60, 0, 192.168.1.1, 16)
=> Operation not permitted
```

## 5.15 TCPWrapper

Debian tends to use the TCPWrapper libraries whenever possible. It is the case for *Openssh* for instance. First, we fix the `/etc/hosts.deny` and deny everything by default:

```
gcux:~# echo "ALL: ALL" >> /etc/hosts.deny
```

As for the `/etc/hosts.allow`, it should contain

```
ALL: 127.0.0.1
sshd: 192.168.2.0/255.255.255.0
```

This way, localhost is allowed to do everything and the management network is allowed to *ssh* to the server. Note that the version of TCPWrapper included in the Debian *testing* release supports the CIDR notation<sup>19</sup>, but as of the *stable* release we still have to enter the full netmask. More details about the Debian releases will be explained in section 5.17.

## 5.16 Inet

Debian keeps the small services running, let us stop them but keep *exim* running to deliver local mails. To achieve this, we comment out all lines in `/etc/inetd.conf` but the one that handles *exim*:

```
gcux:~# cd /etc
gcux:/etc# awk ' !($0 ~ /^(#|smtp)/) { $0 = "#" $0 } { print; } '\
inetd.conf > inetd.conf.new
gcux:/etc# mv inetd.conf.new inetd.conf
gcux:/etc# /etc/init.d/inetd restart
Restarting internet superserver: inetd.
```

It could seem an overkill to keep *inetd* running just for *exim*. However, it may have future uses like running a *tftp* server to upload the Operating System of our network devices. Such a configuration is beyond the scope of this document.

apt-get...	Action
update	Update the list of available packages from the servers indicated in <code>/etc/apt/sources.list</code>
upgrade	Upgrade the installed packages
install <i>package</i>	Install <i>package</i>
remove <i>package</i>	Remove <i>package</i> and keep its configuration files intact <sup>20</sup>

Table 5: *apt-get*

## 5.17 Apt

*apt-get* is the preferred tool to perform Debian updates. Its basic use is described in table 5.

Debian maintains several distributions:

- *stable*, the distribution of choice for production systems. Only security-related updates are performed on this distribution.
- *testing*, a little bit more adventurous. Typically, packages from *unstable* come to *testing* after a little while. The *testing* distribution aims at becoming *stable* during the next major release phase.
- *unstable*, where the actual development takes place on a day to day basis.

Elaborate configurations mixing *stable* and *testing* can be achieved using pin-priority. We will keep a basic form for it, just leaving the door opened for some incursions out of the *stable* branch.

This will be our base `/etc/apt/preferences` file:

```
Package: *
Pin: release a=stable
Pin-Priority: 550

Package: *
Pin: release a=testing
Pin-Priority: 400
```

Setting *stable* at 550 gives it precedence over the default 500 pin priority. This is to ensure that if some day we add external (non Debian) sources, Debian sources will still keep precedence by default.

We will expand a little bit the standard `/etc/sources.list` to handle *testing*. We do not need the CD-ROM anymore. We will put only these lines:

<sup>19</sup>The network would then be noted /24.

---

```
# Security updates
deb http://security.debian.org stable/updates main contrib non-free
deb http://security.debian.org testing/updates main contrib non-free

# Normal distribution
deb ftp://ftp.us.debian.org/debian stable main contrib non-free
deb ftp://ftp.us.debian.org/debian testing main contrib non-free

# Non-US
deb ftp://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb ftp://non-us.debian.org/debian-non-US testing/non-US main contrib non-free
```

Note that we have chosen the nearest mirror (our lab is in the US). The cryptography model makes the distribution independent from the mirror we picked.

Note also that we should not use *dselect* from now on, but *aptitude*. The preferences file is configured in such a way that if a package exists in both *Testing* and *Stable*, then the *Stable* version will be used. *apt-get* features command-line options to force either *Testing* or *Stable* packages. This approach is conservative: by default, we will stay in the *Stable* branch.

`/etc/apt/preferences` can also be tuned further to force certain packages in a given distribution. This feature will be useful for packages that need to be more recent than what *stable* has to offer. Section 5.23 will go into more details.

## 5.18 PGP key

For a task of purely checking PGP keys, creating a private key pair for users on the distribution server is not mandatory. However, doing so can start a web of trust among distribution servers within our company.

Entering an independent web of trust is an initial investment and has a cost as well as a risk. If we decide to deploy a few distribution servers throughout the company, creating a key for the users of these servers makes it possible to distribute the software authentication and assessment work. Figure 2 shows an embryo of web of trust that we will eventually build from this document.

The version of *GnuPG* may vary depending on the version of Debian 3.0 that is used. Even *GnuPG* has its vulnerabilities, this package is actively updated. First, a blank test that creates the directory and the configuration files:

```
user@gcux:~$ gpg
gpg: /home/user/.gnupg: directory created
gpg: /home/user/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options file
```

This first step is not necessary in all versions of *GnuPG*. It is not in the version included in Debian 3.0 r0 (1.0.6). It is, however, for more recent versions (like version 1.2.1).

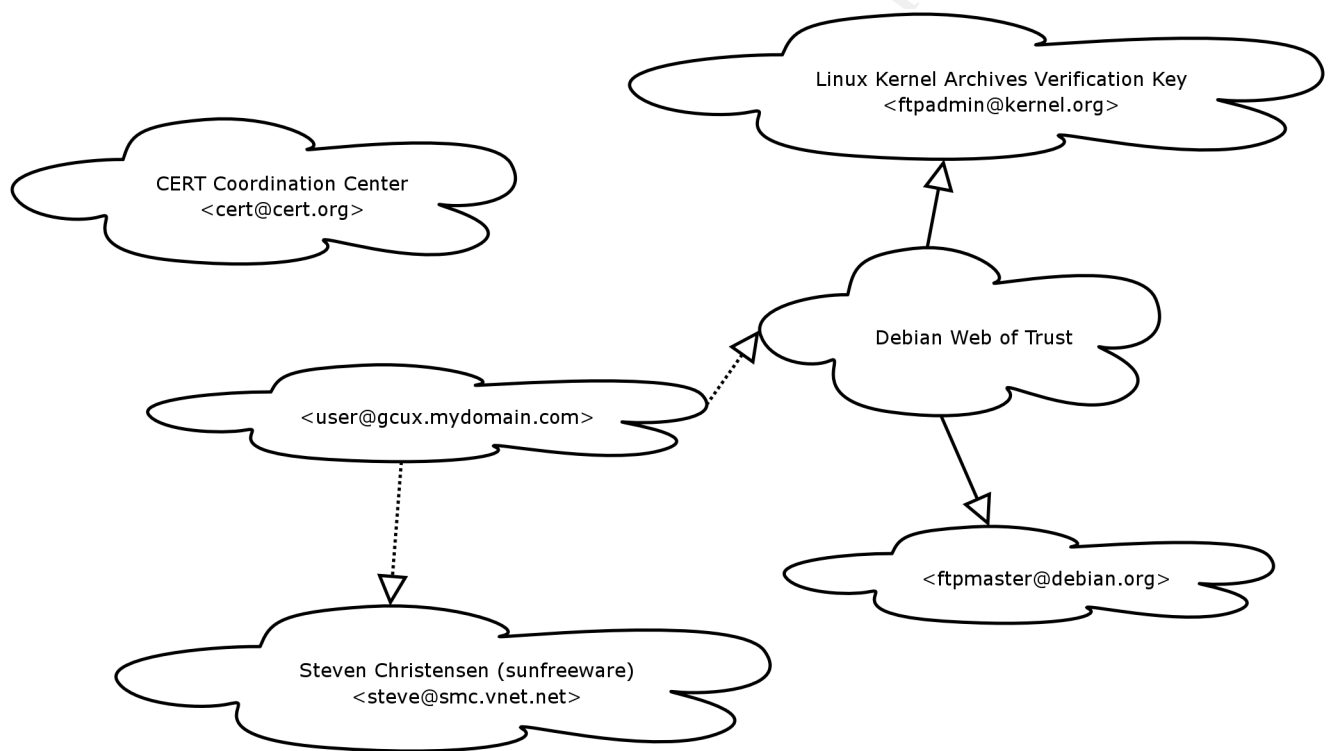


Figure 2: Initial web of trust



---

We can straight away indicate a key server so that we avoid to have to repeatedly use the option `--keyserver`:

```
test@gcux:~$ echo "keyserver wwwkeys.pgp.net" >> .gnupg/gpg.conf
```

In more recent versions of *GnuPG*, the individual configuration file is no longer in `~/.gnupg/gpg.conf` but in `~/.gnupg/options`.

It is time to generate the key pair:

- We want DSA and ElGamal keys<sup>21</sup>;
- Keysize of 2048<sup>22</sup>;
- Key never expires<sup>23</sup>.

```
user@gcux:~$ gpg --gen-key
gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.
```

```
gpg: /home/user/.gnupg/secring.gpg: keyring created
gpg: /home/user/.gnupg/pubring.gpg: keyring created
Please select what kind of key you want:
```

- (1) DSA and ElGamal (default)
- (2) DSA (sign only)
- (4) ElGamal (sign and encrypt)

Your selection? **1**

DSA keypair will have 1024 bits.

About to generate a new ELG-E keypair.

minimum keysize is 768 bits

default keysize is 1024 bits

highest suggested keysize is 2048 bits

What keysize do you want? (1024) **2048**

Requested keysize is 2048 bits

Please specify how long the key should be valid.

0 = key does not expire

<n> = key expires in n days

<n>w = key expires in n weeks

<n>m = key expires in n months

<n>y = key expires in n years

Key is valid for? (0) **0**

Key does not expire at all

Is this correct (y/n)? **y**

---

<sup>21</sup>DSA keys are used for signature and ElGamal keys for encryption.

<sup>22</sup>1024 is said to be enough as of today, we take a security margin.

<sup>23</sup>You should refer to your company policy for this aspect of key expiration.

---

You need a User-ID to identify your key; the software constructs the user id from Real Name, Comment and Email Address in this form:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: **Test User**

Email address: **user@gcux**

Comment: **Test User for GCUX practical**

You selected this USER-ID:

"Test User (Test User for GCUX practical) <user@gcux>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? **O**

You need a Passphrase to protect your secret key.

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

```
+++++.....+++++.....+++++.....+++++.....+++++
.+++++.....+++++.....+++++.....+++++.....+++++>
+++++.....+++++
```

Not enough random bytes available. Please do some other work to give the OS a chance to collect more entropy! (Need 170 more bytes)

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

```
+++++.....+++++.....+++++.....+++++.....+++++>
+++++.....+++++.....+++++.....+++++.....+++++>
+++++>+++++.....>+++++.....
.....+++++^^^
```

gpg: /home/user/.gnupg/trustdb.gpg: trustdb created

public and secret key created and signed.

**key marked as ultimately trusted.**

pub 1024D/EE727807 2003-05-09 Test User (Test User for GCUX practical) <user@gcux>

Key fingerprint = D804 E4CE 668A 2FB0 D96D CE41 92D9 013C EE72 7807

sub 2048g/8C1F892E 2003-05-09

Unlike the signatures, the trust database is not stored with the key. Ultimately trusting a key means trusting its choices as if they were ours, which should obviously be the case for the key we just created.

We should also generate a revocation certificate for the key. This way, in the case we loose our private key (forgetting the passphrase for instance) we can revoke it. It is difficult to know before the fact for what particular reason we may want to revoke the key. We choose the most serious reason: *because it has been compromised*.

user@gcux:~\$ **gpg --gen-revoke Test User**

---

sec 1024D/34B857D0 2003-05-23 Test User (Test User for GCUX practical) <user@gcux>

Create a revocation certificate for this key? **yes**

Please select the reason for the revocation:

- 0 = No reason specified
- 1 = Key has been compromised
- 2 = Key is superseded
- 3 = Key is no longer used
- Q = Cancel

(Probably you want to select 1 here)

Your decision? **1**

Enter an optional description; end it with an empty line:

> **Test Line**

>

Reason for revocation: Key has been compromised

Test Line

Is this okay? **yes**

You need a passphrase to unlock the secret key for

user: "Test User (Test User for GCUX practical) <user@gcux>"

1024-bit DSA key, ID 34B857D0, created 2003-05-23

ASCII armored output forced.

Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets access to this certificate he can use it to make your key unusable. It is smart to print this certificate and store it away, just in case your media become unreadable. But have some caution: The print system of your machine might store the data and make it available to others!

-----BEGIN PGP PUBLIC KEY BLOCK-----

Version: GnuPG v1.2.1 (GNU/Linux)

Comment: A revocation certificate should follow

iFIEIBECABIFaj7NmyELHQJUZxN0IExpbmUACgkQCJlpBTS4V9BmkACfRnKnAjIN

VuustQ6bRn4Umv8/B08An0tHi2/RPVsoj69QKQdz7RNol0bv

=TWbq

-----END PGP PUBLIC KEY BLOCK-----

The `addrevoker` command of the *GnuPG* key edition menu makes it also possible to specify a revoker for the key we just created. Who should keep this revocation certificate, who else should decide to publish it, who should act as a revoker are many questions that need to be answered as part of a company-wide policy.

## 5.19 Upgrade

Right before the physical connection, a final check of the services we are offering:

---

```
gcux:~# lsof -i
COMMAND PID USER   FD   TYPE DEVICE SIZE NODE NAME
inetd    177 root    lu    IPv4    123          TCP *:smtp (LISTEN)
```

Now comes the time of connecting to the network. We make sure that we plugged the network cables in the correct interfaces. We bring the interface back up (we shut it down in 5.12): `ifup eth0`. We make sure that we can ping our default gateway, and that the name resolution is working (e.g. by launching `host security.debian.org`).

## 5.20 Debian Master Keys

As of Debian 3.0, the only (partially) implemented package signature check relies on Release files. There are plans to move forward and add a per-package signature scheme. As it had not reached a satisfactory level of development by the time Debian 3.0 has been release, it is currently not used and all signatures are stripped off when packages enter the Debian distribution servers<sup>24</sup>.

Let us go back to the Release files. Each Debian distribution (or more precisely each URI in `/etc/apt/sources.list`) includes a Release file that contains the list and location of all Release, Packages, Packages.gz, Sources and Sources.gz files provided by this URI. A set of these files defines a component of the distribution (the first block after the URI in the `/etc/apt/sources.list` file).

It also contains a MD5 and a SHA1 checksum of all these Release, Packages and Source files to ensure their authenticity.

- Each Packages file contains the list of the packages referring to this component, along with the packaging information (location, versions, dependencies, description) and MD5 checksum to ensure the authenticity of the package;
- Each Sources file has the same role as the Packages file but for source packages<sup>25</sup>;
- Each Release file contains the the component information.

If we follow the security association:

1. `apt-get` downloads automatically the Packages files, the Release files and the packages themselves.
2. At installation time, `apt-get` installs a package because the checksum of the package matches the checksum information in the Packages file.
3. `apt-get` trusts the Packages file because it trusts the Release file.

---

<sup>24</sup>See <http://www.debian.org/doc/manuals/securing-debian-howto/ch7.en.html#s-deb-pack-sign>.

<sup>25</sup>A source package is a package that can be compiled to get a binary package.

4. We trust the `Release` file because we trust its signature in `Release.gpg`. This check can be automated using the `apt-release-check` file as provided in section [D](#). We placed this script in `/usr/local/bin`, with permissions of 755, owned by `root`.
5. We trust the `Release.gpg` signature, or in fact we will at the end of section [5.21](#).

## 5.21 Release File

We are going to check the `Release` file once, to understand how things work. To avoid doing this as `root` (and because it would not bring anything), let us operate as the user `user`. First, download a test file and its signature:

```
user@gcux:~$ wget http://security.debian.org/dists/stable/updates/Release
--08:39:30-- http://security.debian.org/dists/stable/updates/Release
=> 'Release'
Resolving security.debian.org... done.
Connecting to security.debian.org[194.109.137.218]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18,456 [text/plain]

100%[=====>] 18,456
      9.44K/s   ETA 00:00

08:39:33 (9.44 KB/s) - 'Release' saved [18456/18456]

user@gcux:~$ wget http://security.debian.org/dists/stable/updates/Release.gpg
--08:40:04-- http://security.debian.org/dists/stable/updates/Release.gpg
=> 'Release.gpg'
Resolving security.debian.org... done.
Connecting to security.debian.org[194.109.137.218]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 197 [text/plain]

100%[=====>] 197
      192.38K/s   ETA 00:00

08:40:05 (192.38 KB/s) - 'Release.gpg' saved [197/197]
```

Then, we can try to check the signature. The original configuration for *GnuPG* has been performed in section [5.18](#).

```
user@gcux:~$ gpg --verify Release.gpg Release
gpg: Signature made Mon Dec 16 12:45:49 2002 CST using DSA key ID 722F1AED
gpg: Can't check signature: public key not found
gpg: Signature made Mon Jan 27 04:25:14 2003 CST using DSA key ID 38C6029A
gpg: Can't check signature: public key not found
```

We have to import this key because it is not in the Debian keyring<sup>26</sup>:

```
user@gcux:~$ gpg --recv-keys 38C6029A
gpg: key 38C6029A: public key "Debian Archive Automatic Signing Key (2003)
<ftpmaster@debian.org>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

The only keys we have so as to trust this *Debian Archive Automatic Signing Key* is in the Debian Keyring, provided by the *debian-keyring* package installed from the CD:

```
user@gcux:~$ gpg --keyring /usr/share/keyrings/debian-keyring.gpg \
--check-sigs 38C6029A
pub 1024D/38C6029A 2002-12-20 Debian Archive Automatic Signing Key (2003)
<ftpmaster@debian.org>
sig!3 38C6029A 2002-12-20 Debian Archive Automatic Signing Key (2003)
<ftpmaster@debian.org>
sig! 7172DAED 2002-12-20 Anthony Towns <ajt@debian.org>
```

We can trust the key belonging to Anthony Towns, because it comes from the Debian CD that we decided to trust. We insert it in our keyring:

```
user@gcux:~$ gpg --keyring /usr/share/keyrings/debian-keyring.gpg \
-a --export 7172DAED | gpg --import
gpg: key 7172DAED: public key "Anthony Towns <ajt@debian.org>" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

Leaving this key in its original location would make it impossible for non-root users to sign it (no write access to this keyring!). We decide to sign it locally, to show that we trust the key as belonging to Anthony Towns but without giving the opportunity to export this signature to a key server:

```
user@gcux:~$ gpg --lsign 7172DAED

gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
pub 1024R/7172DAED created: 1996-06-15 expires: never trust: -/-
(1) Anthony Towns
(2) Anthony Towns <aj@humbug.org.au>
(3) Anthony Towns <aj@mailbox.uq.edu.au>
(4) Anthony Towns <ajt@acm.org>
(5) Anthony Towns <aj@star.brisnet.org.au> *EXPIRED*
(6) Anthony Towns <aj@azure.humbug.org.au>
(7) Anthony Towns <s343676@student.uq.edu.au>
(8) Anthony Towns <aj@star.brisnet.org.au>
(9) Anthony Towns <aj@s343676.slip.cc.uq.edu.au>
```

<sup>26</sup>We only pay attention to the most recent signature.

---

(10). Anthony Towns <ajt@debian.org>

Really sign all user IDs? **yes**

pub 1024R/7172DAED created: 1996-06-15 expires: never trust: -/-  
Primary key fingerprint: 70 B6 04 9B A2 C8 7D AA 00 5D DC 82 58 9D 49 6E

Anthony Towns  
Anthony Towns <aj@humbug.org.au>  
Anthony Towns <aj@mailbox.uq.edu.au>  
Anthony Towns <ajt@acm.org>  
Anthony Towns <aj@star.brisnet.org.au> \*EXPIRED\*  
Anthony Towns <aj@azure.humbug.org.au>  
Anthony Towns <s343676@student.uq.edu.au>  
Anthony Towns <aj@star.brisnet.org.au>  
Anthony Towns <aj@s343676.slip.cc.uq.edu.au>  
Anthony Towns <ajt@debian.org>

How carefully have you verified the key you are about to sign actually belongs to the person named above? If you don't know what to answer, enter "0".

- (0) I will not answer. (default)
- (1) I have not checked at all.
- (2) I have done casual checking.
- (3) I have done very careful checking.

Your selection? **2**

Are you really sure that you want to sign this key  
with your key: "Test User (Test User for GCUX practical) <user@gcux>"

The signature will be marked as non-exportable.

I have done casual checking.

Really sign? **yes**

You need a passphrase to unlock the secret key for  
user: "Test User (Test User for GCUX practical) <user@gcux>"  
1024-bit DSA key, ID 1123481F, created 2003-05-09

Now that we declared that this key is valid, we can trust it to sign other keys:

user@gcux:~\$ **gpg --edit-key 7172DAED**  
gpg (GnuPG) 1.2.1; Copyright (C) 2002 Free Software Foundation, Inc.  
This program comes with ABSOLUTELY NO WARRANTY.  
This is free software, and you are welcome to redistribute it  
under certain conditions. See the file COPYING for details.

---

```
pub 1024R/7172DAED  created: 1996-06-15 expires: never      trust: -/f
(1) Anthony Towns
(2) Anthony Towns <aj@humbug.org.au>
(3) Anthony Towns <aj@mailbox.uq.edu.au>
(4) Anthony Towns <ajt@acm.org>
(5) Anthony Towns <aj@star.brisnet.org.au> *EXPIRED*
(6) Anthony Towns <aj@azure.humbug.org.au>
(7) Anthony Towns <s343676@student.uq.edu.au>
(8) Anthony Towns <aj@star.brisnet.org.au>
(9) Anthony Towns <aj@s343676.slip.cc.uq.edu.au>
(10). Anthony Towns <ajt@debian.org>
```

Command> **trust**

```
pub 1024R/7172DAED  created: 1996-06-15 expires: never      trust: -/f
(1) Anthony Towns
(2) Anthony Towns <aj@humbug.org.au>
(3) Anthony Towns <aj@mailbox.uq.edu.au>
(4) Anthony Towns <ajt@acm.org>
(5) Anthony Towns <aj@star.brisnet.org.au> *EXPIRED*
(6) Anthony Towns <aj@azure.humbug.org.au>
(7) Anthony Towns <s343676@student.uq.edu.au>
(8) Anthony Towns <aj@star.brisnet.org.au>
(9) Anthony Towns <aj@s343676.slip.cc.uq.edu.au>
(10). Anthony Towns <ajt@debian.org>
```

Please decide how far you trust this user to correctly  
verify other users' keys (by looking at passports,  
checking fingerprints from different sources...)?

- 1 = Don't know
- 2 = I do NOT trust
- 3 = I trust marginally
- 4 = I trust fully
- 5 = I trust ultimately
- m = back to the main menu

Your decision? **4**

```
pub 1024R/7172DAED  created: 1996-06-15 expires: never      trust: f/f
(1) Anthony Towns
(2) Anthony Towns <aj@humbug.org.au>
(3) Anthony Towns <aj@mailbox.uq.edu.au>
(4) Anthony Towns <ajt@acm.org>
(5) Anthony Towns <aj@star.brisnet.org.au> *EXPIRED*
(6) Anthony Towns <aj@azure.humbug.org.au>
(7) Anthony Towns <s343676@student.uq.edu.au>
(8) Anthony Towns <aj@star.brisnet.org.au>
(9) Anthony Towns <aj@s343676.slip.cc.uq.edu.au>
(10). Anthony Towns <ajt@debian.org>
```



---

Please note that the shown key validity is not necessarily correct unless you restart the program.

Command> **quit**

We are now fully equipped to check the Release file:

```
user@gcux:~$ gpg --verify Release.gpg Release
gpg: Signature made Mon Dec 16 12:45:49 2002 CST using DSA key ID 722F1AED
gpg: Can't check signature: public key not found
gpg: Signature made Mon Jan 27 04:25:14 2003 CST using DSA key ID 38C6029A
gpg: Good signature from "Debian Archive Automatic Signing Key (2003)"
    <ftpmaster@debian.org>
gpg: checking the trustdb
gpg: checking at depth 0 signed=1 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: checking at depth 1 signed=1 ot(-/q/n/m/f/u)=0/0/0/0/1/0
gpg: checking at depth 2 signed=0 ot(-/q/n/m/f/u)=1/0/0/0/0/0
gpg: next trustdb check due at 2004-01-24
```

*apt-release-check* as reproduced in section D uses a simplified version of the key trusting algorithm. It is implemented by *gpgv* versus the one explained in section 5.2 that is implemented by *gpg*. This simplified version just trusts all the keys in `~/.gnupg/trustedkeys.gpg`. We are entitled to initialize this special keyring with the *Debian Archive Automatic Signing Key (2003)*:

```
user@gcux:$ gpg --export 38C6029A > ~/.gnupg/trustedkeys.gpg
```

## 5.22 Software Updates

Everything is now in place to perform the actual updates.

1. First step is performed as *root*:

```
gcux:~# apt-get update
Hit http://security.debian.org stable/updates/main Packages
[...]
Fetched 2312kB in 11m6s (3467B/s)
Reading Package Lists... Done
Building Dependency Tree... Done
```

This command updates the local database of all packages that are available in the various URI we entered in the `sources.list` file.

2. Second step is performed as *user* (swapping between UIDs is a little bit cumbersome, but *sudo* will help make it simpler in operational mode):

---

```
user@gcux:$ apt-release-check
```

```
Checking sources in /etc/apt/sources.list:
```

```
~~~~~
```

You should take care to ensure that the distributions you're downloading are the ones you think you are downloading, and that they are as up to date as you would expect (testing and unstable should be no more than two or three days out of date, stable-updates no more than a few weeks or a month).

```
Source: deb http://security.debian.org stable/updates main contrib non-free
  o Origin: Debian/Debian-Security
  o Suite: stable/woody
  o Date: Fri, 16 May 2003 00:18:00 UTC
  o Description: Debian 3.0 Security Updates
  o Signed by: Debian Archive Automatic Signing Key (2003) <ftpmaster@debian.org>
  o Okay: main contrib non-free
```

```
[...]
```

```
Results
```

```
~~~~~
```

*Everything seems okay!*

### 3. Last step is performed by root:

```
gcux:~# apt-get upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
```

The output really depends on how outdated the packages on our CD are compared to the one of the current Debian release. If `apt-get` complains because packages have been kept back, use

```
gcux:~# apt-get dist-upgrade
```

This last step compares the recently updated list of packages provided by the `sources.list` file with the database of the packages that are actually installed. It then upgrades the packages, following the rules we put in the `preferences` file, if any (see section 5.17).

## 5.23 Additional Software

Let us install the software we are missing.

---

```
gcux:~# apt-get install sudo aptitude apache harden-doc \
nmap nessus nessusd ssh libpam-cracklib ntp tiger logcheck ippl
```

We can check from the output that all the packages come from *Stable*. *Apache* is more than likely to come from the security updates site.

The pre-installation scripts ask some questions, the next sections will give the appropriate answers along with configuration tuning. Note that we can reconfigure a package at any time with

```
dpkg-reconfigure package
```

### 5.23.1 Sudo

We use `visudo` to add the following line to the configuration of *sudo*:

```
user ALL=(ALL) ALL
```

There are at least two reasons to use `visudo` versus `vi /etc/sudoers`:

- **Concurrent access:** Potentially two or more super users are likely to modify the `/etc/sudoers` file simultaneously. Depending on when these users save their work, the resulting file might be something that fits none of the users' needs.
- **Configuration check:** A configuration check is performed by `visudo` before actually updating `/etc/sudoers`. A syntax error in the `/etc/sudoers` file would lock the use of *sudo* for all users. Only *root* can then fix the situation (back to the sealed envelope).

The purpose is clearly to avoid multiplying these all-mighty users. However, we will need at least one of those once the *root* password is buried in some secure place. Demoting such a *sudo*-powered *superuser* can then be performed at the *sudo* level, we do not have to worry about updating the *root* password anymore. Provided that *user* does not `sudo su -`, all her actions are also logged. We are really talking about convenience: anyone with super user privileges can leave backdoors opened to get back in the system anyway.

### 5.23.2 Ippl

*ippl* does not much but log the connection attempts. It is quite handy because we do not have to change the rules of the firewall to enable the logging of IP events. *ippl* is usually rather verbose and put everything in `/var/log/syslog`. Since our lab facing interface might generate a lot of traffic, we uncomment this line (remove the `#` sign placed at the beginning of the line) in the `/etc/ippl.conf` file:

```
log-in all /var/log/ippl/all.log
```

---

Sending logs to a separate file will make the correlation of events a little bit less easy. However, it will help to keep a clean `syslog` file. Log rotation is managed automatically, using a script called `ippl` in `/etc/logrotate.d`.

Depending on the lab traffic, we might decide to also log UDP queries. To achieve this, we uncomment these lines:

```
run udp
log-in udp /var/log/ippl/udp.log
ignore udp from localhost
ignore udp port domain
ignore udp srcport domain
```

UDP broadcast packets are far too frequent to be logged, we ignore them:

```
ignore udp to 255.255.255.255
ignore udp to 172.16.1.255
ignore udp to 192.168.1.255
```

`ippl` enables us to know most of what happens on the lab facing interface without having to use the Firewall (which is not available for this interface). After these changes, we restart the `ippl` daemon:

```
user@gcux:~$ sudo /etc/init.d/ippl restart
Stopping IP Protocols Logger.
Starting IP Protocols Logger: ippl.
```

### 5.23.3 Network Time Protocol

As the distribution server is definitely oriented toward the lab network, we will make it use the NTP server of the lab network. Generally speaking, our lab has a NTP server because it aims at reflecting the production network. The question we are asked at the installation of the package is:

- Specify NTP time servers: **172.16.1.2** (if our lab NTP server has this IP address)

This particular aspect of the NTP configuration should be changed using

```
user@gcux:~$ sudo dpkg-reconfigure ntp-simple
```

We need to restart NTP after making the changes:

```
user@gcux:~$ sudo /etc/init.d/ntp restart
Restarting NTP server: ntpd... done.
```

All other aspects can be freely changed in `/etc/ntp.conf`. As usual,

```
user@gcux:~$ zless /usr/share/doc/ntp/README.Debian.gz
```

gives Debian specific information about this package.

The rotation of the log files is handled automatically using `cron` and the file

```
/etc/cron.daily/ntp-simple
```

---

### 5.23.4 Ssh

As for the initial questions the *ssh* package asks:

- <Yes>, we want to be running only protocol version 2.
- <Yes>, privilege separation is a good thing.
- <Yes>, installing `/usr/lib/ssh-keysign` SUID *root* is fine.
- <Yes>, we want to run the `sshd` server.

We can then tune the `/etc/ssh/sshd_config` file:

- `Protocol 2` to force protocol 2 only.
- `PermitRootLogin no` to deny *root* the right to log in directly.
- `PasswordAuthentication no`, public key authentication is preferred.
- `ListenAddress 192.168.1.2`, no reason to have *ssh* listening on the lab facing interface(s).
- `DenyUsers root daemon bin sys sync games man lp mail news uucp proxy postgres www-data backup operator list irc gnats nobody`, all those users do not need to log in.

We can restart the *ssh* server now:

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

To create an *ssh* DSA keypair, we execute on the client system:

```
user@client:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
a5:10:2f:49:31:87:01:72:49:cf:f4:e1:78:59:77:39 user@client
```

Then, we can copy `/home/user/.ssh/id_dsa.pub` from *client* to

`/home/user/.ssh/authorized_keys`

on *gcux* (or copy-paste from a console).

---

### 5.23.5 Console Access

It is definitely convenient to have a serial console access to the distribution server. However, it is also dangerous. It makes dictionary attacks theoretically possible, although unlikely (it is a very slow process and very, very visible if the administrator watches the logs). We will also make the assumption that the console server is adequately hardened<sup>27</sup> and that it is accessed using a secure protocol.

As usual, we will make the life of an attacker a little bit more complicated, and make her compromise a user account before trying to compromise the *root* account. */etc/securetty* contains the list of terminals on which *root* is allowed to login. Let us forbid *root* to log in from the console:

```
gcux:~# awk ' ($1 ~ /^console$/) { $1 = "#console" } { print } ' \
/etc/securetty > /etc/securetty.new \
&& mv /etc/securetty.new /etc/securetty
```

To let people know that this is not a public system that can be used for any purpose, the usual recommended banners should be placed in */etc/issue*, */etc/issue.net* and at the end of */etc/motd*.

### 5.23.6 Cracklib

Cracklib is a library to enforce stricter policies when it comes to choosing passwords. It can be used in conjunction with PAM:

- */etc/pam.d/ssh*: Comment out (add a # sign at the beginning of the line)

```
password    required    pam_unix.so
```

as suggested and uncomment (remove the # sign placed at the beginning of the lines)

```
password required    pam_cracklib.so retry=3 minlen=6 difok=3
password required    pam_unix.so use_authtok nullok md5
```

- */etc/pam.d/passwd*: Comment out

```
password    required    pam_unix.so nullok obscure min=4
```

as suggested and uncomment

---

<sup>27</sup>Note also that certain combination of *getty*, Linux kernel and Sun hardware misbehaves when it has both no keyboard and no console. *init* keeps respawning and eventually locks the server. Patches exist that either stop *getty* or check the hardware setup and stop the respawns.

---

```
password required      pam_cracklib.so retry=3 minlen=6 difok=3
password required      pam_unix.so use_authtok nullok md5
```

- /etc/pam.d/login: Comment out

```
password    required    pam_unix.so nullok obscure min=4
```

as suggested and uncomment

```
password required      pam_cracklib.so retry=3 minlen=6 difok=3
password required      pam_unix.so use_authtok nullok md5
```

The changes implement a tighter password policy.

It is rather difficult to decide at what point our password policy stops to increase security and starts to be counter-productive. Once we have reached this point, even security aware users start to put the same password at different places or even write them down somewhere (be it in their PDA or in a sheet of paper in their wallet).

### 5.23.7 Apache

We can first notice that we chose to install *Apache* and not *Apache-ssl*. Even if we had decided to put less trust in the lab network, we would not have used *Apache-ssl*. *GnuPG* would have been preferred to ensure the authenticity of the distributed software downloaded on the end devices. Indeed, *Apache-ssl* would start a Public Key Infrastructure, whereas the trust model we have deployed so far is based on a web of trust. Getting one of these two infrastructures deployed and fully understood is complex enough, we will stick with the web of trust.

The *Apache* install script does not ask any question. Before any change, let us prepare the home directories:

```
user@gcux:~$ chmod 751 .
user@gcux:~$ mkdir public_html
user@gcux:~$ sudo chown user:www-data public_html
user@gcux:~$ chmod 750 public_html
```

This way, *www-data* (the user *Apache* runs under) has only the permission to *enter* in the home directories. It can still read the entire content of the `public_html` subdirectory.

The log rotation is handled by *logrotate* and configured in `/etc/logrotate.d/apache`. In the default configuration, logs are rotated weekly, compressed and kept for one year. They are archived and owned by `root:adm` with permissions `640`, which is fine.

In the reasonably recent versions of *Apache*, the entire configuration of *httpd* is in `/etc/apache/httpd.conf`. We need some changes for our distribution server:

- `BindAddress 172.16.1.1` so that Apache binds on the internal interface only.

---

Debian comes with *apacheconfig*. This program checks the *Apache* configuration and tunes the `LoadModule` directives accordingly. It also automatically archives the configuration files and restarts *Apache* after changes. It is time to launch it:

```
user@gcux:~$ sudo apacheconfig
```

The default configuration uses the module *userdir*. It perfectly fits in our model, where users publish the software on the web site, after checking their validity.

First, Debian specifies the directory in the users' homes that will be used:

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

Then, the restrictions and functionalities:

```
1  <Directory /home/*/public_html>
2      AllowOverride FileInfo AuthConfig Limit
3      Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
4      <Limit GET POST OPTIONS PROPFIND>
5          Order allow,deny
6          Allow from all
7      </Limit>
8      <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
9          Order deny,allow
10         Deny from all
11     </Limit>
12 </Directory>
```

**Line 1** specifies the Unix directory where the configuration applies. This way, no matter what *location* is used to access this directory, these restrictions will apply.

**Line 2** allows some local and very limited override of the global Apache configuration.

**Line 3** allows Apache to follow symbolic links if link and file belong to the same owner. It also disables the execution of Server Side Includes.

**Lines 4-7** allow read-only actions on these directories.

**Lines 8-11** explicitly forbid write actions.

This configuration gives a completely read-only access.

The URL `http://172.16.1.1/~user` from any lab machine then gives access to the software *user* wants to distribute. The directory is browsable, which makes maintenance easier for *user*. She could create a directory structure per vendor, and then per software to keep different versions. The list of software on the web page comes with icons to help identifying the files. Per user customization is also possible:



- A file `HEADER.html` or `HEADER.txt` can be added in each directory under `~/public_html`. It will be prepended by Apache itself to the list of software.
- A file `README.html` or `README.txt` can be added in each directory under `~/public_html`. It will be appended by Apache itself to the list of software.

The default page in `/var/www` can be customized if needed, for example to give the list of users that distribute software.

### 5.23.8 Nessus

The purpose of the distribution server is to provide a trusted source of software. Once software has been installed or patches applied, it is good to assess the remaining vulnerabilities, if any. *Nessus* is a great tool for this purpose.

Installing the `nessus` client in addition to the `nessusd` server triggers the installation of `xfree86-common` along with basic graphical libraries. It is not an issue in itself because the server components of X are not part of this package.

We will be able to launch the *Nessus* client on the distribution server itself using `ssh` and its `-X` flag. There is no real point in keeping the *Nessus* server running indefinitely, we can configure `sudo` to let people start the daemon on demand. Using `visudo`, we add these lines:

```
User_Alias      AUDIT = otheruser
Cmnd_Alias      NESSUS = /usr/sbin/nessusd "", /usr/bin/killall nessusd ""

AUDIT    ALL=NESSUS
```

where `otheruser` would be another user<sup>28</sup> that we would have created with

```
user@gcux:~$ sudo adduser otheruser
```

The `" "` as argument in `/etc/sudoers` specifies that the `AUDIT` users cannot use any additional argument when launching `nessusd` or when killing it.

Our configuration of *TCP Wrapper* from section 5.15 does not allow `nessusd` to be queried from anywhere else than localhost itself.

We need then to add a *Nessus* user:

```
gcux:~# nessus-adduser
Generating primes: .....q.....pg
Using /var/tmp as a temporary file holder

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

<sup>28</sup>user has already full privileges from `sudo`.

---

Login : **user**  
Authentication method (cipher/plaintext) [cipher] : **cipher**  
Is "user" a local user on this machine [y|n]? **y**

Ok, treating user "user" as a local user.

User rules

-----

nessusd has a rules system which allows you to restrict the hosts that user 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)

**deny 172.16.1.1**

**accept 172.16.1.0/24**

**default deny**

Login : user  
Auth. method : cipher, local user connecting from 127.0.0.1

Rules :  
deny 172.16.1.1  
accept 172.16.1.0/24  
default deny

Is that ok ? (y/n) [y] **y**  
Generating the user key for "user" (please be patient)  
Generating primes: .....;  
Retrying: .....pg

To protect your private key just generated, enter your personal pass phrase, now. Keep that pass phrase secret. And each time when you restart `nessus`, re-enter that pass phrase when you are asked, for. This prevents anybody else from logging in to the `nessus` server using your account.

The drawback of a pass phrase is that it will prevent you from being able to use `nessus(1)` in a cron job or in a quiet script. If you do not want to use a pass phrase, enter a blank one.

To change or remove the pass phrase, later on read in the manual page `nessus(1)` about the `-C` option.

---

```
New pass phrase:
Repeat          :
Pass phrase:
user added.
```

The user will be allowed to scan the lab network and only the lab network, with the exception of the lab facing interface on the distribution server. If more interfaces were to be used, they should be added to the deny list.

### 5.23.9 Getting Newer Versions

There are several ways to get a version of a package that is more recent than the one we can find in the *Stable* distribution. This section will focus on the case of *Nessus* version 2.x (provided by *Testing*), although the rest of the document will assume that we are using the 1.x branch of *Nessus* (provided by *Stable*).

We will also assume that *Nessus* 1.x and all its direct dependencies (including libraries) installed in section 5.23.8 have been removed from the server, using the `remove` option of `apt-get`.

There are different approaches to keep *Nessus* up-to-date:

- We can stick with the *Stable* version of the *Nessus* package and only update the plug-ins using the script provided with *Nessus*, *nessus-update-plugins*. We will recommend against this because no authentication check is performed, as documented in the script itself:

```
user@gcux:~$ head /usr/sbin/nessus-update-plugins
#!/bin/sh
#
# nessus-update-plugins
#
# This script will retrieve all the newest plugins from
# www.nessus.org.
#
# NOTE: the use of this script is dangerous as the authenticity of
#       the scripts is not checked for. USE THIS SCRIPT WITH CAUTION
#
```

- We can download the tarball from the upstream developer. We would first install with *apt-get* the development libraries required to build *Nessus*, then the *Nessus* tarball under `~/src` and install it under `/usr/local`. This option is not consistent with our choice of using a distribution. In particular, the package dependencies are not managed anymore.
- We can choose to install the Debian package from the *Testing* distribution. We have to add these lines to the `/etc/apt/preferences` file:

---

```
Package: nessusd
Pin: release a=testing
```

```
Package: nessus
Pin: release a=testing
```

```
Package: nessus-plugins
Pin: release a=testing
```

This way, a much higher priority (990 versus around 500) is set to the *Testing* version of these packages.

The risk here is to install not only the direct dependencies of *Nessus* from *Testing*, but also the *libc6* package. Most of the software running on Linux are dynamically linked against the C library. We definitely want to keep control on its version and its quality.

The version of *libc6* that *Nessus* will require definitely depends on the current development cycle phase of *Testing*.

We have two ways to implement this choice to update *Nessus*, using *apt-get*, or one of its front-end, *Aptitude*.

- Using *apt-get*, we might have an error message similar to

```
user@gcux:~$ sudo apt-get install nessus nessusd
Reading Package Lists... Done
Building Dependency Tree... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

Sorry, but the following packages have unmet dependencies:
 nessus: Depends: libc6 (>= 2.3.1-1) but 2.2.5-11.5 is to be installed
          Depends: libnessus2 (>= 2.0.5) but it is not going to be installed
          Depends: libssl0.9.7 but it is not going to be installed
 nessusd: Depends: libc6 (>= 2.3.1-1) but 2.2.5-11.5 is to be installed
          Depends: libnsl2 (>= 2.0.5) but it is not going to be installed
          Depends: libnessus2 (>= 2.0.5) but it is not going to be installed
          Depends: libssl0.9.7 but it is not going to be installed
          Depends: nessus-plugins (>= 1.3) but it is not going to be installed
E: Sorry, broken packages
```

In this case, *libc6* needs upgraded. As a rule of thumb, we generally avoid to upgrade the C library. If it is our choice to pursue, however, we will need to specifically ask for the *testing* version of *libc6*:

---

```
user@gcux:~$ sudo apt-get install nessus nessusd libc6/testing
Reading Package Lists... Done
Building Dependency Tree... Done
Selected version 2.3.1-16 (Debian:testing) for libc6
[...]
```

The rest of the installation is automatic.

– Using *Aptitude*, it is good to know that

- \* We better first update the list of packages using `u`;
- \* `/` launches a search;
- \* `\` continues the previous search;
- \* `+` installs a package;
- \* When the line is red with a `B` on the left, it means that the dependencies are *Broken*;
- \* `<ENTER>` makes it possible to know why;
- \* The up and down arrows make us browse through the package list;
- \* `<ENTER>` again on the faulty packages;
- \* `+` on the correct version of the package to fix this dependency (equivalent of the “-” in the *package/version* tag of *apt*’s command line);
- \* `-` on a package removes it just like in the main screen (equivalent of the *package-* tag of *apt*’s command line);
- \* `q` to quit the screen with the package;
- \* `/` followed by `~b` jumps to the next broken package;
- \* `g` to let *Aptitude* finish the work of fixing up dependencies, and `g` again to install the proposed solution.

*Aptitude* brings some value in the complicated cases, otherwise `apt-get` is just fine.

- We can download the Debian source file from *Testing* and backport it (compile it for *Stable*). This is definitely the safest way, although it requires more manual work. Note that the source files are authenticated by the process explained in section 5.20. In addition, the files are signed by the Debian maintainer in a `.dsc` file.

We first have to add the correct URI in `/etc/apt/sources.list`:

```
deb-src ftp://ftp.us.debian.org/debian testing main contrib non-free
```

Then, we update the list of packages using `apt-get update`. We can usually get the packages required by the compilation using the `build-dep` option of *Apt*:

---

```
user@gcux:/usr/local/src$ sudo apt-get build-dep nessus
```

However, the gap between the versions of *Nessus* from *Stable* to *Testing* requires to first backport the *Nessus* library package from *Testing*.

We also need some packages from the *Stable* distribution, as the compiling process will remind us. The list of required packages is also present in the `debian/control` file, in the unpacked source package of *Nessus* that you can get using `apt-get source`: *debhelper*, *xlibs-dev*, *libgmp3-dev*, *libz-dev*, *libpcap-dev*, *libgl1.2-dev*, *libgtk1.2-dev*, *libgd-gif1-dev*, *libwrap0-dev*, *libssl-dev* and *libnet1-dev*. We can install these packages using simply

```
sudo apt-get install packages
```

We are not going to compile as *root* in the wild. The `Release` file already provides us with an authenticity check of the package. We will see later on that we have another per source package digital signature at our disposal<sup>29</sup>. This signature is performed by the package maintainer himself. One more step toward a Strong Distribution.

*fakeroot* is a utility that makes it possible to avoid compiling as *root*. We would have to download using *apt* as *root*, change the owner to *user*, make sure that *user* has write access to the current directory, compile as *user* (using *fakeroot* just the way we are using *sudo*) and finally install as *root*. To keep it short, we will compile as *root* knowing that we may prefer the longer way.

```
user@gcux:/usr/local/src$ sudo apt-get source -b nessus-libraries
user@gcux:/usr/local/src$ sudo apt-get source -b nessus-dev
user@gcux:/usr/local/src$ sudo apt-get source -b libnasl2
```

The `-b` flag asks *apt-get* to compile the package after downloading its source. A little bit later, we will see with the *Nessus* package itself how to insert an additional authenticity check between the software download (provided by `apt-get source`) and the compilation.

For now, we can install the freshly compiled *Nessus* libraries:

```
user@gcux:/usr/local/src$ sudo dpkg -i libnessus2_2.0.5-1_sparc.deb
user@gcux:/usr/local/src$ sudo dpkg -i libnessus-dev_2.0.5-1_sparc.deb
user@gcux:/usr/local/src$ sudo dpkg -i libnasl2_2.0.5-2_sparc.deb\
libnasl-dev_2.0.5-2_sparc.deb
```

---

<sup>29</sup>The binary packages do not have such an individual signature yet.

---

Our development environment is ready, we can download the source packages of *Nessus*.

```
user@gcux:/usr/local/src$ sudo apt-get source nessus
user@gcux:/usr/local/src$ sudo apt-get source nessus-plugins
```

We will take the time to check the authenticity of the *Nessus* package source using the per package signing scheme<sup>30</sup>.

We can trust the file `nessus-core_2.0.5-1.dsc`, because

1. Its GnuPG signature is valid:

```
user@gcux:/usr/local/src$ gpg --keyring /usr/share/keyrings/debian-keyring.gpg \
--verify-files nessus-core_2.0.5-1.dsc
gpg: Signature made Sun May 18 07:47:34 2003 CDT
using DSA key ID 200D1596
gpg: Good signature from "Josip Rodin <joy@debian.org>"
gpg:          aka "Josip Rodin <jrodin@jagor.srce.hr>"
gpg:          aka "Josip Rodin <joy@cibalia.gkvk.hr>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs
to the owner.
Primary key fingerprint: 29B6 F1DD 777F 3DB7 C267 F20D 0B54
47A2 200D 1596
```

2. We trust the signing key because it comes from the Debian CD. We could even sign it locally as explained in section 5.21.

The `nessus-core_2.0.5-1.dsc` file contains the MD5 checksums:

Files:

```
c47ba407996a7fb6f68c228c7a55a302 648766 nessus-core_2.0.5.orig.tar.gz
e04404bb765ae4c36e39f7823b34caab 89087 nessus-core_2.0.5-1.diff.gz
```

We trust these MD5 checksums, because the GnuPG signature is valid and the owner of the signature is trusted (as shown in 1.). We compare these checksums with the calculated versions:

```
user@gcux:/usr/local/src$ openssl md5 nessus-core_2.0.5.orig.tar.gz
MD5(nessus-core_2.0.5.orig.tar.gz)= c47ba407996a7fb6f68c228c7a55a302
user@gcux:/usr/local/src$ openssl md5 nessus-core_2.0.5-1.diff.gz
MD5(nessus-core_2.0.5-1.diff.gz)= e04404bb765ae4c36e39f7823b34caab
```

---

<sup>30</sup>In the future version of Debian, the per package signature schema might be extended to binary packages and automated using *apt*.

---

Both the *Release* file and the package signature have validated this source package. We can compile it:

```
user@gcux:/usr/local/src/nessus-core-2.0.5$ sudo debian/rules binary
```

We can then install it:

```
user@gcux:/usr/local/src$ sudo dpkg -i nessus_2.0.5-1_sparc.deb
user@gcux:/usr/local/src$ sudo dpkg -i nessusd_2.0.5-1_sparc.deb
```

As part of the installation of this 2.x version of the *Nessus* server, we are asked some questions about the *Nessus* server SSL certificate. The default answers for the duration of the certificate are fine except otherwise stated by a company policy. The choice needs to be documented to renew the certificate before its expiration date.

We can then compile and install the *nessus-plugins* package using the same set of commands:

- `apt-get source -b` and `dpkg -i` or
- `apt-get source`, check the digital signature, `debian/rules binary` and `dpkg -i`.

## 6 Ongoing Maintenance

### 6.1 Backups

We can use our SCSI tape drive to perform backups, provided that the tape drive is in a safe location. Our kernel, as compiled in section 5.13, supports SCSI tape drives.

As far as user land is concerned, the *mt-st* package does the job:

```
user@gcux:~$ sudo apt-get install mt-st
```

Debian's *crontab* is configured in such a way that it executes all the scripts in `/etc/cron.{daily,weekly,monthly}` with the periodicity indicated by the directory name. The filenames of the scripts should only consist of upper and lower case letters, digits, underscores and hyphens. We place the script `backup` (owned by `root:root`, permissions set to 750) in `/etc/cron.weekly`:

```
#!/bin/sh
LOGFILE=/var/log/backup.log
OPTIONS="-Ou -f /dev/nst0"
echo "Starting backup at `date`" > $LOGFILE
# Clean up the tape
/bin/mt erase
for FILESYSTEM in / /home /var /opt
do
```



---

```
/sbin/dump $OPTIONS $FILESYSTEM >> $LOGFILE 2>&1
echo "Dump of $FILESYSTEM completed at `date`" >> $LOGFILE
done
echo "Dump completed at `date`" >> $LOGFILE
# Rewind and don't eject the tape
/bin/mt rewind
```

We do not backup anything outside `/`, `/home`, `/var` and `/opt`. To keep track of the backup logs, we can add the file `backuplog` in the `/etc/logrotate.d` directory. It would be owned by `root:root` and with permissions set to `644`:

```
/var/log/backup.log
weekly
missingok
rotate 7
compress
notifempty
create 0660 root root
```

The exact periodicity of the backup, the tape handling procedure, the manual intervention and the retention period should be part of your site policy. However, it is clear that the backup of this system is highly sensitive and should happen at least weekly and be securely stored off-site.

## 6.2 Updates

### 6.2.1 Key Updates

The `--recv-keys` option of the `gpg` command makes it possible to retrieve updated versions of the keys, along with revocation certificates if relevant. The web of trust suffers from the same limitation as a Public Key Infrastructure: Revocations are quite poorly handled. The fact that revocations have to diffuse among the various key servers tends to make things even worse (think about a malicious key server that refuses to forward or store a particular revocation).

To limit the risk of using a revoked key, keys should be updated before they are used. We update a key just the same way we import it from a key server, using

```
gpg --recv-keys key_id
```

### 6.2.2 Software Updates

The standard procedure has already been covered in [5.22](#):

1. `user@gcux:~$ sudo apt-get update`

---

2. `user@gcux:~$ apt-release-check`

Make sure that the script does not complain about irrelevancies.

3. `user@gcux:~$ sudo apt-get upgrade`

The best way to know if the version of a package has been recently patched against a vulnerability is to have a look in the Debian *changelog* file<sup>31</sup>:

```
user@gcux:~$ zless /usr/share/doc/package/changelog.Debian.gz
```

The vulnerability is often referenced with either a Debian bug ID, the CERT vulnerability note number or similar. A Debian maintainer will rarely, if ever, upgrade a package to its latest greatest version in case of vulnerability. By policy, she will perform as few changes as required to get rid of the vulnerability. The purpose is to make sure that things do not dramatically change or break unexpectedly as a result of a security fix.

Here is an example of an entry of *changelog.Debian.gz* as a result of a vulnerability fix:

```
user@gcux:~$ zcat /usr/share/doc/openssl/changelog.Debian.gz | head -14
openssl (0.9.6c-2.woody.3) stable-security; urgency=high
```

```
* Non-maintainer upload by the Security Team
* [rsa_eay.c, rsa_lib.c] Apply upstream patch from
  http://www.openssl.org/news/secadv_20030317.txt
  to fix CAN-2003-0147 by enabling RSA blinding to
  prevent a timing attack
* [ssl/s3_srvr.c] Apply upstream patch from
  http://www.openssl.org/news/secadv_20030319.txt
  to fix CAN-2003-0131 (The Klima-Pokorny-Rosa extension
  of Bleichenbacher's attack)
```

```
-- Matt Zimmerman <mdz@debian.org> Mon, 7 Apr 2003 16:51:30 -0400
```

That's for the update in case of vulnerability. On the other hand, to make sure that no pathologically unsafe package is added, we will install some additional packages:

```
gcux:~# apt-get install harden-localflaws harden-remoteflaws \
harden-servers harden-3rdflaws
```

These few packages do not install much (`dpkg -L package` to get the list of files installed by package). Their purpose is to conflict with unsafe packages. For example, once these packages have been installed:

```
gcux:~# apt-get install talkd
Reading Package Lists... Done
Building Dependency Tree... Done
```

---

<sup>31</sup>NMU means Non-Maintainer Upload, typically the package has been patched by the security team.

---

**The following packages will be REMOVED:**

**harden-servers**

The following NEW packages will be installed:

talkd

0 packages upgraded, 1 newly installed, 1 to remove and 0 not upgraded.

Need to get 0B/18.9kB of archives. After unpacking 52.2kB will be used.

Do you want to continue? [Y/n]

At least, the administrator will have been warned... Making sure that these packages are still installed gives a quick audit of the evolution of the server in term of packages installed.

Another good way to keep track of the installed packages is

```
user@gcux:~$ dpkg --get-selections | gzip --best > dpkg-get-selections.gz
user@gcux:~$ gpg -es dpkg-get-selections.gz
```

You need a passphrase to unlock the secret key for

```
user: "Test User (Test User for GCUX practical) <user@gcux>"
1024-bit DSA key, ID 1123481F, created 2003-05-09
```

You did not specify a user ID. (you may use "-r")

Enter the user ID. End with an empty line: Test User

```
Added 2048g/7F1785DC 2003-05-09 "Test User"
(Test User for GCUX practical) <user@gcux>"
```

Enter the user ID. End with an empty line:

These two lines extract, compress and encrypt the list of installed packages. To get the list back after a little while:

```
user@gcux:~$ gpg dpkg-get-selections.gz.gpg
```

You need a passphrase to unlock the secret key for

```
user: "Test User (Test User for GCUX practical) <user@gcux>"
2048-bit ELG-E key, ID 7F1785DC, created 2003-05-09 (main key ID 1123481F)
```

```
gpg: encrypted with 2048-bit ELG-E key, ID 7F1785DC, created 2003-05-09
```

```
"Test User (Test User for GCUX practical) <user@gcux>"
```

```
gpg: Signature made Sat May 17 18:10:31 2003 CDT using DSA key ID 1123481F
```

```
gpg: Good signature from "Test User (Test User for GCUX practical) <user@gcux>"
```

`gzip -dc dpkg-get-selections.gz | dpkg --set-selections` makes it possible to restore the initial list of installed packages. `apt-get upgrade` will perform the actual synchronization of the installed packages.

We can also use the file `dpkg-get-selections` in conjunction with *diff* to quickly know the changes:

```
user@gcux:~$ dpkg --get-selections | diff - dpkg-get-selections
```

---

### 6.2.3 Kernel Updates

Kernel source packages provide newer versions of the Linux kernel. We could typically get those newer sources from the *Testing* distribution. Since the more recent versions of the kernel sources do not exist under *stable*, *apt* will automatically pick them from *Testing* because of our configuration from section 5.17.

```
user@gcux:~$ sudo apt-get install kernel-source-2.4.20
Password:
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  kernel-source-2.4.20
0 packages upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 27.1MB of archives. After unpacking 27.2MB will be used.
Get:1 ftp://ftp.us.debian.org testing/main kernel-source-2.4.20 2.4.20-6 [27.1MB]
```

As *root* (using `sudo su -` if necessary)<sup>32</sup>, we then have to unpack the kernel sources and create the symbolic link as we did in section 5.13. `make mrproper` followed by `make menuconfig` brings us to the *ncurses* interface to recompile the kernel.


We can Load an Alternate Configuration File of the kernel we are running to have a good baseline. If we are running a 2.4.18 kernel, then we will find its configuration file in `/boot/config-2.4.18`. We make the appropriate changes (if any), and then Save Configuration to an Alternate File. We can call this file `config-2.4.20`. We can then `<Exit>`, and `<Yes>`, we wish to save our new kernel configuration.

```
make-kpkg --revision gcux.1 kernel_image
```

will then compile the kernel for us and create a Debian package under `/usr/src` that we can install using `dpkg -i`.

### 6.3 Checking Logs with Logcheck

*Logcheck* is basically an intelligent log parser maintained by both a dedicated maintainer and those maintainers who want *logcheck* to be aware of the specific behavior of their package. The questions we are asked are at the configuration time are:

- security level: `server`.
- The email address to which the mails should be sent: `user`.
- Automatically create `/etc/logcheck/logcheck.logfiles`   
from `/etc/syslog.conf`?: `Yes`.

---

<sup>32</sup>I insist on the “-”: It makes sure that we acquire the same safe environment variables as *root* when she logs in.

---

As advised by the *logcheck-database* package, we create a file called

```
/etc/logcheck/ignore.d/local
```

After a couple of emails full of Firewall logs, we will be able to know what *syslog* lines are really useful versus the one that are just pure noise. Unfortunately, it extensively depends on the kind of devices that are in the lab network. If we notice that 192.168.1.7 floods us with *legitimate* traffic coming from port 1234, we could decide to ignore these log entries and put in the `local` file:

```
kernel: IPTABLES UDP-IN: IN=eth0 .* SRC=192.168.1.7 .* PROTO=UDP SPT=1234 .*
```

This final step heavily depends on the exact network configuration and on the neighboring hosts.

`/usr/share/doc/logcheck/README.Debian` is definitely to be read before making any change to the standard configuration.

## 6.4 Security Audits with *Tiger*

As the description of the package puts it,

Debian's TIGER incorporates new checks primarily oriented toward Debian distribution including: md5sums checks of installed files, location of files not belonging to packages, check of security advisories and analysis of local listening processes.

The installation process asks the following question:

- Who should receive the daily mails?: **user**

The time and frequency of the checks are listed in `/etc/tiger/cronrc`.

The first emails are definitely worth reading and modifying bits and pieces (especially the section with permissions tuning). Installing *Tiger* is a good opportunity to make sure we follow the Filesystem Hierarchy standard. Any report from *Tiger* of a file sitting outside of its normal location should be treated with diligence.

## 7 Check your Configuration

### 7.1 Remaining Services

The distribution server has a limited number of processes that are supposed to be running:

```
gcux:~# netstat -na
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.1:22          0.0.0.0:*               LISTEN
tcp        0      0 172.16.1.1:80           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
raw        0      0 0.0.0.0:1               0.0.0.0:*               7
raw        0      0 0.0.0.0:6               0.0.0.0:*               7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags               Type               State              I-Node Path
unix    1      [ ]                 DGRAM              59                 /dev/log
unix    0      [ ]                 DGRAM              63
```

Any port opened beyond those is to be checked.

## 7.2 Firewall and TCPWrapper Configuration

To get the firewall rules, we can type

```
gcux:~# iptables -L
```

However, this output does not give the interface the rule is applied to. It is quite confusing when one of the first rules is to allow everything on the loopback.

It is a better idea to use

```
gcux:~# iptables-save
```

To have an idea if all the rules have been applied, there should be 81 lines:

```
gcux:~# iptables-save | wc -l
81
```

Reading the output of `iptables-save` is definitely worth it.

`nmap` is the tool of choice to test the validity of the Firewall. Just a `nmap -P0 192.68.1.2` gives an idea of the efficiency of the Firewall.

For each and every port that is firewalled, the Firewall logs the attempt. Here is a sample of the `syslog.log` file:

```
May 22 23:25:55 gcux kernel: IPTABLES TCP-IN:
IN=eth0 OUT= MAC=xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx
SRC=192.168.3.4 DST=192.168.1.2 LEN=60 TOS=0x10 PREC=0x00
TTL=63 ID=24573 DF PROTO=TCP SPT=33362 DPT=23 WINDOW=5840
RES=0x00 SYN URGP=0
```

We can test `nmap` from the different locations (from the management LAN, from the Intranet, from the lab network). In particular, from the management LAN, we try to get a little bit more (`-o` attempts to fingerprint the Operating System):

---

```
user@managementsystem:~$ sudo nmap -ss -p 20,21,22,23,24 -O -P0 192.168.1.2
```

```
Starting nmap 3.20 ( www.insecure.org/nmap/ ) at 2003-05-22 19:12 CDT
```

**Warning: OS detection will be MUCH less reliable**

**because we did not find at least 1 open and 1 closed TCP port**

```
Interesting ports on gcux.mydomain.com (192.168.1.2):
```

Port	State	Service
20/tcp	filtered	ftp-data
21/tcp	filtered	ftp
22/tcp	open	ssh
23/tcp	filtered	telnet
24/tcp	filtered	priv-mail

```
Remote OS guesses: FreeSCO 0.27 (Linux 2.0.38 kernel),
```

```
Linux kernel 2.4.18 (x86), Linux Kernel 2.4.0 - 2.5.20
```

```
Uptime 0.043 days (since Thu May 22 18:11:08 2003)
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 57.076 seconds
```

The information about Linux Kernel 2.4.18 (x86) is rather interesting, only the platform is wrong. The port for *ssh* shows open as expected, any other port shows closed. *nmap* did not find one closed TCP port because all our ports are either opened (*ssh*) or filtered (all the others) from this segment.

- A closed port is typically a port that answers by resetting the connection. The Firewall could simulate a closed port and reset the connection<sup>33</sup>. Unless there is a specific reason to do otherwise, the Firewall usually just silently discards the offending packets. On the client side, here is how a closed port behaves:

```
user@gcux:~$ telnet localhost 21
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
```

- A filtered port, on this other hand, just timeouts after a while:

```
user@gcux:~$ sudo iptables -I INPUT --protocol tcp --dport 12345 -j DROP
user@gcux:~$ telnet localhost 12345
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection timed out
```

The TCPWrapper utility completes our configuration and is especially useful on the internal interface. On the interface facing the Intranet, it is just another layer of security (we cannot even dare to imagine that we will never make a mistake in configuring our Firewall). A good way to know if an executable file is linked with the TCPWrapper library is using *ldd* and looking for *libwrap.so*:

---

<sup>33</sup>Our Firewall does that with the *ident* requests. If the *ident* port was filtered, we would have to wait for the TCP timeout of the *ident* request launched by certain server applications.

---

```
user@gcux:/usr/sbin$ sudo ldd sshd
      libwrap.so.0 => /lib/libwrap.so.0 (0x4001f000)
      libpam.so.0 => /lib/libpam.so.0 (0x40027000)
[...]
```

### 7.3 Verify Network Time Protocol (NTP) functionality

`/var/log/daemon.log` is the file where NTP logs by default. If we entered a Fully Qualified Domain Name that the server cannot resolve, then it will appear in this log file with

```
May 19 19:37:44 gcux ntpd_initres[1993]:
couldn't resolve 'fqdn.yourntpserver.com', giving up on it
```

We can simulate the effect of a tough drift in our internal clock:

```
user@gcux:~$ sudo date
Mon May 19 19:45:32 CDT 2003
user@gcux:~$ sudo date -s "Mon May 19 19:40:32 CDT 2003"
Mon May 19 19:40:32 CDT 2003
```

The file `/var/log/ntpstats/peerstats` shows the effect of this change:

```
52779 2451.870 172.16.1.2 9014 321.794282556 2.219511016 1.938042328
320.511610157
```

After a little while, this drift file shows that we are back on track:

```
52779 2645.960 172.16.1.2 9014 321.794282556 2.219511016 0.190147541
0.030646015
52779 2709.830 172.16.1.2 9014 320.787203956 0.178210648
0.064311529 1.007078600
52779 3095.637 172.16.1.2 8014 0.000000000 0.000000000 0.000000000
4.000000000
52779 3100.607 172.16.1.2 9024 -0.007395171 0.169541479 7.937515259
0.000015259
```

The `daemon.log` file shows the same adjustment:

```
May 19 19:51:35 gcux ntpd[2071]: time set 320.787204 s
May 19 19:51:35 gcux ntpd[2071]: synchronisation lost
```

Curiously, these two lines seem inverted.

If we mess up our Firewall configuration, we first get a

```
May 22 23:26:37 gcux ntpd[363]: sendto(172.16.1.2): Operation not permitted
```

and then a little bit later, once we have fixed the Firewall:

```
May 22 23:27:41 gcux ntpd[363]: Connection re-established to 172.16.1.2
```



## 7.4 Verify Software Download functionality

We need to make sure that the firewall is not too aggressive and still lets our traffic through. http, ftp and PGP key download on the Internet should be allowed. `wget`<sup>34</sup> is the tool of choice to download files in *http* or in *ftp*.

We will see the steps involved if a server being staged in the lab needs *bash* from `sunfreeware.com`. We first download the package of *bash* for Solaris 8:

```
user@gcux:~$ wget http://ftp.wayne.edu/pub/sun_freeware/sparc/8/\
bash-2.05-sol8-sparc-local.gz
--21:05:46--  http://ftp.wayne.edu/pub/sun_freeware
 /sparc/8/bash-2.05-sol8-sparc-local.gz
      => 'bash-2.05-sol8-sparc-local.gz'
Resolving ftp.wayne.edu... done.
Connecting to ftp.wayne.edu[141.217.1.55]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1,656,019 [text/plain]
[...]
```

Additionally, we need to get the list of MD5 checksums from

`http://sunfreeware.secsup.org/md5.html`

and the signature of the list from

`http://sunfreeware.secsup.org/md5.html.asc`

If we try to authenticate the signature straight away:

```
user@gcux:~$ gpg --verify md5.html.asc
gpg: Signature made Tue May  6 23:37:21 2003 CDT using DSA key ID 1D7860F0
gpg: Can't check signature: public key not found
```

We can get the key:

```
user@gcux:~$ gpg --recv-keys 1D7860F0
gpg: key 1D7860F0: public key "Steven Christensen
 (sunfreeware) <steve@smc.vnet.net>" imported
gpg: Total number processed: 1
gpg:          imported: 1
```

However, our web of trust does not reach Steven Christensen:

```
user@gcux:~$ gpg --verify md5.html.asc
gpg: Signature made Tue May  6 23:37:21 2003 CDT using DSA key ID 1D7860F0
gpg: Good signature from "Steven Christensen (sunfreeware) <steve@smc.vnet.net>"
gpg: checking the trustdb
gpg: checking at depth 0 signed=32 ot(-/q/n/m/f/u)=0/0/0/0/0/1
```

<sup>34</sup>We install the `wget-ssl` package to be able to download using also *https*

---

```
gpg: checking at depth 1 signed=252 ot(-/q/n/m/f/u)=32/0/0/0/0/0
gpg: next trustdb check due at 2003-06-20
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4CB A179 2D71 8E67 9FC1 36E2 31A7 EF28 1D78 60F0
```

Unfortunately, the key does not have a wide trust relationship and only has a self signature:

```
user@gcux:~$ gpg --edit-key steve@smc.vnet.net
gpg (GnuPG) 1.2.1; Copyright (C) 2002 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.
```

```
pub 1024D/1D7860F0 created: 2001-07-10 expires: never      trust: -/-
sub 1024g/1FE6052C created: 2001-07-10 expires: never
(1). Steven Christensen (sunfreeware) <steve@smc.vnet.net>
```

```
Command> check
uid Steven Christensen (sunfreeware) <steve@smc.vnet.net>
sig!3      1D7860F0 2001-07-10 [self-signature]
```

We can always compare it with the version stored on `sunfreeware.com` (or what we think is the version stored on `sunfreeware.com`):

```
user@gcux:~$ wget http://www.sunfreeware.com/gpgsig
[...]
user@gcux:~$ gpg --export -a steve@smc.vnet.net | diff - ./gpgsig
2c2,3
< Version: GnuPG v1.2.1 (GNU/Linux)
---
> Version: GnuPG v1.0.0 (SunOS)
> Comment: For info see http://www.gnupg.org
```

The keys seem to be identical. Even for a real life check, all we need to authenticate is really just the fingerprint:

```
rey@linus:~$ gpg --edit-key steve@smc.vnet.net
Command> fpr
pub 1024D/1D7860F0 2001-07-10 Steven Christensen (sunfreeware) <steve@smc.vnet.net>
Primary key fingerprint: E4CB A179 2D71 8E67 9FC1 36E2 31A7 EF28 1D78 60F0
```

At some point, we have to realize that the check will never be of high quality as long as we do not physically meet the person (and check its ID). There remain other lower quality authentications: fax, phone, physical mail... It is an investment that has to be done only once and is definitely worth it.

Now that we trust the list of MD5 checksums, we can wrap up and compare the calculated value of the checksum with its signed value:

```

user@gcux:~$ openssl md5 bash-2.05-sol8-sparc-local.gz
MD5(bash-2.05-sol8-sparc-local.gz)= 630685aa9f59dfbc61743fabb320440b

user@gcux:~$ grep bash-2.05-sol8-sparc-local.gz md5.html
MD5 (bash-2.05-sol8-sparc-local.gz) = 630685aa9f59dfbc61743fabb320440b

```

We trust the MD5 checksum to be correct because it has been signed by a key that we decided to trust. The *bash* package has the correct MD5, we can trust it to be the one distributed by [sunfreeware.com](http://sunfreeware.com).

## 7.5 Verify Software Distribution functionality

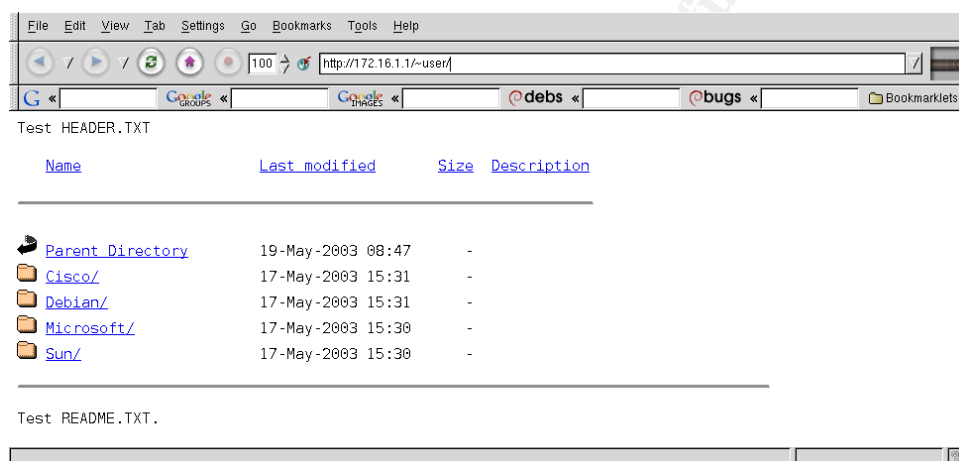


Figure 3: Web Browser

The lab network is supposed to have access to the web server. *Apache* has been configured in section 5.23.7, for the user *user*. If we open a browser on one of lab systems and point the URL to the distribution server, we should be able to get something similar to figure 3.

Some installation procedures (like the installation of Sun Solaris 8) handle directly such an http distribution server. To manually download a file using http, there exists text-only or command line software<sup>35</sup>:

**wget** at <http://www.gnu.org/software/wget/wget.html>

**links** at <http://links.browser.org/>

**lynx** at <http://lynx.browser.org/>

**w3m** at <http://w3m.sourceforge.net/>

<sup>35</sup>All these software are available as Debian packages.

---

***lwp-download*** as part of *Perl*, *libwww-perl* package

at <http://search.cpan.org/author/GAAS/libwww-perl-5.69/>

A good way to ensure the authenticity of these software would be to download the Debian source package from Debian. We would then make it available as a distributed software and compile it on the targeted platform.

To get a source package, we first need to add these line to `/etc/apt/sources.list`:

```
deb-src http://http.us.debian.org/debian testing main contrib non-free
deb-src http://http.us.debian.org/debian stable main contrib non-free
```

We have to update the list of available packages using `apt-get update`. The tar file of *w3m* would then be provided by

```
user@gcux:~$ sudo apt-get source w3m
```

---

## A Kernel Configuration

Note that if you are using *minicom* to get console access, you can pretty easily send any ASCII files using <CTRL>+A S. Choose then *ascii*, and pick the file you want to send. If you had previously opened a *vi* in insert mode, the content of the file is sent to your *vi* buffer.

To get this kernel configuration file from the pdf document, you can copy paste the listing from your favorite pdf viewer to your favorite text editor. Do not forget to strip the line numbers out.

The kernel configuration file used for this installation is:

```
1  #
   # Automatically generated by make menuconfig: don't edit
   #

5  #
   # Code maturity level options
   #
   CONFIG_EXPERIMENTAL=y

10 #
   # Loadable module support
   #
   CONFIG_MODULES=y
   CONFIG_MODVERSIONS=y

15 CONFIG_KMOD=y

   #
   # General setup
   #

20 CONFIG_BBC_I2C=m
   CONFIG_VT=y
   CONFIG_VT_CONSOLE=y
   # CONFIG_SMP is not set
   CONFIG_SPARC64=y

25 # CONFIG_HOTPLUG is not set
   CONFIG_HAVE_DEC_LOCK=y
   # CONFIG_RWSEM_GENERIC_SPINLOCK is not set
   CONFIG_RWSEM_XCHGADD_ALGORITHM=y
   # CONFIG_ISA is not set

30 # CONFIG_ISAPNP is not set
   # CONFIG_EISA is not set
```

---

```
# CONFIG_MCA is not set
# CONFIG_PCMCIA is not set
CONFIG_SBUS=y
35 CONFIG_SBUSCHAR=y
CONFIG_BUSMOUSE=y
CONFIG_SUN_MOUSE=y
CONFIG_SERIAL=y
CONFIG_SUN_SERIAL=y
40 CONFIG_SERIAL_CONSOLE=y
CONFIG_SUN_KEYBOARD=y
CONFIG_SUN_CONSOLE=y
CONFIG_SUN_AUXIO=y
CONFIG_SUN_IO=y
45 CONFIG_PCI=y
CONFIG_RTC=y
CONFIG_PCI_NAMES=y
CONFIG_SUN_OPENPROMFS=m
CONFIG_NET=y
50 CONFIG_SYSVIPC=y
CONFIG_BSD_PROCESS_ACCT=y
CONFIG_SYSCTL=y
CONFIG_KCORE_ELF=y
CONFIG_SPARC32_COMPAT=y
55 CONFIG_BINFMT_ELF32=y
# CONFIG_BINFMT_AOUT32 is not set
CONFIG_BINFMT_ELF=y
CONFIG_BINFMT_MISC=y
# CONFIG_SUNOS_EMUL is not set
60 # CONFIG_SOLARIS_EMUL is not set

#
# Parallel port support
#
65 CONFIG_PARPORT=m
CONFIG_PARPORT_PC=m
CONFIG_PARPORT_PC_CML1=m
# CONFIG_PARPORT_SERIAL is not set
CONFIG_PARPORT_PC_FIFO=y
70 # CONFIG_PARPORT_PC_SUPERIO is not set
# CONFIG_PARPORT_AMIGA is not set
# CONFIG_PARPORT_MFC3 is not set
```

---

```
# CONFIG_PARPORT_ATARI is not set
# CONFIG_PARPORT_GSC is not set
75 # CONFIG_PARPORT_SUNBPP is not set
# CONFIG_PARPORT_OTHER is not set
CONFIG_PARPORT_1284=y
CONFIG_PRINTER=m
CONFIG_ENVCTRL=m
80 # CONFIG_DISPLAY7SEG is not set
# CONFIG_WATCHDOG_CP1XXX is not set
# CONFIG_WATCHDOG_RIO is not set

#
85 # Console drivers
#
CONFIG_PROM_CONSOLE=y

#
90 # Frame-buffer support
#
# CONFIG_FB is not set

#
95 # Misc Linux/SPARC drivers
#
CONFIG_SUN_OPENPROMIO=m
CONFIG_SUN_MOSTEK_RTC=y
CONFIG_SAB82532=y
100 # CONFIG_OBP_FLASH is not set
# CONFIG_SUN_BPP is not set
# CONFIG_SUN_VIDEOPIX is not set
# CONFIG_SUN_AURORA is not set

105 #
# Linux/SPARC audio subsystem (EXPERIMENTAL)
#
# CONFIG_SPARCAUDIO is not set
# CONFIG_SPARCAUDIO_CS4231 is not set
110 # CONFIG_SPARCAUDIO_DUMMY is not set

#
# Memory Technology Devices (MTD)
```

---

```
#
115 # CONFIG_MTD is not set

#
# Block devices
#
120 CONFIG_BLK_DEV_FD=y
    CONFIG_BLK_DEV_LOOP=m
    CONFIG_BLK_DEV_NBD=m

#
125 # Multi-device support (RAID and LVM)
#
# CONFIG_MD is not set
# CONFIG_BLK_DEV_MD is not set
# CONFIG_MD_LINEAR is not set
130 # CONFIG_MD_RAID0 is not set
# CONFIG_MD_RAID1 is not set
# CONFIG_MD_RAID5 is not set
# CONFIG_MD_MULTIPATH is not set
# CONFIG_BLK_DEV_LVM is not set
135 # CONFIG_BLK_DEV_RAM is not set
# CONFIG_BLK_DEV_INITRD is not set

#
# Networking options
140 #
    CONFIG_PACKET=y
    CONFIG_PACKET_MMAP=y
    CONFIG_NETLINK_DEV=y
    CONFIG_NETFILTER=y
145 # CONFIG_NETFILTER_DEBUG is not set
    CONFIG_FILTER=y
    CONFIG_UNIX=y
    CONFIG_INET=y
# CONFIG_IP_MULTICAST is not set
150 # CONFIG_IP_ADVANCED_ROUTER is not set
# CONFIG_IP_PNP is not set
# CONFIG_NET_IPIP is not set
    CONFIG_NET_IPGRE=m
# CONFIG_ARPD is not set
```



---

```
155 # CONFIG_INET_ECN is not set
    CONFIG_SYN_COOKIES=y

    #
    # IP: Netfilter Configuration
160 #
    CONFIG_IP_NF_CONNTRACK=m
    CONFIG_IP_NF_FTP=m
    CONFIG_IP_NF_IRC=m
    CONFIG_IP_NF_QUEUE=m
165 CONFIG_IP_NF_IPTABLES=m
    CONFIG_IP_NF_MATCH_LIMIT=m
    CONFIG_IP_NF_MATCH_MAC=m
    CONFIG_IP_NF_MATCH_MARK=m
    CONFIG_IP_NF_MATCH_MULTIPORT=m
170 CONFIG_IP_NF_MATCH_TOS=m
    CONFIG_IP_NF_MATCH_AH_ESP=m
    CONFIG_IP_NF_MATCH_LENGTH=m
    CONFIG_IP_NF_MATCH_TTL=m
    CONFIG_IP_NF_MATCH_TCPMSS=m
175 CONFIG_IP_NF_MATCH_STATE=m
    CONFIG_IP_NF_MATCH_UNCLEAN=m
    CONFIG_IP_NF_MATCH_OWNER=m
    CONFIG_IP_NF_FILTER=m
    CONFIG_IP_NF_TARGET_REJECT=m
180 CONFIG_IP_NF_TARGET_MIRROR=m
    CONFIG_IP_NF_NAT=m
    CONFIG_IP_NF_NAT_NEEDED=y
    CONFIG_IP_NF_TARGET_MASQUERADE=m
    CONFIG_IP_NF_TARGET_REDIRECT=m
185 CONFIG_IP_NF_NAT_SNMP_BASIC=m
    CONFIG_IP_NF_NAT_IRC=m
    CONFIG_IP_NF_NAT_FTP=m
    CONFIG_IP_NF_MANGLE=m
    CONFIG_IP_NF_TARGET_TOS=m
190 CONFIG_IP_NF_TARGET_MARK=m
    CONFIG_IP_NF_TARGET_LOG=m
    CONFIG_IP_NF_TARGET_ULOG=m
    CONFIG_IP_NF_TARGET_TCPMSS=m
    # CONFIG_IP_NF_COMPAT_IPCHAINS is not set
195 # CONFIG_IP_NF_COMPAT_IPFWADM is not set
```

---

```
# CONFIG_IPV6 is not set
# CONFIG_KHTTPD is not set
# CONFIG_ATM is not set
CONFIG_VLAN_8021Q=m
200 # CONFIG_IPX is not set
# CONFIG_ATALK is not set
# CONFIG_DECNET is not set
# CONFIG_BRIDGE is not set
# CONFIG_X25 is not set
205 # CONFIG_LAPB is not set
# CONFIG_LLC is not set
# CONFIG_NET_DIVERT is not set
# CONFIG_ECONET is not set
# CONFIG_WAN_ROUTER is not set
210 # CONFIG_NET_FASTROUTE is not set
# CONFIG_NET_HW_FLOWCONTROL is not set

#
# QoS and/or fair queueing
215 #
CONFIG_NET_SCHED=y
CONFIG_NET_SCH_CBQ=m
CONFIG_NET_SCH_CSZ=m
CONFIG_NET_SCH_PRIO=m
220 CONFIG_NET_SCH_RED=m
CONFIG_NET_SCH_SFQ=m
CONFIG_NET_SCH_TEQL=m
CONFIG_NET_SCH_TBF=m
CONFIG_NET_SCH_GRED=m
225 CONFIG_NET_SCH_DSMARK=m
# CONFIG_NET_SCH_INGRESS is not set
CONFIG_NET_QOS=y
CONFIG_NET_ESTIMATOR=y
CONFIG_NET_CLS=y
230 CONFIG_NET_CLS_TCINDEX=m
CONFIG_NET_CLS_ROUTE4=m
CONFIG_NET_CLS_ROUTE=y
CONFIG_NET_CLS_FW=m
CONFIG_NET_CLS_U32=m
235 CONFIG_NET_CLS_RSVP=m
CONFIG_NET_CLS_RSVP6=m
```

---

```
CONFIG_NET_CLS_POLICE=y

#
240 # ATA/IDE/MFM/RLL support
#
CONFIG_IDE=y

#
245 # IDE, ATA and ATAPI Block devices
#
CONFIG_BLK_DEV_IDE=y
# CONFIG_BLK_DEV_HD_IDE is not set
# CONFIG_BLK_DEV_HD is not set
250 CONFIG_BLK_DEV_IDEDISK=y
# CONFIG_IDEDISK_MULTI_MODE is not set
# CONFIG_BLK_DEV_IDEDISK_VENDOR is not set
# CONFIG_BLK_DEV_IDEDISK_FUJITSU is not set
# CONFIG_BLK_DEV_IDEDISK_IBM is not set
255 # CONFIG_BLK_DEV_IDEDISK_MAXTOR is not set
# CONFIG_BLK_DEV_IDEDISK_QUANTUM is not set
# CONFIG_BLK_DEV_IDEDISK_SEAGATE is not set
# CONFIG_BLK_DEV_IDEDISK_WD is not set
# CONFIG_BLK_DEV_COMMERIAL is not set
260 # CONFIG_BLK_DEV_TIVO is not set
# CONFIG_BLK_DEV_IDECS is not set
CONFIG_BLK_DEV_IDECD=y
CONFIG_BLK_DEV_IDETAPE=m
CONFIG_BLK_DEV_IDEFLOPPY=m
265 # CONFIG_BLK_DEV_IDESCSI is not set
# CONFIG_BLK_DEV_CMD640 is not set
# CONFIG_BLK_DEV_CMD640_ENHANCED is not set
# CONFIG_BLK_DEV_ISAPNP is not set
# CONFIG_BLK_DEV_RZ1000 is not set
270 CONFIG_BLK_DEV_IDEPCI=y
# CONFIG_IDEPCI_SHARE_IRQ is not set
CONFIG_BLK_DEV_IDEDMA_PCI=y
CONFIG_BLK_DEV_ADMA=y
# CONFIG_BLK_DEV_OFFBOARD is not set
275 CONFIG_IDEDMA_PCI_AUTO=y
CONFIG_BLK_DEV_IDEDMA=y
# CONFIG_IDEDMA_PCI_WIP is not set
```

---

```
# CONFIG_IDEDMA_NEW_DRIVE_LISTINGS is not set
# CONFIG_BLK_DEV_AEC62XX is not set
280 # CONFIG_AEC62XX_TUNING is not set
# CONFIG_BLK_DEV_ALI15X3 is not set
# CONFIG_WDC_ALI15X3 is not set
# CONFIG_BLK_DEV_AMD74XX is not set
# CONFIG_AMD74XX_OVERRIDE is not set
285 CONFIG_BLK_DEV_CMD64X=y
# CONFIG_BLK_DEV_CY82C693 is not set
# CONFIG_BLK_DEV_CS5530 is not set
# CONFIG_BLK_DEV_HPT34X is not set
# CONFIG_HPT34X_AUTODMA is not set
290 # CONFIG_BLK_DEV_HPT366 is not set
CONFIG_BLK_DEV_NS87415=y
# CONFIG_BLK_DEV_OPTI621 is not set
# CONFIG_BLK_DEV_PDC202XX is not set
# CONFIG_PDC202XX_BURST is not set
295 # CONFIG_PDC202XX_FORCE is not set
# CONFIG_BLK_DEV_SVWKS is not set
# CONFIG_BLK_DEV_SIS5513 is not set
# CONFIG_BLK_DEV_SLC90E66 is not set
# CONFIG_BLK_DEV_TRM290 is not set
300 # CONFIG_BLK_DEV_VIA82CXXX is not set
# CONFIG_IDE_CHIPSETS is not set
CONFIG_IDEDMA_AUTO=y
# CONFIG_IDEDMA_IVB is not set
# CONFIG_DMA_NONPCI is not set
305 CONFIG_BLK_DEV_IDE_MODES=y
# CONFIG_BLK_DEV_ATA RAID is not set
# CONFIG_BLK_DEV_ATA RAID_PDC is not set
# CONFIG_BLK_DEV_ATA RAID_HPT is not set

310 #
# SCSI support
#
CONFIG_SCSI=y
CONFIG_BLK_DEV_SD=y
315 CONFIG_SD_EXTRA_DEVS=40
CONFIG_CHR_DEV_ST=y
CONFIG_CHR_DEV_OSST=m
CONFIG_BLK_DEV_SR=y
```

---

```

CONFIG_BLK_DEV_SR_VENDOR=y
320 CONFIG_SR_EXTRA_DEVS=2
CONFIG_CHR_DEV_SG=m
CONFIG SCSI_MULTI_LUN=y
CONFIG SCSI_CONSTANTS=y
# CONFIG SCSI_LOGGING is not set
325
#
# SCSI low-level drivers
#
# CONFIG SCSI_SUNESP is not set
330 # CONFIG SCSI_QLOGICPTI is not set
CONFIG SCSI_AIC7XXX=m
CONFIG_AIC7XXX_CMDS_PER_DEVICE=253
CONFIG_AIC7XXX_RESET_DELAY_MS=5000
# CONFIG_AIC7XXX_BUILD_FIRMWARE is not set
335 # CONFIG SCSI_AIC7XXX_OLD is not set
CONFIG SCSI_SYM53C8XX_2=y
CONFIG SCSI_SYM53C8XX_DMA_ADDRESSING_MODE=1
CONFIG SCSI_SYM53C8XX_DEFAULT_TAGS=16
CONFIG SCSI_SYM53C8XX_MAX_TAGS=64
340 # CONFIG SCSI_SYM53C8XX_IOMAPPED is not set
CONFIG SCSI_QLOGIC_ISP=m
CONFIG SCSI_QLOGIC_FC=m
CONFIG SCSI_QLOGIC_FC_FIRMWARE=y

345 #
# Fibre Channel support
#
# CONFIG_FC4 is not set
# CONFIG_FC4_SOC is not set
350 # CONFIG_FC4_SOCAL is not set
# CONFIG SCSI_PLUTO is not set
# CONFIG SCSI_FCAL is not set

#
355 # Fusion MPT device support
#
# CONFIG_FUSION is not set
# CONFIG_FUSION_BOOT is not set
# CONFIG_FUSION_ISENSE is not set

```

---

```
360 # CONFIG_FUSION_CTL is not set
    # CONFIG_FUSION_LAN is not set

    #
    # IEEE 1394 (FireWire) support (EXPERIMENTAL)
365 #
    # CONFIG_IEEE1394 is not set

    #
    # Network device support
370 #
    CONFIG_NETDEVICES=y

    #
    # ARCnet devices
375 #
    # CONFIG_ARCNET is not set
    CONFIG_DUMMY=m
    # CONFIG_BONDING is not set
    # CONFIG_EQUALIZER is not set
380 # CONFIG_TUN is not set
    # CONFIG_ETHERTAP is not set

    #
    # Ethernet (10 or 100Mbit)
385 #
    CONFIG_NET_ETHERNET=y
    CONFIG_SUNLANCE=m
    CONFIG_HAPPYMEAL=m
    # CONFIG_SUNBMAC is not set
390 CONFIG_SUNQE=m
    CONFIG_SUNGEM=m
    CONFIG_NET_VENDOR_3COM=y
    # CONFIG_EL1 is not set
    # CONFIG_EL2 is not set
395 # CONFIG_ELPLUS is not set
    # CONFIG_EL16 is not set
    # CONFIG_ELMC is not set
    # CONFIG_ELMC_II is not set
    CONFIG_VORTEX=m
400 # CONFIG_LANCE is not set
```

---

```
CONFIG_NET_VENDOR_SMC=y
# CONFIG_WD80x3 is not set
# CONFIG_ULTRAMCA is not set
# CONFIG_ULTRA is not set
405 # CONFIG_ULTRA32 is not set
# CONFIG_SMC9194 is not set
CONFIG_NET_VENDOR_RACAL=y
# CONFIG_NI5010 is not set
# CONFIG_NI52 is not set
410 # CONFIG_NI65 is not set
# CONFIG_HP100 is not set
# CONFIG_NET_ISA is not set
CONFIG_NET_PCI=y
CONFIG_PCNET32=m
415 CONFIG_ADAPTEC_STARFIRE=m
# CONFIG_APRICOT is not set
# CONFIG_CS89x0 is not set
CONFIG_TULIP=m
# CONFIG_TULIP_MWI is not set
420 CONFIG_TULIP_MMIO=y
CONFIG_DE4X5=m
CONFIG_DGRS=m
CONFIG_DM9102=m
CONFIG_EEPRO100=m
425 # CONFIG_LNE390 is not set
CONFIG_FEALNX=m
CONFIG_NATSEMI=m
# CONFIG_NATSEMI_CABLE_MAGIC is not set
CONFIG_NE2K_PCI=m
430 # CONFIG_NE3210 is not set
# CONFIG_ES3210 is not set
# CONFIG_8139CP is not set
CONFIG_8139TOO=m
CONFIG_8139TOO_PIO=y
435 # CONFIG_8139TOO_TUNE_TWISTER is not set
CONFIG_8139TOO_8129=y
# CONFIG_8139_NEW_RX_RESET is not set
CONFIG_SIS900=m
CONFIG_EPIC100=m
440 CONFIG_SUNDANCE=m
# CONFIG_TLAN is not set
```

---

```
CONFIG_VIA_RHINE=m
# CONFIG_VIA_RHINE_MMIO is not set
CONFIG_WINBOND_840=m
445 # CONFIG_NET_POCKET is not set

#
# Ethernet (1000 Mbit)
#
450 # CONFIG_ACENIC is not set
CONFIG_DL2K=m
# CONFIG_MYRI_SBUS is not set
# CONFIG_NS83820 is not set
# CONFIG_HAMACHI is not set
455 # CONFIG_YELLOWFIN is not set
# CONFIG_SK98LIN is not set
# CONFIG_FDDI is not set
# CONFIG_HIPPI is not set
# CONFIG_PLIP is not set
460 # CONFIG_PPP is not set
# CONFIG_SLIP is not set

#
# Wireless LAN (non-hamradio)
465 #
# CONFIG_NET_RADIO is not set

#
# Token Ring devices
470 #
# CONFIG_TR is not set
# CONFIG_NET_FC is not set
# CONFIG_RCPCI is not set
# CONFIG_SHAPER is not set
475 #
# Wan interfaces
#
# CONFIG_WAN is not set
480 #
# Unix 98 PTY support
```



---

```
#
CONFIG_UNIX98_PTYS=y
485 CONFIG_UNIX98_PTY_COUNT=256

#
# Video For Linux
#
490 # CONFIG_VIDEO_DEV is not set

#
# XFree86 DRI support
#
495 # CONFIG_DRM is not set
# CONFIG_DRM_FFB is not set
# CONFIG_DRM_TDFX is not set
# CONFIG_DRM_R128 is not set

500 #
# Input core support
#
CONFIG_INPUT=y
CONFIG_INPUT_KEYBDEV=y
505 CONFIG_INPUT_MOUSEDEV=y
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
# CONFIG_INPUT_JOYDEV is not set
CONFIG_INPUT_EVDEV=y
510 #
# File systems
#
# CONFIG_QUOTA is not set
515 # CONFIG_AUTOFS_FS is not set
# CONFIG_AUTOFS4_FS is not set
# CONFIG_REISERFS_FS is not set
# CONFIG_REISERFS_CHECK is not set
# CONFIG_REISERFS_PROC_INFO is not set
520 # CONFIG_ADFS_FS is not set
# CONFIG_ADFS_FS_RW is not set
# CONFIG_AFFS_FS is not set
# CONFIG_HFS_FS is not set
```

---

```
# CONFIG_BFS_FS is not set
525 CONFIG_EXT3_FS=y
CONFIG_JBD=y
# CONFIG_JBD_DEBUG is not set
CONFIG_FAT_FS=m
CONFIG_MSDOS_FS=m
530 # CONFIG_UMSDOS_FS is not set
CONFIG_VFAT_FS=m
# CONFIG_EFS_FS is not set
# CONFIG_JFFS_FS is not set
# CONFIG_JFFS2_FS is not set
535 # CONFIG_CRAMFS is not set
# CONFIG_TMPFS is not set
CONFIG_RAMFS=y
CONFIG_ISO9660_FS=m
CONFIG_JOLIET=y
540 CONFIG_ZISOFS=y
# CONFIG_MINIX_FS is not set
# CONFIG_VXFS_FS is not set
# CONFIG_NTFS_FS is not set
# CONFIG_NTFS_RW is not set
545 # CONFIG_HPFS_FS is not set
CONFIG_PROC_FS=y
# CONFIG_DEVFS_FS is not set
# CONFIG_DEVFS_MOUNT is not set
# CONFIG_DEVFS_DEBUG is not set
550 CONFIG_DEVPTS_FS=y
# CONFIG_QNX4FS_FS is not set
# CONFIG_QNX4FS_RW is not set
# CONFIG_ROMFS_FS is not set
CONFIG_EXT2_FS=y
555 # CONFIG_SYSV_FS is not set
CONFIG_UDF_FS=m
# CONFIG_UDF_RW is not set
CONFIG_UFS_FS=m
# CONFIG_UFS_FS_WRITE is not set
560
#
# Network File Systems
#
# CONFIG_CODA_FS is not set
```

---

```
565 # CONFIG_INTERMEZZO_FS is not set
    # CONFIG_NFS_FS is not set
    # CONFIG_NFS_V3 is not set
    # CONFIG_ROOT_NFS is not set
    # CONFIG_NFSD is not set
570 # CONFIG_NFSD_V3 is not set
    # CONFIG_SUNRPC is not set
    # CONFIG_LOCKD is not set
    # CONFIG_SMB_FS is not set
    # CONFIG_NCP_FS is not set
575 # CONFIG_NCPFS_PACKET_SIGNING is not set
    # CONFIG_NCPFS_IOCTL_LOCKING is not set
    # CONFIG_NCPFS_STRONG is not set
    # CONFIG_NCPFS_NFS_NS is not set
    # CONFIG_NCPFS_OS2_NS is not set
580 # CONFIG_NCPFS_SMALLDOS is not set
    # CONFIG_NCPFS_NLS is not set
    # CONFIG_NCPFS_EXTRAS is not set
    CONFIG_ZISOFS_FS=m
    CONFIG_ZLIB_FS_INFLATE=m
585
    #
    # Partition Types
    #
    CONFIG_PARTITION_ADVANCED=y
590 # CONFIG_ACORN_PARTITION is not set
    # CONFIG_OSF_PARTITION is not set
    # CONFIG_AMIGA_PARTITION is not set
    # CONFIG_ATARI_PARTITION is not set
    # CONFIG_MAC_PARTITION is not set
595 # CONFIG_MSDOS_PARTITION is not set
    # CONFIG_LDM_PARTITION is not set
    # CONFIG_SGI_PARTITION is not set
    # CONFIG_ULTRIX_PARTITION is not set
    CONFIG_SUN_PARTITION=y
600 # CONFIG_SMB_NLS is not set
    CONFIG_NLS=y

    #
    # Native Language Support
605 #
```

---

```
CONFIG-NLS_DEFAULT="iso8859-1"
CONFIG-NLS_CODEPAGE_437=m
CONFIG-NLS_CODEPAGE_737=m
CONFIG-NLS_CODEPAGE_775=m
610 CONFIG-NLS_CODEPAGE_850=m
CONFIG-NLS_CODEPAGE_852=m
CONFIG-NLS_CODEPAGE_855=m
CONFIG-NLS_CODEPAGE_857=m
CONFIG-NLS_CODEPAGE_860=m
615 CONFIG-NLS_CODEPAGE_861=m
CONFIG-NLS_CODEPAGE_862=m
CONFIG-NLS_CODEPAGE_863=m
CONFIG-NLS_CODEPAGE_864=m
CONFIG-NLS_CODEPAGE_865=m
620 CONFIG-NLS_CODEPAGE_866=m
CONFIG-NLS_CODEPAGE_869=m
CONFIG-NLS_CODEPAGE_936=m
CONFIG-NLS_CODEPAGE_950=m
CONFIG-NLS_CODEPAGE_932=m
625 CONFIG-NLS_CODEPAGE_949=m
CONFIG-NLS_CODEPAGE_874=m
CONFIG-NLS_ISO8859_8=m
CONFIG-NLS_CODEPAGE_1250=m
CONFIG-NLS_CODEPAGE_1251=m
630 CONFIG-NLS_ISO8859_1=m
CONFIG-NLS_ISO8859_2=m
CONFIG-NLS_ISO8859_3=m
CONFIG-NLS_ISO8859_4=m
CONFIG-NLS_ISO8859_5=m
635 CONFIG-NLS_ISO8859_6=m
CONFIG-NLS_ISO8859_7=m
CONFIG-NLS_ISO8859_9=m
CONFIG-NLS_ISO8859_13=m
CONFIG-NLS_ISO8859_14=m
640 CONFIG-NLS_ISO8859_15=m
CONFIG-NLS_KOI8_R=m
CONFIG-NLS_KOI8_U=m
CONFIG-NLS_UTF8=m

645 #
# Sound
```

---

```
#
# CONFIG_SOUND is not set

650 #
# USB support
#
# CONFIG_USB is not set
# CONFIG_USB_UHCI is not set
655 # CONFIG_USB_UHCI_ALT is not set
# CONFIG_USB_OHCI is not set
# CONFIG_USB_AUDIO is not set
# CONFIG_USB_BLUETOOTH is not set
# CONFIG_USB_STORAGE is not set
660 # CONFIG_USB_STORAGE_DEBUG is not set
# CONFIG_USB_STORAGE_DATAFAB is not set
# CONFIG_USB_STORAGE_FREECOM is not set
# CONFIG_USB_STORAGE_ISD200 is not set
# CONFIG_USB_STORAGE_DPCM is not set
665 # CONFIG_USB_STORAGE_HP8200e is not set
# CONFIG_USB_STORAGE_SDDR09 is not set
# CONFIG_USB_STORAGE_JUMPSHOT is not set
# CONFIG_USB_ACM is not set
# CONFIG_USB_PRINTER is not set
670 # CONFIG_USB_HID is not set
# CONFIG_USB_HIDDEV is not set
# CONFIG_USB_KBD is not set
# CONFIG_USB_MOUSE is not set
# CONFIG_USB_WACOM is not set
675 # CONFIG_USB_DC2XX is not set
# CONFIG_USB_MDC800 is not set
# CONFIG_USB_SCANNER is not set
# CONFIG_USB_MICROTEK is not set
# CONFIG_USB_HPUSBSCSI is not set
680 # CONFIG_USB_PEGASUS is not set
# CONFIG_USB_KAWETH is not set
# CONFIG_USB_CATC is not set
# CONFIG_USB_CDCETHER is not set
# CONFIG_USB_USBNET is not set
685 # CONFIG_USB_USS720 is not set

#
```

---

```
# USB Serial Converter support
#
690 # CONFIG_USB_SERIAL is not set
# CONFIG_USB_SERIAL_GENERIC is not set
# CONFIG_USB_SERIAL_BELKIN is not set
# CONFIG_USB_SERIAL_WHITEHEAT is not set
# CONFIG_USB_SERIAL_DIGI_ACCELEPORT is not set
695 # CONFIG_USB_SERIAL_EMPEG is not set
# CONFIG_USB_SERIAL_FTDI_SIO is not set
# CONFIG_USB_SERIAL_VISOR is not set
# CONFIG_USB_SERIAL_IPAQ is not set
# CONFIG_USB_SERIAL_IR is not set
700 # CONFIG_USB_SERIAL_EDGEPORT is not set
# CONFIG_USB_SERIAL_KEYSPAN_PDA is not set
# CONFIG_USB_SERIAL_KEYSPAN is not set
# CONFIG_USB_SERIAL_KEYSPAN_USA28 is not set
# CONFIG_USB_SERIAL_KEYSPAN_USA28X is not set
705 # CONFIG_USB_SERIAL_KEYSPAN_USA28XA is not set
# CONFIG_USB_SERIAL_KEYSPAN_USA28XB is not set
# CONFIG_USB_SERIAL_KEYSPAN_USA19 is not set
# CONFIG_USB_SERIAL_KEYSPAN_USA18X is not set
# CONFIG_USB_SERIAL_KEYSPAN_USA19W is not set
710 # CONFIG_USB_SERIAL_KEYSPAN_USA49W is not set
# CONFIG_USB_SERIAL_MCT_U232 is not set
# CONFIG_USB_SERIAL_KLSI is not set
# CONFIG_USB_SERIAL_PL2303 is not set
# CONFIG_USB_SERIAL_CYBERJACK is not set
715 # CONFIG_USB_SERIAL_XIRCOM is not set
# CONFIG_USB_SERIAL_OMNINET is not set
# CONFIG_USB_RIO500 is not set

#
720 # Bluetooth support
#
# CONFIG_BLUEZ is not set

#
725 # Watchdog
#
# CONFIG_SOFT_WATCHDOG is not set
```

```
#
730 # Kernel hacking
#
# CONFIG_DEBUG_KERNEL is not set
```

## B Firewall Configuration

Copy and save this file as `fw.sh`. The variables (in the Definitions section) need to be updated to fit your networking environment. This script is a free adaptation of [\[15\]](#).

```
#!/bin/sh
#
# Load appropriate modules.
modprobe ip_tables
modprobe ip_conntrack
modprobe ip_conntrack_ftp

#####
# Definitions
IPTABLES=/sbin/iptables
IFACE="eth0"
IFACE2="eth1"
IPADDR="192.168.1.2"
NAMESERVER_1="x.x.x.x"
NAMESERVER_2="x.x.x.x"
MGT_LAN="192.168.2.0/24"
BROADCAST="192.168.1.255"
LOOPBACK="127.0.0.0/8"
P_PORTS="0:1023"
UP_PORTS="1024:65535"
TR_SRC_PORTS="32769:65535"
TR_DEST_PORTS="33434:33523"
CLASS_D_MULTICAST="224.0.0.0/4"
CLASS_E_RESERVED_NET="240.0.0.0/5"

#####
# DROP policy for the built-in chains.
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP
# Flush all previous rules
$IPTABLES -F
$IPTABLES -X
$IPTABLES -Z
#####
# Behaviour of the TCP/IP Stack
#
```

---

```

# CONFIG_SYSCTL enabled in the kernel
# Disable response to broadcasts (avoid being a Smurf amplifier)
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
# Refuse source routing
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
# Refuse ICMP redirects
for interface in /proc/sys/net/ipv4/conf/*/accept_redirects; do
/bin/echo "0" > $interface
done
# Enable bad error message protection.
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
# Turn on reverse path filtering.
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
/bin/echo "1" > $interface
done
# Log spoofed packets, source routed packets, redirect packets.
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
# No IP forwarding
/bin/echo "0" > /proc/sys/net/ipv4/ip_forward
#####
# Firewall Rules
#
# No restriction on Loopback
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT
# Internal interface wide open
$IPTABLES -A INPUT -i $IFACE2 -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE2 -j ACCEPT
# SYN Flooding protection
# (only ssh is running, so we can live without it)
# Would work on Sparc32, but not Sparc64
# To best tested on the still to be released Linux 2.4.21
# $IPTABLES -N syn-flood
# $IPTABLES -A INPUT -i $IFACE -p tcp --syn -j syn-flood
# $IPTABLES -A syn-flood -m limit --limit 1/s \
#   --limit-burst 4 -j RETURN
# $IPTABLES -A syn-flood -j DROP
# NEW tcp connections should be SYN packets
$IPTABLES -A INPUT -i $IFACE -p tcp ! --syn -m state \
  --state NEW -j DROP
# anti-SPOOFING
# Another layer of security, besides the one provided above by the IP stack
# Refuse spoofed packets pretending to be from your IP address.
$IPTABLES -A INPUT -i $IFACE -s $IPADDR -j DROP
# Refuse Class D multicast addresses.
# Multicast is illegal as a source address.
$IPTABLES -A INPUT -i $IFACE -s $CLASS_D_MULTICAST -j DROP
# Refuse Class E reserved IP addresses.
$IPTABLES -A INPUT -i $IFACE -s $CLASS_E_RESERVED_NET -j DROP

```



---

```

# Refuse packets claiming to be to the loopback interface.
# Refusing packets claiming to be to the loopback interface
# protects against source quench, whereby a machine can be
# told to slow itself down by an icmp quench to the loopback.
$IPTABLES -A INPUT -i $IFACE -d $LOOPBACK -j DROP
# Refuse broadcast address packets.
$IPTABLES -A INPUT -i $IFACE -d $BROADCAST -j DROP
# ICMP
$IPTABLES -N icmp-in
$IPTABLES -N icmp-out
$IPTABLES -A INPUT -i $IFACE -p icmp -j icmp-in
$IPTABLES -A OUTPUT -o $IFACE -p icmp -j icmp-out
# Accept 0,3,4,11,12,14,16,18 in.
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 0 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 3 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 4 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 11 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 12 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 14 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 16 \
-s 0/0 -d $IPADDR -j RETURN
$IPTABLES -A icmp-in -i $IFACE -p icmp --icmp-type 18 \
-s 0/0 -d $IPADDR -j RETURN
# Allow 4,8,12,13,15,17 out.
$IPTABLES -A icmp-out -o $IFACE -p icmp --icmp-type 4 \
-s $IPADDR -d 0/0 -j RETURN
$IPTABLES -A icmp-out -o $IFACE -p icmp --icmp-type 8 \
-s $IPADDR -d 0/0 -j RETURN
$IPTABLES -A icmp-out -o $IFACE -p icmp --icmp-type 12 \
-s $IPADDR -d 0/0 -j RETURN
$IPTABLES -A icmp-out -o $IFACE -p icmp --icmp-type 13 \
-s $IPADDR -d 0/0 -j RETURN
$IPTABLES -A icmp-out -o $IFACE -p icmp --icmp-type 15 \
-s $IPADDR -d 0/0 -j RETURN
$IPTABLES -A icmp-out -o $IFACE -p icmp --icmp-type 17 \
-s $IPADDR -d 0/0 -j RETURN
# Any ICMP not already allowed is logged and then dropped.
$IPTABLES -A icmp-in -i $IFACE -j LOG \
--log-prefix "IPTABLES ICMP-BAD-TYPE-IN: "
$IPTABLES -A icmp-in -i $IFACE -j DROP
$IPTABLES -A icmp-out -o $IFACE -j LOG \
--log-prefix "IPTABLES ICMP-BAD-TYPE-OUT: "
$IPTABLES -A icmp-out -o $IFACE -j DROP

```

---

```

# Certains inbound (resp. outbound) ICMP
# has return to the INPUT (resp. OUTPUT) chain,
# the others have been dropped and logged
$IPTABLES -A INPUT -i $IFACE -p icmp -m state \
--state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p icmp -m state \
--state NEW,ESTABLISHED,RELATED -j ACCEPT
# DNS outbound
# Both UDP and TCP (used for large transfers / when UDP fails)
# Allow in for DNS client from nameservers.
$IPTABLES -A INPUT -i $IFACE -p udp -s $NAMESERVER_1 \
--sport 53 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -i $IFACE -p udp -s $NAMESERVER_2 \
--sport 53 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -i $IFACE -p tcp -s $NAMESERVER_1 \
--sport 53 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -i $IFACE -p tcp -s $NAMESERVER_2 \
--sport 53 -m state --state ESTABLISHED -j ACCEPT
# Allow UDP packets to DNS servers from client.
$IPTABLES -A OUTPUT -o $IFACE -p udp -d $NAMESERVER_1 \
--dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p udp -d $NAMESERVER_2 \
--dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp -d $NAMESERVER_1 \
--dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp -d $NAMESERVER_2 \
--dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
# SSH inbound
$IPTABLES -A INPUT -i $IFACE -p tcp --dport 22 -m state \
-s $MGT_LAN --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --sport 22 -m state \
-d $MGT_LAN --state ESTABLISHED -j ACCEPT
# pgp outbound
$IPTABLES -A INPUT -i $IFACE -p tcp --sport 11371 \
-m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --dport 11371 \
-m state --state NEW,ESTABLISHED -j ACCEPT
# http outbound
$IPTABLES -A INPUT -i $IFACE -p tcp --sport 80 \
-m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --dport 80 \
-m state --state NEW,ESTABLISHED -j ACCEPT
# https outbound
$IPTABLES -A INPUT -i $IFACE -p tcp --sport 443 \
-m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --dport 443 \
-m state --state NEW,ESTABLISHED -j ACCEPT
# FTP outbound
$IPTABLES -A INPUT -i $IFACE -p tcp --sport 21 \

```

---

```

-m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --dport 21 \
-m state --state NEW,ESTABLISHED -j ACCEPT
# Active ftp.
$IPTABLES -A INPUT -i $IFACE -p tcp --sport 20 \
-m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --dport 20 \
-m state --state ESTABLISHED -j ACCEPT
# Passive ftp.
$IPTABLES -A INPUT -i $IFACE -p tcp --sport $UP_PORTS \
--dport $UP_PORTS -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $IFACE -p tcp --sport $UP_PORTS \
--dport $UP_PORTS -m state --state ESTABLISHED,RELATED -j ACCEPT
# Reject ident probes with a tcp reset
$IPTABLES -A INPUT -i $IFACE -p tcp --dport 113 \
-j REJECT --reject-with tcp-reset
# Outgoing TRACEROUTE
$IPTABLES -A OUTPUT -o $IFACE -p udp --sport $TR_SRC_PORTS \
--dport $TR_DEST_PORTS -m state --state NEW -j ACCEPT
#####
# LOGGING
# Cannot implement rate limiting on logging (again bug)
# We would add -m limit --limit 6/h --limit-burst 5
# UDP
$IPTABLES -A INPUT -i $IFACE -p udp -j LOG \
--log-prefix "IPTABLES UDP-IN: "
$IPTABLES -A INPUT -i $IFACE -p udp -j DROP
$IPTABLES -A OUTPUT -o $IFACE -p udp -j LOG \
--log-prefix "IPTABLES UDP-OUT: "
$IPTABLES -A OUTPUT -o $IFACE -p udp -j DROP
# ICMP
$IPTABLES -A INPUT -i $IFACE -p icmp -j LOG \
--log-prefix "IPTABLES ICMP-IN: "
$IPTABLES -A INPUT -i $IFACE -p icmp -j DROP
$IPTABLES -A OUTPUT -o $IFACE -p icmp -j LOG \
--log-prefix "IPTABLES ICMP-OUT: "
$IPTABLES -A OUTPUT -o $IFACE -p icmp -j DROP
# TCP
$IPTABLES -A INPUT -i $IFACE -p tcp -j LOG \
--log-prefix "IPTABLES TCP-IN: "
$IPTABLES -A INPUT -i $IFACE -p tcp -j DROP
$IPTABLES -A OUTPUT -o $IFACE -p tcp -j LOG \
--log-prefix "IPTABLES TCP-OUT: "
$IPTABLES -A OUTPUT -o $IFACE -p tcp -j DROP
# Rest
$IPTABLES -A INPUT -i $IFACE -j LOG \
--log-prefix "IPTABLES PROTOCOL-X-IN: "
$IPTABLES -A INPUT -i $IFACE -j DROP
$IPTABLES -A OUTPUT -o $IFACE -j LOG \

```

---

```
--log-prefix "IPTABLES PROTOCOL-X-OUT: "
$IPTABLES -A OUTPUT -o $IFACE -j DROP
# END
```

## C Linux Kernel

The object of this section is to detail the steps to authenticate a Linux kernel, from a Debian system. It can be used to install a trusted Linux kernel on non-Debian or on non-Sparc platforms.

1. We get the software and its signature:

```
user@gcux:~$ wget \
ftp://ftp.us.kernel.org/pub/linux/\
kernel/v2.4/linux-2.4.20.tar.bz2
[...]
user@gcux:~$ wget \
ftp://ftp.us.kernel.org/pub/linux/\
kernel/v2.4/linux-2.4.20.tar.bz2.sign
[...]
```

2. We get the signing key: <http://www.kernel.org/signature.html> gives the instructions.

```
user@gcux:~$ gpg --recv-keys 517D0F0E
gpg: key 517D0F0E: removed multiple subkey binding
gpg: key 517D0F0E: public key "Linux Kernel Archives <
Verification Key <ftpadmin@kernel.org>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

We could have checked the signature first, and it would have complained about not knowing the key that produced it: 517D0F0E.

3. First blank test to check the signature:

```
user@gcux:~$ gpg linux-2.4.20.tar.bz2.sign
gpg: Signature made Thu Nov 28 17:57:29 2002 CST using DSA key ID 517D0F0E
gpg: Good signature from "Linux Kernel Archives Verification Key <
<ftpadmin@kernel.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: C75D C40A 11D7 AF88 9981 ED5B C86B A06A 517D 0F0E
```

From a security perspective, we have not learned much: The signature is correct, but from a key we do not know.

---

4. Let us check its authenticity using the Debian keyring:

```
user@gcux:~$ gpg --keyring /usr/share/keyrings/debian-keyring.gpg \
--list-sigs "Linux Kernel Archives Verification Key <ftpadmin@kernel.org>"
```

5. We see that *Warren A. Layton (Debian)* <zeevon@debian.org> signed the key. We can import it, sign it locally and trust it because it came with the Debian CD that we trust:

```
user@gcux:~$ gpg --keyring /usr/share/keyrings/debian-keyring.gpg \
-a --export BFB880A3 | gpg --import
gpg: key BFB880A3: public key "Warren A. Layton (Debian) <zeevon@debian.org>"
imported
gpg: Total number processed: 1
gpg: imported: 1
user@gcux:~$ gpg --lsign BFB880A3
```

```
gpg: checking the trustdb
gpg: checking at depth 0 signed=1 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: checking at depth 1 signed=1 ot(-/q/n/m/f/u)=0/0/0/0/1/0
gpg: checking at depth 2 signed=0 ot(-/q/n/m/f/u)=1/0/0/0/0/0
gpg: next trustdb check due at 2004-01-24
pub 1024D/BFB880A3 created: 2000-08-19 expires: never trust: -/-
sub 1024g/4196F4F9 created: 2000-08-19 expires: never
(1) Warren A. Layton <zeevon@netwinder.org>
(2). Warren A. Layton (Debian) <zeevon@debian.org>
```

Really sign all user IDs? **yes**

```
pub 1024D/BFB880A3 created: 2000-08-19 expires: never trust: -/-
Primary key fingerprint: F54C 019D 18BE 6ED8 678D 39D0 21FD D515 BFB8 80A3

Warren A. Layton <zeevon@netwinder.org>
Warren A. Layton (Debian) <zeevon@debian.org>
```

How carefully have you verified the key you are about to sign actually belongs to the person named above? If you don't know what to answer, enter "0".

- (0) I will not answer. (default)
- (1) I have not checked at all.
- (2) I have done casual checking.
- (3) I have done very careful checking.

Your selection? **2**

Are you really sure that you want to sign this key  
with your key: "Test User (Test User for GCUX practical) <user@gcux>"

The signature will be marked as non-exportable.

---

I have checked this key casually.

Really sign? **yes**

You need a passphrase to unlock the secret key for  
user: "Test User (Test User for GCUX practical) <user@gcux>"  
1024-bit DSA key, ID 1123481F, created 2003-05-09

user@gcux:/usr/src\$ **gpg --edit-key BFB880A3**  
gpg (GnuPG) 1.2.1; Copyright (C) 2002 Free Software Foundation, Inc.  
This program comes with ABSOLUTELY NO WARRANTY.  
This is free software, and you are welcome to redistribute it  
under certain conditions. See the file COPYING for details.

gpg: checking the trustdb  
gpg: checking at depth 0 signed=2 ot(-/q/n/m/f/u)=0/0/0/0/0/1  
gpg: checking at depth 1 signed=2 ot(-/q/n/m/f/u)=1/0/0/0/1/0  
gpg: checking at depth 2 signed=0 ot(-/q/n/m/f/u)=1/0/0/0/0/0  
gpg: next trustdb check due at 2004-01-24  
pub 1024D/BFB880A3 created: 2000-08-19 expires: never trust: -/f  
sub 1024g/4196F4F9 created: 2000-08-19 expires: never  
(1) Warren A. Layton <zeevon@netwinder.org>  
(2). Warren A. Layton (Debian) <zeevon@debian.org>

Command> **trust**  
pub 1024D/BFB880A3 created: 2000-08-19 expires: never trust: -/f  
sub 1024g/4196F4F9 created: 2000-08-19 expires: never  
(1) Warren A. Layton <zeevon@netwinder.org>  
(2). Warren A. Layton (Debian) <zeevon@debian.org>

Please decide how far you trust this user to correctly  
verify other users' keys (by looking at passports,  
checking fingerprints from different sources...)?

- 1 = Don't know
- 2 = I do NOT trust
- 3 = I trust marginally
- 4 = I trust fully
- 5 = I trust ultimately
- m = back to the main menu

Your decision? **4**

pub 1024D/BFB880A3 created: 2000-08-19 expires: never trust: f/f  
sub 1024g/4196F4F9 created: 2000-08-19 expires: never  
(1) Warren A. Layton <zeevon@netwinder.org>  
(2). Warren A. Layton (Debian) <zeevon@debian.org>

---

Please note that the shown key validity is not necessarily correct unless you restart the program.

Command> **quit**

## 6. We check the signature of the Linux kernel for good:

```
user@gcux:/usr/src$ gpg linux-2.4.20.tar.bz2.sign
gpg: Signature made Thu Nov 28 17:57:29 2002 CST using DSA key ID 517D0F0E
gpg: Good signature from "Linux Kernel Archives Verification Key"
gpg: checking the trustdb
gpg: checking at depth 0 signed=2 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: checking at depth 1 signed=2 ot(-/q/n/m/f/u)=0/0/0/0/2/0
gpg: checking at depth 2 signed=0 ot(-/q/n/m/f/u)=2/0/0/0/0/0
gpg: next trustdb check due at 2004-01-24
```

## D Apt-release-check

This script can be downloaded from <http://people.debian.org/~ajt/apt-check-sigs>. A good way to trust and understand it is to read it.

This script is also available in the documentation of the *harden-doc* package<sup>36</sup>.

```
#!/bin/bash
# This script is copyright (c) 2001, Anthony Towns
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.

rm -rf /tmp/apt-release-check
mkdir /tmp/apt-release-check || exit 1
cd /tmp/apt-release-check

>OK
>MISSING
>NOCHECK
>BAD
```

---

<sup>36</sup>The script resides in `/usr/share/doc/harden-doc/html/securing-debian-howto`

---

```

arch='dpkg --print-installation-architecture'

am_root ()
[ 'id -u' -eq 0 ]

get_md5sumsize ()
    cat "$1" | awk '/^MD5Sum:\/,/^SHA1:\/' |
    MYARG="$2" perl -ne \
    '@f = split /\s+\/; if ($f[3] eq $ENV"MYARG") print "$f[1] $f[2]\n"; exit(0); '

checkit ()
    local FILE="$1"
    local LOOKUP="$2"

    Y="'get_md5sumsize Release "$LOOKUP"'"
    Y="'echo "$Y" | sed 's/^ *//;s/ */ /g'"

    if [ ! -e "/var/lib/apt/lists/$FILE" ]; then
        if [ "$Y" = "" ]; then
            # No file, but not needed anyway
            echo "OK"
            return
        fi
        echo "$FILE" >>MISSING
        echo "MISSING $Y"
        return
    fi
    if [ "$Y" = "" ]; then
        echo "$FILE" >>NOCHECK
        echo "NOCHECK"
        return
    fi
    X="'md5sum < /var/lib/apt/lists/$FILE' `wc -c < /var/lib/apt/lists/$FILE`"
    X="'echo "$X" | sed 's/^ *//;s/ */ /g'"
    if [ "$X" != "$Y" ]; then
        echo "$FILE" >>BAD
        echo "BAD"
        return
    fi
    echo "$FILE" >>OK
    echo "OK"

echo
echo "Checking sources in /etc/apt/sources.list:"
echo "~~~~~"
echo

```



---

```

(echo "You should take care to ensure that the distributions you're downloading"
 echo "are the ones you think you are downloading, and that they are as up to"
 echo "date as you would expect (testing and unstable should be no more than"
 echo "two or three days out of date, stable-updates no more than a few weeks"
 echo "or a month).")
) | fmt
echo

cat /etc/apt/sources.list |
sed 's/^ *//' | grep '^[^#]' |
while read ty url dist comps; do
    if [ "$url%:*" = "http" -o "$url%:*" = "ftp" ]; then
        baseurl="$url#*://"
    else
        continue
    fi
    echo "Source: $ty $url $dist $comps"

    rm -f Release Release.gpg
    wget -q -O Release "$url/dists/$dist/Release"

    if ! grep -q '^' Release; then
        echo " * NO TOP-LEVEL Release FILE"
    else
        origline=`sed -n 's/^Origin: */p' Release | head -1`
        lablline=`sed -n 's/^Label: */p' Release | head -1`
        suitline=`sed -n 's/^Suite: */p' Release | head -1`
        codeline=`sed -n 's/^Codename: */p' Release | head -1`
        dateline=`grep "^Date:" Release | head -1`
        dscrline=`grep "^Description:" Release | head -1`
        echo " o Origin: $origline/$lablline"
        echo " o Suite: $suitline/$codeline"
        echo " o $dateline"
        echo " o $dscrline"

        if [ "$dist%/*" != "$suitline" -a "$dist%/*" != "$codeline" ]; then
            echo " * WARNING: asked for $dist, got $suitline/$codeline"
        fi

        wget -q -O Release.gpg "$url/dists/$dist/Release.gpg"
        sigline="`gpgv --status-fd 3 Release.gpg Release 3>&1 >/dev/null 2>&1 \
| sed -n 's/^[GNUPG:] GOODSIG [0-9A-Fa-f]* //p'"`"
        if [ "$sigline" ]; then
            echo " o Signed by: $sigline"
        else
            echo " * NO VALID SIGNATURE"
            >Release
        fi
    fi
fi

```

---

```

okaycomps=""
for comp in $comps; do
    if [ "$ty" = "deb" ]; then
        X=$(checkit "`echo "$baseurl/dists/$dist/$comp/binary-$arch/Release" \
| sed 's,/*,_,g'`" "$comp/binary-$arch/Release")
        Y=$(checkit "`echo "$baseurl/dists/$dist/$comp/binary-$arch/Packages" \
| sed 's,/*,_,g'`" "$comp/binary-$arch/Packages")
        if [ "$X $Y" = "OK OK" ]; then
            okaycomps="$okaycomps $comp"
        else
            echo " * PROBLEMS WITH $comp ($X, $Y)"
        fi
    elif [ "$ty" = "deb-src" ]; then
        X=$(checkit "`echo "$baseurl/dists/$dist/$comp/source/Release" \
| sed 's,/*,_,g'`" "$comp/source/Release")
        Y=$(checkit "`echo "$baseurl/dists/$dist/$comp/source/Sources" \
| sed 's,/*,_,g'`" "$comp/source/Sources")
        if [ "$X $Y" = "OK OK" ]; then
            okaycomps="$okaycomps $comp"
        else
            echo " * PROBLEMS WITH component $comp ($X, $Y)"
        fi
    fi
done
[ "$okaycomps" = "" ] || echo " o Okay:$okaycomps"
echo
done

echo "Results"
echo "~~~~~"
echo

allokay=true

cd /tmp/apt-release-check
diff <(cat BAD MISSING NOCHECK OK | sort) \
<(cd /var/lib/apt/lists && find . -type f -maxdepth 1 \
| sed 's,^\./,,g' | grep '_' | sort) | sed -n 's/^> //p' >UNVALIDATED

cd /tmp/apt-release-check
if grep -q ^UNVALIDATED; then
    allokay=false
    (echo "The following files in /var/lib/apt/lists have not been validated."
    echo "This could turn out to be a harmless indication that this script"
    echo "is buggy or out of date, or it could let trojaned packages get onto"
    echo "your system."
    ) | fmt
    echo
    sed 's/^/    /' < UNVALIDATED

```

---

```

    echo
fi

if grep -q ^ BAD; then
    allokay=false
    (echo "The contents of the following files in /var/lib/apt/lists does not"
     echo "match what was expected. This may mean these sources are out of date,"
     echo "that the archive is having problems, or that someone is actively"
     echo "using your mirror to distribute trojans."
     if am_root; then
         echo "The files have been renamed to have the extension .FAILED and"
         echo "will be ignored by apt."
         cat BAD | while read a; do
             mv /var/lib/apt/lists/$a /var/lib/apt/lists/$a.FAILED
         done
         fi) | fmt
    echo
    sed 's/^/    /' < BAD
    echo
fi

if grep -q ^ MISSING; then
    allokay=false
    (echo "The following files from /var/lib/apt/lists were missing. This"
     echo "may cause you to miss out on updates to some vulnerable packages."
    ) | fmt
    echo
    sed 's/^/    /' < MISSING
    echo
fi

if grep -q ^ NOCHECK; then
    allokay=false
    (echo "The contents of the following files in /var/lib/apt/lists could not"
     echo "be validated due to the lack of a signed Release file, or the lack"
     echo "of an appropriate entry in a signed Release file. This probably"
     echo "means that the maintainers of these sources are slack, but may mean"
     echo "these sources are being actively used to distribute trojans."
     if am_root; then
         echo "The files have been renamed to have the extension .FAILED and"
         echo "will be ignored by apt."
         cat NOCHECK | while read a; do
             mv /var/lib/apt/lists/$a /var/lib/apt/lists/$a.FAILED
         done
         fi) | fmt
    echo
    sed 's/^/    /' < NOCHECK
    echo
fi

```

---

```
if $alokay; then
    echo 'Everything seems okay!'
    echo
fi

rm -rf /tmp/apt-release-check
```

© SANS Institute 2003, Author retains full rights.

---

## References

- [1] ALFRED J. MENEZES, PAUL C. VAN OORSCHOT, S. A. V. *Handbook of Applied Cryptography*. CRC Press, 1996, ch. 9 - Hash Functions and Data Integrity.
- [2] BRENNEN, V. A. Gnupg keysigning party howto.  
<http://www.cryptnet.net/fdp/crypto/gpg-party.html>, May 2003.
- [3] BRENNEN, V. A. Strong distribution howto.  
[http://www.cryptnet.net/fdp/crypto/strong\\_distro.html](http://www.cryptnet.net/fdp/crypto/strong_distro.html), April 2003.
- [4] BUCHANAN, M. *Nexus: Small Worlds and the Groundbreaking Science of Networks*. W.W. Norton & Company, 2002.
- [5] CVE. Can-2002-0684 (under review).  
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0684>, 2002.
- [6] DEBIAN. Debian developers' corner. <http://www.debian.org/devel>, May 2003.
- [7] DEBIAN. Debian gnu/linux - security information.  
<http://www.debian.org/security/>, May 2003.
- [8] FOUNDATION, T. F. S. The gnu privacy handbook.  
<http://www.gnupg.org/gph/en/manual.html>, 1999.
- [9] INC, R. S. Rsa laboratories - cryptography faq - what is a hard problem?  
<http://www.rsasecurity.com/rsalabs/faq/2-3-1.html>, 2003.
- [10] INITIATIVE, O. S. Open source definition.  
<http://www.opensource.org/docs/definition.php>.
- [11] JOHN VIEGA, G. M. *Building Secure Software*. Addison-Wesley, 2002, ch. 4 - On Open Source and Closed Source.
- [12] MARK J COX, JOE ORTON, M. M. T. Apache week. vendor modified apache versions. <http://www.apacheweek.com/issues/03-02-21>, February 2003.
- [13] POWER, M. Vulnerabilities in operating-system patch distribution.  
<http://razor.bindview.com/publish/papers/os-patch.html>, December 2000.  
BindView Corporation, RAZOR Team.
- [14] SLEGGERS, W. The comp.security.pgp faq.  
<http://www.cam.ac.uk.pgp.net/pgpnet/pgp-faq/>, 2002.
- [15] STEPHENS, J. C. Iptables: Connection tracking.  
<http://www.sns.ias.edu/~jns/security/iptables/rules.html>, October 2002.