Global Information Assurance Certification Paper

Securing a Redhat 7.2 Linux machine on a home network

Martin L. Purschke

Introduction

With the widespread availability of broadband Internet connections in private homes, home computers have become a new target for hackers, and the threats to the security of home computers have increased significantly. While traditional dial-up connections are by no means inherently secure, dial-up systems used to be connected to the Internet for relatively short periods of time only, and then with dynamically assigned and changing IP addresses, which offered some measure of "security by obscurity". And a dial-up connection did not normally offer high bandwidth, which made a successful attack more difficult. Even if a system was compromised, the usefulness of the system for the attacker was again limited by the relatively low bandwidth of the connection. On top of that, with a system online through a dial-up connection, there is a higher chance that the user is actively using it to browse the Web or checking Email, so there is a higher risk of detection.
All that changes with an upgrade to an "always-on" broadband connection through DSL or a Cable modem. Typically, the IP addresses are either static or at least quasi-static. The author's former DSL connection through the now-defunct provider NorthPoint featured a truly static IP address, and the current Cable service provides a dynamic address through PPPoE, which, however, appears not to change over a period of at least several months. In this way, a home system that is left running is exposed to the Internet with a fixed address and open to attacks. The static IP address allows an attacker to come back to a once-compromised system, and the high-bandwidth connection will enable him or her to down- or upload files unnoticed.

In a professional setting, there is usually a system administrator available who is supposedly trained to set up a secure network and systems. Companies routinely have firewalls and closely monitored proxy servers in place. Those give some level of protection, which a directly connected home PC often does not have. In addition, a private user is less likely to involve law enforcement agencies than a company whose systems have been compromised. All these factors make home computers attractive for hackers. They could read your files or mail, use the system as a "lilypad" to attack other machines, or to launch denial-of-service attacks from the compromised system. There are indications that recent denial-of-service (DOS) attacks were launched from compromised home computers [1].

Another difference between a professional and a home environment is the number of machines involved. It is easier for a company to dedicate machines to just a single task, and harden them appropriately. For example, one would set up a dedicated login server, a dedicated web server, a dedicated syslog server, and the eventual number of machines plays a minor role in the design decisions. At home, there is a strong incentive to keep the number of machines low. Usually there is only one computer, or each household member has a dedicated machine. Each computer takes up floor space, costs money for the initial investment and in networking hardware and the cost to

install cables, and uses power (at current energy rates, it costs about $250/year to operate a 200W PC around the clock). Therefore, home computers are more likely to be general-purpose machines, and "general purpose" at home can mean a much broader spectrum of activities than in a professional setting.

This document describes the considerations I had and the steps that I took to harden my home machines against attacks. It could serve as a guide for a home computer user considering an upgrade to a broadband Internet connection. In the last chapter I explain and present a bootable "Computer Forensics" CD-ROM which can be downloaded from the Internet. It contains the most-wanted tools to examine a compromised system, and it will help you collecting the evidence you will need if you involve the police.

Before you begin

In this document I will describe a setup where a router between my private network and the Internet acts as a firewall at the same time. I strongly advise that you get such a router/firewall, too. They cost significantly less than $100 these days, and in addition to dramatically increasing the security, they offer a lot of features which I learned to appreciate, such as connecting more than one computer, network printers, and so on.

Of course it is possible to harden your system appropriately without a standalone firewall, and you will read about my host-based firewall configuration later on. But how often have you found yourself just disabling that firewall momentarily if you couldn't get a service to connect? Just to see if it is indeed the firewall that prevents your machine from connecting? Or you just read about a security flaw discovered in a service that you had open to the net for the past 6 months? Or if you have a dual-boot machine and boot into Windows, how sure are you that it does not connect to the network? Or that that setup is secure enough? Are you sure that the game that you just installed for your children does not re-enable the network in order to look for updates? Or that it does not disable whatever firewall you installed?

Behind a standalone firewall/router, you are not defenseless in such a situation, and the security of your computer does not depend exclusively on having the host-based firewall on and properly configured at all times.

So, for your peace of mind, go and buy a router. You will learn to appreciate it.

The Home Setup

I run a private 192.168 network routed to the cable modem through a Linksys BEFSR41 router, which also provides a DHCP server on the LAN side. My house is cabled with CAT5 wiring in several rooms, and I operate a wireless 802.11b network through a Linksys WAP11 access point for a notebook. The Linksys router also acts as a firewall, which will be important in the risk analysis down below. The LAN hosts 3 computers: my main home machine running Redhat 7.2 (which is the subject of this article), my notebook, also running Redhat 7.2, and the "family PC", normally used by my wife to read email and access the web, and for my children to play games on. This machine runs Windows 98.

My main computer is a DELL Dimension XPS T550. It has a 550MHz Intel Pentium III processor,

512MB of RAM, an 80GB harddisk, a DVD/CD-ROM and a second CD-RW drive. It has a voice modem and a 3Com Fast Ethernet network card. A special piece of hardware is a Hauppauge Digital TV card that allows to watch (and capture) TV or in general video signals.

The machine is a dual-boot machine (RedHat 7.2 and Windows98). I keep the Windows98 system almost exclusively to watch DVD movies occasionally; the machine runs the Linux operating system most of the time. The current kernel version is 2.4.18.

I run the X-Windows software with the GNOME desktop environment [2]. Most of the KDE [3] components (which are usually supported by the GNOME environment) are installed as well; there are very good utilities that come with KDE, such as kmail for reading mail, and kcalc, which is a very good programmer's calculator, just to name a few.

I use the Mozilla Browser [4], currently version 0.9.8, as my standard browser, and to read mail. Mozilla supports both imap- and POP-based mail accounts, which makes it a good choice. However, some Web sites do not recognize its identification and refuse to serve pages. For example, www.myccinfo.com, where you can access your credit card information online, insists on the Netscape browser. For that reason, I use Netscape (version 6.2.1) on occasion to access those sites.

This machine is used to read email, browse the web, order goods online, perform financial transactions such as online banking, online stock trading and online bill payments. It holds our family photo album, and runs a Web server to access the album from the other machines on the LAN. It also runs a digital answering machine with the voice modem. In addition, I use the machine to telecommute to machines at work.

It runs the Apache [5] web server (version 1.3.20-16). For the answering machine, it runs mgetty-voice version 1.1.26-6.

As far as services which open a network port go, in routine mode the machine runs sshd (port 22), httpd (port 80), and X11 (port 6000). Other services (such as Samba on the LAN and NFS) are started manually on demand only, and shut down again when they are no longer needed.

Risk Analysis

Overall, a compromise of my home system, while not entirely fatal, could potentially put the confidentiality of my email, my financial records, and other private information at risk. In order to analyze the risk, it is necessary to look not just at the main machine running Linux, but also at the network environment in which this machine operates.

There are the following considerations to "environmental" security:

- Someone could break into my house and steal the main computer (or any other). This would certainly compromise a good amount of private information, but would *not* compromise my financial online accounts (no passwords are stored on the machines). It would compromise my passwords, which are not used anywhere else, and also some ssh keys that I use to access other machines, in particular my machines at work. However, such a break-in would not go undetected for long, and appropriate steps to limit the damage could be taken.
- The Linksys router acts as a firewall. I need to be able to log into my home system from outside,

and the only port open from outside is port 22 (ssh). The router forwards incoming ssh requests to my main machine. Unless someone has developed a way to defeat the Linksys firewall firmware, port 22 is the only way in from the outside, so it is important to keep the ssh daemons on the main machine on the latest patch level, and engage the tcp wrappers.

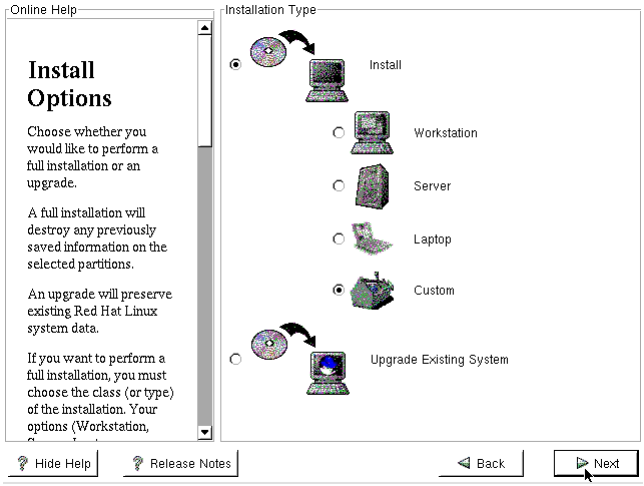
- Someone with a notebook and a wireless network card could get on my wireless network, and access services that are open to the LAN only ("drive-by-hacking"), just use my resources to get on the Internet, or sniff packets on the wireless network. The WAP11 offers 40-bit WEP encryption, which I use. However, a 40-bit key is not really strong encryption, and of course, WEP encryption has been broken recently [6], so a dedicated and modestly resourceful attacker could conceivably park his car in the neighborhood and successfully access my home network. The Linksys firmware allows selecting the MAC addresses of the interfaces which can access the network. This limits the threat to people who are knowledgeable enough to spoof MAC addresses. In addition, a new undocumented feature was discovered just recently, which allows controlling the output power setting of the WAP transmitter [7]. In this way, I can reduce the power of the transmitter so that the signal fades within a short distance of my house. The coverage does not "leak" beyond the boundaries of my property, and would require substantial hardware (dish antennae, etc) to capture. Therefore, I consider operating the wireless network an acceptable risk.
- The Windows PC could get infected with a virus that attacks my computer from the LAN, or leaks information to an outside location. The latter scenario is certainly more likely than the former. The Windows machine runs ZoneAlarm (www.zonealarm.com), which blocks all (in and out) ports except a few standard ones. This PC is checked for viruses regularly.
- On a case-by-case basis, I give visitors in my home access to my home network. I usually power down my main machine when we leave the house with such visitors present. The concern is not that they would attack my machines, but that their notebooks are infected with a virus that scans the network, etc.
- A firewall is an absolute necessity, and the choice of the Linksys router is very convenient and cost-effective. It does not, however, provide firewall logs, so it is not possible to detect patterns or recurring probing attacks.
- I keep a small Windows 98 installation on my home machine for the sole purpose of watching DVD movies. I have no incentive to match the security measures described here for a Windows system.While in Windows mode, the machine does not connect to the LAN.

Installation

I have operated a Linux machine since the days of RedHat Version 5.1. When I bought the DELL machine, the system got transferred from the old machine to the new computer. That machine has since undergone several operating system and hardware upgrades (more memory, and a bigger harddisk). When the upgrade to Redhat Version 7.2 was due, I decided to start the installation completely from scratch, in order not to have to deal with legacy configuration files and other programs or utilities which might pose a security risk. I only restored my user directories from a backup later.

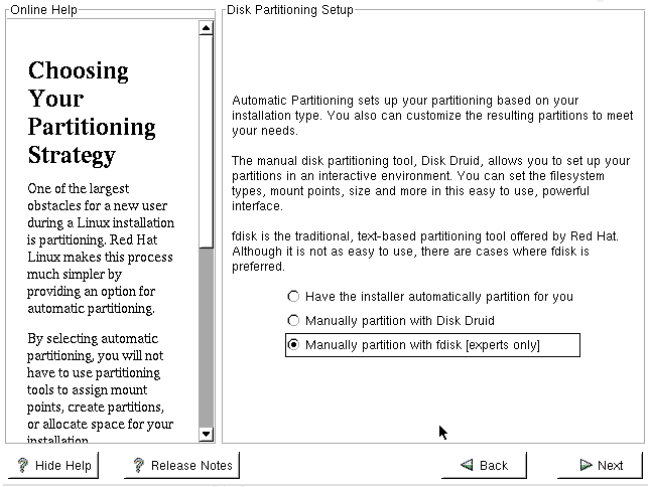
Custom Install

There is nothing special about the installation process. Because it is important to restrict the installed packages to the minimum you need, select “Custom Install” in order to have a more fine-grained control over what gets installed. It not only saves disk space, but if a package is not installed, a potential hacker cannot take advantage of it.



Disk Partitioning

I like to partition the disk using the traditional `fdisk` utility. Since the system is a dual-boot setup, there is a constraint that the first partition, `/dev/hda1`, or “`C:`” for Windows, must be a primary partition. For a detailed description of a dual-boot setup, consult ref. [8].



If you are intimidated by the fdisk utility, by all means, use Disk Druid. I just happen to be comfortable with fdisk, which does for me what I need, and gives me all the control I want.

When I installed this system on the new harddisk several months ago, I set up the partitions on the disk in a way which I know now was not optimal. I did *not* use individual partitions for the root file system (/) and the /usr, /var, and /tmp file systems. This would allow to mount / and /usr read-only, and increase the security, because a possible flaw in the permissions would still not allow to write or change files in the /usr file system. If an attacker could gain root access, he would need to re-mount

the file system for write in order to make changes. This cannot be undone without rebooting the system, and would be easily noticeable.

I will correct this flaw as soon as I find the time. In a moment I will show a transcript of a `fdisk` session that performs the proper partitioning. First I show here my current partition table for completeness only:

```
# fdisk /dev/hda
Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 9726 cylinders
Units = cylinders of 16065 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       261   2096451    b  Win95 FAT32
/dev/hda2           262      9726  76027612+   5  Extended
/dev/hda5           262       913   5237158+  83  Linux
/dev/hda6           914      3040  17085096   83  Linux
/dev/hda7          3041      3224   1477948+  82  Linux swap
/dev/hda8          3225      9204  48034318+  83  Linux
/dev/hda9          9205      9726   4192933+   b  Win95 FAT32
```

I chose to format the Linux partitions as `ext3` file systems. I also keep yet another FAT32 partition of 4 Gb, which was used in the past for files which I needed to read when the machine booted in Windows mode (Windows cannot easily read ext2 or ext3 partitions). The relevant mounts are shown here:

```
# mount
/dev/hda5 on / type ext3 (rw)
/dev/hda8 on /data type ext3 (rw,nosuid,nodev)
/dev/hda6 on /home type ext3 (rw,nosuid,nodev)
/dev/hda1 on /win98 type vfat (rw,nosuid,nodev)
/dev/hda9 on /wxchange type vfat (rw,gid=31013,uid=1980,umask=002)
```

The options for the /wxchange partition ("rw,gid=31013,uid=1980,umask=002") assign values to the files which the file system does not support natively. My group id is 31013 and my user id is 1980, so those options make the files appear to be owned by me (although the vfat file system does not have an ownership concept). The umask value specifies the default permission bits that are *not* present. This value of 002 generates a default permission of 775 (`"-rwxrwxr-x"`; the `"w"` permission for "other" users is missing).

A proper Partition Table

In order to show a proper partition setup for the reader to use, I got hold of another disk of the same size, and performed the partitioning with the fdisk utility. This is the proper way the partition table might look:

```
Device Boot    Start       End    Blocks   Id  System
/dev/hda1          1       262   2104483+   b  Win95 FAT32
/dev/hda2        263      9726  76019580    5  Extended
/dev/hda5        263       352    722893+  83  Linux
```

```
/dev/hda6          353      875   4200966   83  Linux
/dev/hda7          876     1006   1052226   83  Linux
/dev/hda8         1007     1137   1052226   83  Linux
/dev/hda9         1138     1316   1437786   82  Linux swap
/dev/hda10        1317     2361   8393931   83  Linux
/dev/hda11        2362     2884   4200966    b  Win95 FAT32
/dev/hda12        2885     9726  54958333+  83  Linux
```

With a disk size of 80GB, we can afford to be generous with space. With a smaller disk, we probably want to trim down the allocations a bit. I give the sizes that I chose, and the sizes that I consider the minimum for a general-purpose system. They are only guidelines. I have found that the space needed in /usr has grown significantly over time, from about 1.1GB in a RedHat 5.1 system in the past, to well above 2GB with RedHat 7.2 (which, of course, depends on what you choose to install). I consider the 4GB allocated for the /usr file system safe for many versions to come.

If you have only a small disk, you probably do not want separate /home and /data file systems. The rule of thumb for the size of the swap partition is "2.5 times the size of the RAM", which holds for RAM sizes of up to 256MB. For larger RAM sizes, the factor can be lower. I used 1400MB, which is *really* generous, and is driven by the virtual memory requirements of a particular program ("Persistence of Vision", a high-quality ray-tracing program [8]), which I use once in a while. A swap size of 1GB, which would be a factor of 2 of my 512MB RAM, is probably enough for most other applications.

The desired partition map is given in the following table:

Partition Nr	Mount point	My size (MB)	Minimum size
1	/win98 (vfat)	2048	n/a
5	/	700	50
6	/usr	4096	1500
7	/var	1024	500
8	/tmp	1024	300
9	swap	1400	1024
10	/home	8192	n/a
11	/wxchange (vfat)	4096	n/a
12	/data	55,000	n/a

Here is a transcript of the fdisk session which generated the above partition table. We start with a primary partition of 2048MB for the Windows system, and another "extended" partition that takes the rest of the whole disk. This extended partition will contain the other logical partitions (5 through 12). The repeating pattern is the command "n" (new), followed by the choice of primary, extended,

or logical partition. I always accept the suggestion of the *first* cylinder, which is always the next available one if we go in sequence. Except for the extended and the final partition, which take up the remainder of the disk, I specify the sizes in MB, e.g. "+2048M". In the end, I change the partition type for partition 9 to "swap" (hex code 82), and partition 1 and 11 to "vfat" (hex code b). This is done with the command "t".

```
# fdisk /dev/hdb

The number of cylinders for this disk is set to 9726.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-9726, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-9726, default 9726): +2048M
```

Here we make the extended partition, which takes up the rest of the disk:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 2
First cylinder (263-9726, default 263):
Using default value 263
Last cylinder or +size or +sizeM or +sizeK (263-9726, default 9726):
Using default value 9726
```

Here we start the series of the logical partitions 5 through 12. Note that now that we have an extended partition, we are given the "logical" choice for the first time.

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (263-9726, default 263):
Using default value 263
Last cylinder or +size or +sizeM or +sizeK (263-9726, default 9726): +700M

Command (m for help): n
```

```
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (353-9726, default 353):
Using default value 353
Last cylinder or +size or +sizeM or +sizeK (353-9726, default 9726): +4096M
```

This continues now for the other partitions, giving 1024, 1024, 1400, 8192, and 4096MB sizes as given in the table above. This brings us to the last partition, where we just accept the last cylinder and get the remainder of the free space:

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (2885-9726, default 2885):
Using default value 2885
Last cylinder or +size or +sizeM or +sizeK (2885-9726, default 9726):
Using default value 9726
```

We now use the print ("p") command to see what we have:

```
 Command (m for help): p

Disk /dev/hdb: 255 heads, 63 sectors, 9726 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1       262   2104483+  83  Linux
/dev/hdb2           263      9726  76019580    5  Extended
/dev/hdb5           263       352    722893+  83  Linux
/dev/hdb6           353       875   4200966   83  Linux
/dev/hdb7           876      1006   1052226   83  Linux
/dev/hdb8          1007      1137   1052226   83  Linux
/dev/hdb9          1138      1316   1437786   83  Linux
/dev/hdb10         1317      2361   8393931   83  Linux
/dev/hdb11         2362      2884   4200966   83  Linux
/dev/hdb12         2885      9726  54958333+  83  Linux
```

and will now adjust the type of the swap partition (9) and the two vfat partitions (1 and 11).

```
Command (m for help): t
Partition number (1-12): 9
Hex code (type L to list codes): 82
Changed system type of partition 9 to 82 (Linux swap)

Command (m for help): t
Partition number (1-12): 1
Hex code (type L to list codes): b
Changed system type of partition 1 to b (Win95 FAT32)
```

```
Command (m for help): t
Partition number (1-12): 11
Hex code (type L to list codes): b
Changed system type of partition 11 to b (Win95 FAT32)

Command (m for help): p

Disk /dev/hdb: 255 heads, 63 sectors, 9726 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1       262   2104483+   b  Win95 FAT32
/dev/hdb2           263      9726  76019580    5  Extended
/dev/hdb5           263       352    722893+  83  Linux
/dev/hdb6           353       875   4200966   83  Linux
/dev/hdb7           876      1006   1052226   83  Linux
/dev/hdb8          1007      1137   1052226   83  Linux
/dev/hdb9          1138      1316   1437786   82  Linux swap
/dev/hdb10         1317      2361   8393931   83  Linux
/dev/hdb11         2362      2884   4200966    b  Win95 FAT32
/dev/hdb12         2885      9726  54958333+  83  Linux
```

That looks ok. We now exit and commit our changes with the "w" command:

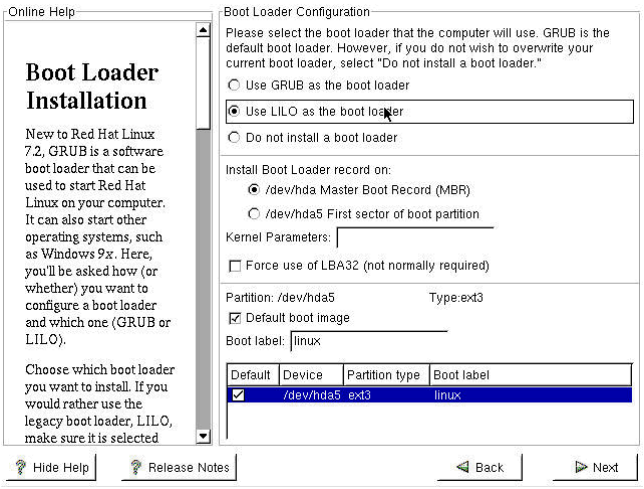
```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
#
```

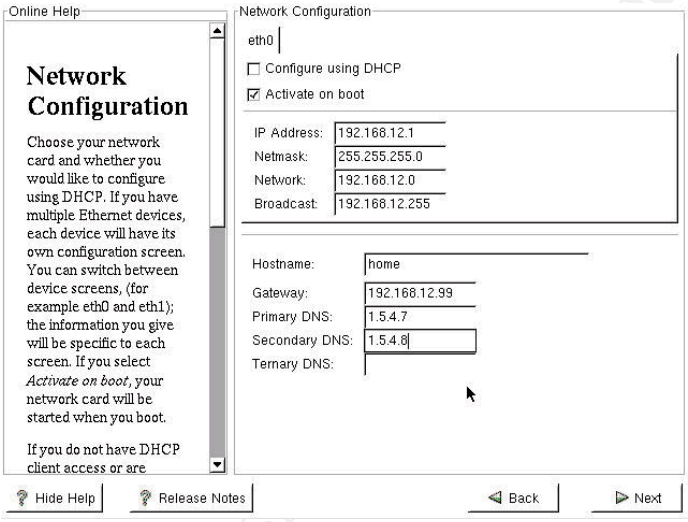
Boot Loader Setup

We select `lilo` as the boot loader, and choose to install it in the master boot record (MBR). Note that an existing Windows partition (which is not on the disk that I used to make those screen shots) will show up in the list of bootable partitions in the lower field.



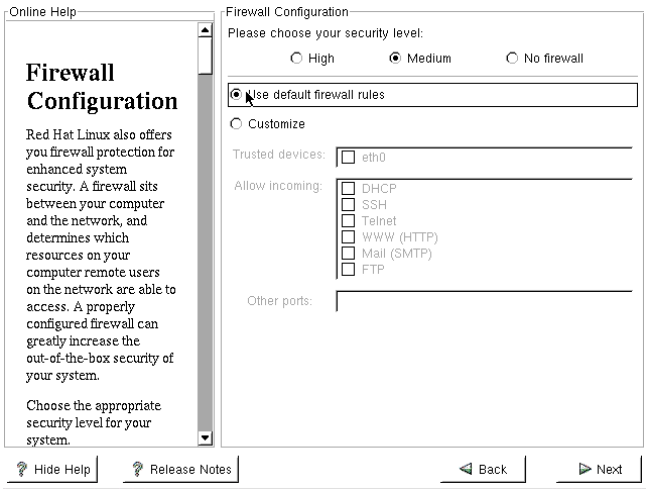
Network Configuration

In this section you will set up the network parameters for the network interface card. On my private network 192.168.12.x, I chose 192.168.12.1 for my main machine. The router setup prescribes the other parameters (the gateway address, the netmask, and the broadcast address). Your Internet Service Provider or your network manager will tell you the addresses of the domain name servers (DNS), and also the other parameters in case you do not have a router. (Ask your ISP. You cannot use those numbers I give here, nor can you just make those numbers up!)



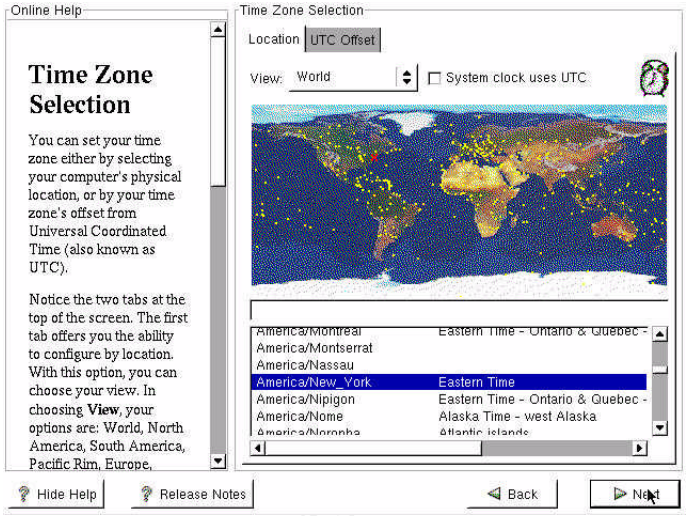
Firewall

I always run a host-based firewall on my machines, which protects against packets which get through the external firewall. I choose “Medium” security level and “Use default firewall rules”, because I will tweak the rules manually later on anyway. Note that enabling this firewall will enable `ipchains`. `ipchains` is labeled legacy software, and `iptables` is more modern and can offer a more sophisticated firewall setup. However, `iptables` comes at the price of a steep learning curve, and I had worked with `ipchains` for a long time, so I decided not to switch to `iptables` for the time being. `ipchains` will offer adequate protection for our purposes here.



Language and Time Zone Selection

You can select the languages that your system will support, and set the time zone where you live in. The only somewhat complicated issue here is a choice in the time configuration, which allows you to specify that the "System clock uses UTC".

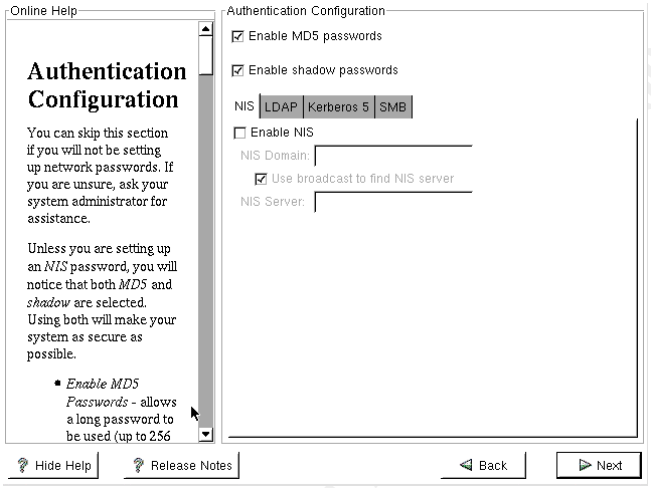


UTC stands for "Universal Time Code". If your system is a single-boot machine, check that. With a dual-boot system, there *may* be a problem. In principle, each Unix machine in the world uses UTC, that is, it counts the seconds since Jan 1, 1970 on the meridian 0 (which passes through Greenwich, England). In theory, a machine in Tokyo and a machine in New York have, except for slight inaccuracies of their clocks, the same value of this counter. The fact that the machines are in different time zones is accounted for by different offsets to that same counter value, which is selected by the time zone specification. Linux behaves properly in this respect, and will do the proper adjustments. The problem comes when the Linux OS shuts down, and you reboot the machine in Windows. Linux will synchronize the hardware clock on the motherboard with the value of its system clock in UTC. Your Windows OS may not know that the time value on the hardware clock is actually the time in England, and when Windows boots, your clock will be off by

that time offset, that is, several hours.
For a dual-boot system, it is generally recommended *not* to check that box. If you find that you took the wrong choice, you can later use the `timeconfig` utility to set things right. For a thorough discussion of the time settings, see, for example, ref. [10].

Account Configuration and Passwords

In the next section, you will set a good root password, and establish user accounts (on a home machine, typically for yourself only). There are several good guides on the Web on how to pick a good password [11,12]. In the section thereafter, select "MD5 passwords" and "shadow passwords". Using shadow passwords allows to make the user information (which is needed for virtually everything that displays user information, for example, the "`ls -l`" command) available without the sensitive encrypted password string. MD5 passwords are much more secure than standard DES-based passwords, because they take about a factor of 1000 more time to crack than a standard one. The only reason not to select the "Shadow" and "MD5" options would be a setup which uses NIS (which conflicts with the use of shadow passwords), or a password file shared between different operating systems which don't all support MD5 passwords. This not the case here.

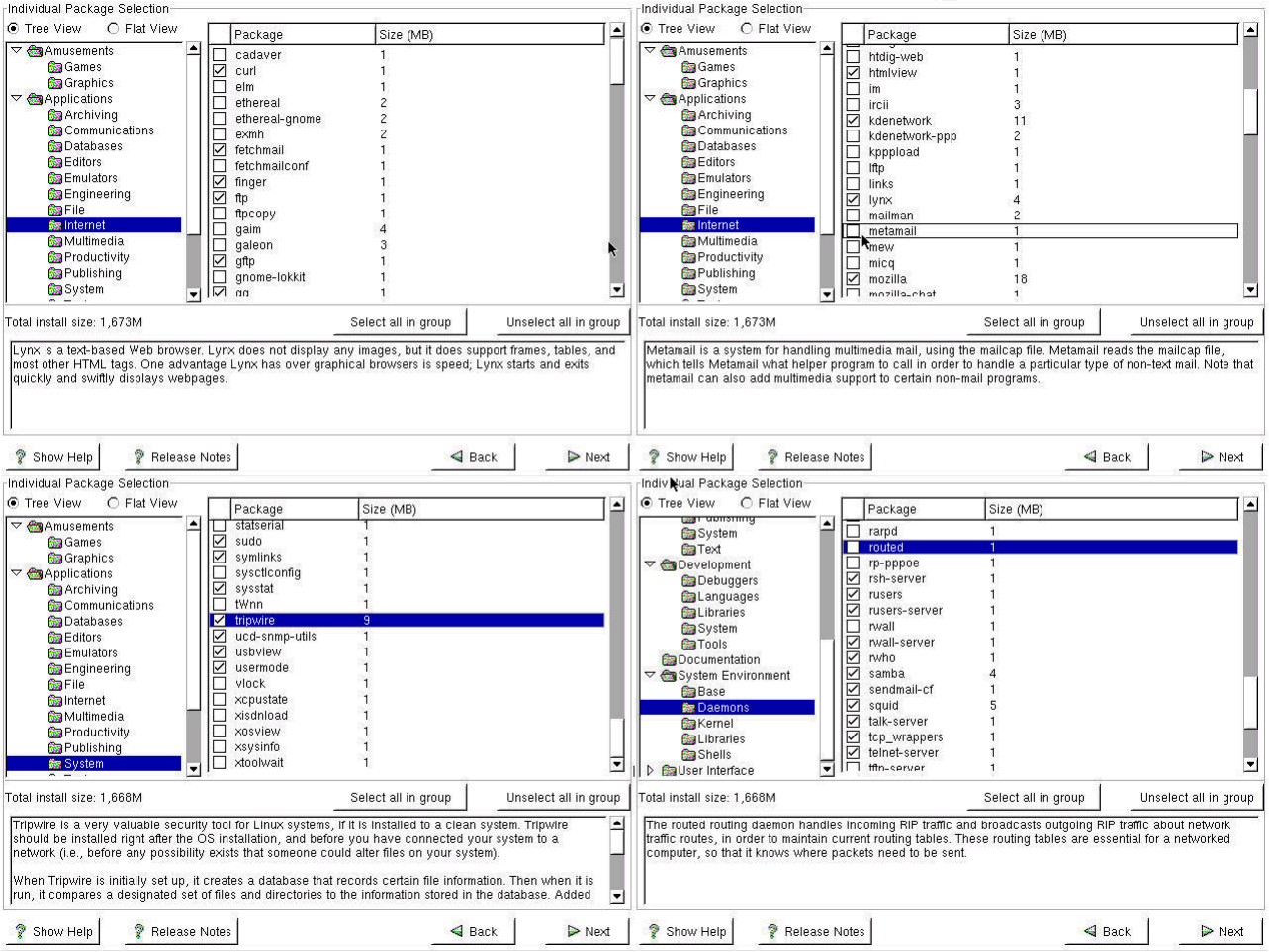


In the next section, I check all except "Laptop support", "News server", "Anonymous ftp server", "SQL database server", "DNS name server", "Kernel development", and, obviously, "Everything".

Note: I make a choice here not to check “Kernel Development” (despite the fact that I will install a newer kernel later). I find it easier not to have the kernel sources under RPM control. In a later chapter, I will show how to automatically install new RPM packages. This procedure assumes that the kernel setup will not be touched by such an automated update, that is, your built-to-suit kernel cannot be removed or altered by updating a RPM file. However, if you have no idea how to build a

kernel, or you just want to use the kernel that RedHat provides with the system upgrades, go ahead and check this option. In this procedure, you install a RPM for the kernel itself (not the kernel sources) that is "benign" in this respect – it will install itself, but does not contain any configuration files.

I check "Select individual packages", and check or uncheck packages that I obviously need or don't need. In particular in the "Internet" section are lots of programs that I will not use. For example, there are several e-mail clients, which I do not install. Also, there are 3 or 4 ftp clients, and I just keep the classic "ftp" package. I make sure that I have `Tripwire`, `nmap`, and `tcpdump` checked. I make sure that `routed` is not installed. Here are a few screen shots how I selected the packages:



In Appendix A is a complete table of the selected packages, in the order they appear in the above "tree view". You can consult the table, but keep in mind that that table reflects my personal preferences and requirements and your selection will likely be different.

Try to be restrictive here. As a general rule, if you do not know what a package does, click on the name, and you will see a short description as shown in the previous screen shots. If you cannot envision that you will ever use it, do not select it here. There is hardly any risk that you de-select packages that are needed by other packages, because after the selection is complete, the installer will notify you of unsatisfied dependencies, if any. You will have the option to go back and change your selection, to leave those dependencies unresolved (which is not recommended), or to add the

required packages. I chose to add the required packages. Also keep in mind that you can install more packages from the RedHat CD later once the system is set up, so it is better to start with a minimum system here.

This concludes the security-related part of the installation setup.

Post-installation tasks

After the system has booted the Linux operating system for the first time, several steps must be taken to enhance the security and reliability of the new system. It has to be stressed that the computer must not be connected to the network at this point. Only after these tasks are complete, it can be connected.

The steps that we will take now are

- Modify `/etc/fstab` to enhance security
- Upgrade the kernel to the latest stable version
- Backup the Master Boot Record (MBR) and the kernel configuration
- Upgrade software packages to the latest version
- Shutting down unnecessary services and daemons
- Install third-party software
- Set up and establish a list of "suid root" programs
- Configuring the xntp daemon to synchronize the clock
- Establish a host-based firewall, and configure the tcp wrapper
- Review and change configuration files
- Lock down sensitive configuration and other files
- Activate and configure `Tripwire`
- Install and configure `logcheck`
- As we approach a stable setup, establish a BIOS password if so desired.

The Tripwire step comes late in the list because the former steps usually add or modify files, and would cause lots of false alarms for Tripwire.

Modifying /etc/fstab

My system has two CD-ROM drives (one DVD player and one CD-Writer), and a floppy, which I would like to be user-mountable. The corresponding entries in `/etc/fstab` look like this

```
/dev/fd0   /mnt/floppy auto   noauto,user,noexec,nodev,nosuid   0 0
/dev/cdrom  /mnt/cdrom  iso9660 noauto,user,exec,nodev,nosuid,ro 0 0
/dev/cdrom1 /mnt/cdrom1 iso9660 noauto,user,exec,nodev,nosuid,ro 0 0
```

which means

- `noauto` - mounted on request only, not on boot
- `user` - a user is able to mount and dismount (the same user— if another user should be able

to dismount, the keyword is "users")
- exec - allow execution of binaries from the device
- nodev - do not recognize special device files on those devices
- nosuid - do not honor a suid bit on an executable on those devices.

Since executable programs are rarely found on floppy disks these days, I chose not to allow to run programs from a floppy, which also protects against scripts. The nodev and nosuid options make sure that no suid executables and no special device files are recognized on those devices, in case a carefully prepared CD (such as an installation CD) contains executables with the suid bit set, or special device files such as /dev/mem. Those special device files, if enabled, would allow a user to access the devices and bypass protections if the ownership is set carefully.

I also add the nosuid and nodev options to the other "user" filesystems (/data and /home):

```
/dev/hda5    /        ext3    defaults                1 1
/dev/hda8    /data     ext3    rw,exec,nodev,nosuid      1 2
/dev/hda6    /home     ext3    rw,exec,nodev,nosuid      1 2
```

Upgrading the kernel

I usually install the latest stable kernel version and keep it current, within reason. Redhat 7.2 ships with kernel 2.4.7. Since versions before 2.4.10 had a serious security hole [13,14], I upgraded to the then-stable version 2.4.16. I recently upgraded to the currently latest version 2.4.18. Because I have a number of special devices (such as the TV card, a Digital Camera, and a Web camera), I need to make a custom kernel in any case.

Keep in mind that it is quite ok not to install your own kernel, as long as you perform the recommended upgrades. The kernel that comes with the installation will work with your system, unless you have some very specific hardware (such as my digital camera) that needs special kernel support. RedHat is reasonably security-conscious, and will usually release security patches shortly after they become available [15]. A new kernel RPM with a fix of the above-mentioned bug was made available a short time later. If you decide to use the RedHat-supplied kernel, there is nothing wrong with that decision. In this case you should probably add the kernel sources in the package list when you install.

If you decide to build your own kernel, you can get support for devices that are somewhat experimental, and have the latest and greatest bug fixes and drivers. You may increase the performance of the system because you can compile the kernel for the processor that that you have, and get all the optimizations. For a detailed discussion of building your own kernel, consult the Kernel-HOWTO [16].

I usually leave the original kernel in place, label it "original" in the lilo.conf file, and add the new kernel as the default one (this is from my previous kernel 2.4.16, I show the file after the upgrade to version 2.4.18 in the "maintenance" section):

```
$ cat /etc/lilo.conf
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
```

```
prompt
timeout=50
message=/boot/message
linear
default=linux

image=/boot/vmlinuz-2.4.16
       label=linux
       read-only
       root=/dev/hda5
       append="hdd=ide-scsi"

image=/boot/vmlinuz-2.4.7-10
       label=original
       initrd=/boot/initrd-2.4.7-10.img
       read-only
       root=/dev/hda5
       append="hdd=ide-scsi"

other=/dev/hda1
       optional
       label=win98
```

After the kernel is installed, and `lilo` has been run, make a backup of the Master Boot Record (MBR) on a floppy, just as an insurance in case something goes wrong some day. While we are at it, also copy the kernel configuration (which is a hidden file called `.config` in the kernel source directory /usr/src/linux) to the floppy. If you play with the kernel configuration and make a mistake, you will be able to go back to a version that works.

```
mount /mnt/floppy
dd if=/dev/hda of=/mnt/floppy/mbr.dat bs=512 count=1
cp /usr/src/linux/.config /mnt/floppy/kernel_config.2.4.16
umount /mnt/floppy
```

The dd command will write the first 512 bytes (the MBR) to the file. In addition to being a safety measure against corruption of the MBR, it is another file that can show how the system had been configured before an attack.

Upgrading to the latest versions of the software

At work I maintain a CD-ROM with the latest software upgrades from ftp://updates.redhat.com/7.2/en/os/. I mount the CD, cd to the directory, and update existing packages with the command

```
find . -name "*.rpm" -exec rpm -Fvh {} \; |& tee r.log
```

The "-F" options requests "freshen"; a RPM is installed only if a previous version already exists.

Remember: This assumes that the kernel sources are either not under RPM control in case you built

your own custom kernel, or that you use the RedHat-supplied kernel. Where I work, virtually everyone who might use the CD has a custom-built kernel, so I do not download the kernel-related RPM's when I make the CD. In this way, the kernel cannot be inadvertently changed.

The above command can require some manual intervention at times, because packages often have interdependencies, which are resolved only if the dependent RPM's are installed in one execution of the command. Still, it is very easy to keep the software up-to-date.

In light of the above discussion of `ssh` as the only way into my system from the outside, these updates are absolutely essential for the integrity of the system. In the `OpenSSH` packages that ship with the standard Redhat 7.2 CD, a security flaw had been discovered [17], which I eliminated with the updates. In the meantime, another ssh exploit has been found [18,19], which required an upgrade to OpenSSH version 3.1.

Shutting down unnecessary services and daemons

This step comes after the upgrade step because the upgrade might re-enable a service at boot time.

Our goal here is to have the absolute minimum set of services running. The fewer there are, the fewer can be exploited. As I explained in the risk analysis, the only services which open a port on the network and which are running all the time are sshd, httpd, and X11.

Services are started at boot time, and their behavior is controlled by a set of symbolic links in `/etc/rc.d/rc{x}.d/`, where x denotes the run level. Usually my machine runs in run level 5 (X-Windows mode). I use `chkconfig` to maintain the services, and the command

```
chkconfig —list | grep 5:on
```

will show the services which are started at boot time for run level 5. You should also test the other run levels, and see if there are additional services which start in level 3 and 4 (levels 1 and 2 do not start the network in the first place, and are less easy to exploit).

The command

```
chkconfig <service> off
```

will prevent a service from starting up the next boot. It is important to note that the service is not shut down; `chkconfig` controls only the behavior at boot time. In order to terminate a service in the running system, use

```
/etc/rc.d/init.d/<service> stop
```

One of the important services to think about is `xinetd`, which starts service programs (such as `telnetd`) on demand, when it recognizes a request on the corresponding port, and if that is enabled in the configuration files.

The two main services, `sshd` and `httpd`, are not normally started by `xinetd`, so unless there is a compelling reason to run the service, it is better to shut it down:

```
chkconfig xinetd off
/etc/rc.d/init.d/xinetd stop
```

It is unlikely that you will receive mail directly on your machine. Most likely, you will send and receive Email through a mail server provided by your ISP. Normally, there is no reason to start the sendmail daemon.

If the system does not run NFS or Samba servers, the portmapper does not need to run. The fewer services are active, the fewer doors are open for attacks to exploit.

I run Samba and NFS services on demand only, usually to access large files from the Windows system or my laptop (which don't have very large disks). If I need to mount file systems, I start the services at that point, and shut them down again when I'm done.

My machine runs an internal HTTP server in order to distribute our family picture albums on the internal network between the machines. Also, I use my machine as a convenient phone answering machine, and use the web server to distribute voice mail. A voice modem is controlled by `mgetty-voice` (http://alpha.greenie.net/vgetty), plays a greeting, and records a message, if the caller chooses to leave one. If a voice message is recorded, mail is sent to my wife and me, specifying the URL of the sound file.

This means that care must be taken to confine the Web server to the LAN, because it could reveal sensitive information. This is accomplished by the external firewall blocking HTTP requests from the Internet, the `httpd` configuration files, and the host-based firewall in the machine, which will be established further down.

Another concern is that the voice modem, running `vgetty`, will not only answer calls and faxes, but also allow dial-in connections into my machine. I disabled the dial-ins in a configuration file controlling this, and I verified that the modem does not accept dial-in connections.

Installing third-party software

There are several third-party software packages installed on my machine. One with some impact on the configuration is Vmware (http://www.vmware.com/), which allows to run "virtual" machines as guest operating systems on a Linux machine. Vmware installs kernel modules and also sets up yet another machine-internal private network, for which I set up `ipchains`-based masquerading (NAT) and enable packet forwarding. It also establishes an internal Samba server for (virtual) machines to access the host file systems. I chose 192.168.22.0 as the Vmware private network.

The other security-related third-party software package that I install is PGP (http://web.mit.edu/network/pgp.html, and http://www.pgpi.org for non-US residents), which is used to encrypt email and sensitive local files. In addition, PGP provides a convenient way to "wipe" a file (with `pgp -w`). Before deleting the file, it is overwritten with a random bit pattern several times, so that the contents of a deleted file cannot be recovered.

There are many more aspects of PGP which are relevant for the security of your system. I have summarized a few of them in Appendix B.

Setting up and establishing a list of "suid root" programs

The suid bit is set by the command

```
chmod +s <filename>
```

In addition to the ones that come with the installation, I have three "suid root" programs for which I set the suid bit myself.

`smbclient` needs `suid` privileges to allow a non-root user to mount Samba shares on other computers; I need to mount disks on my wife's computer on occasion.

`gphoto2` (http://www.gphoto.org/) is a program which allows to access and download images from a variety of digital cameras. For cameras accessed through the Universal Serial Bus (USB), the program needs root privileges to access the bus.

Finally, I gave suid privileges to `cdrecord`, the utility that writes CD's.

In order to obtain the complete list of programs with the suid bit set, use the command

```
find / -type f -perm +4000 -print
```

or

```
find / -type f -perm +4000 -ls
```

This will print a list, but for your records, you should keep some more information about the files in order to find out if they were altered in any way (we will later see that tripwire can do that automatically). Still, you may want to use the command

```
find / -type f -perm +4000 -exec md5sum {} \; > $HOME/suidprograms.list
```

Here are the first few lines of that file that I get:

```
# head -10 $HOME/suidprograms.list
e50031a9ac3702b60d774d41165f05b0  /usr/bin/cdrecord
ce4193f06fde12ccfa46f11d4e413ae6  /usr/bin/at
f046c15fd146b739f490023c8c9377f2  /usr/bin/suidperl
f046c15fd146b739f490023c8c9377f2  /usr/bin/sperl5.6.0
b0ad395f473f023f394731f1b5b9719d  /usr/bin/rcp
fd1aaddda43bc5dcc8d8e3716b4e1667  /usr/bin/rlogin
329a73d54844460852a6bb4e42323c39  /usr/bin/rsh
e4049b9fff52b08354e88587f118faae  /usr/bin/chage
d9a96734d97159daf513e25be43f6cc4  /usr/bin/gpasswd
af17c55ca4ebf31a9efe9f557aad3655  /usr/bin/passwd
```

If I suspect later that a file has been altered, I have a convenient reference.

Configuring NTP (Network Time Protocol)

It is important that the machine has its clock set accurately, or it will be impossible to establish an exact timeline of events, which may taint the evidence collected from the system. We do this at this point because we need to configure the firewall according to our choice of time servers. I selected local NTP servers from a list maintained at http://www.eecis.udel.edu/~mills/ntp/clock2.htm. I settled on a set of servers with an "open access" policy. It is recommended to set up at least three

different servers in different locations, so a problem with one server cannot affect your clock setting. Here are the non-comment lines from the configuration file (`/etc/ntp.conf`), where only the "server" lines has been changed from the default:

```
server ntp.ctr.columbia.edu
server sundial.columbia.edu
server ntp2.mpis.net
fudge 127.127.1.0 stratum 10

driftfile /etc/ntp/drift
broadcastdelay     0.008
authenticate no
```

The ntpd startup file `/etc/rc.d/init.d/ntpd` needs another file, `/etc/ntp/step-tickers`, which must contain the server names again.

```
# cat /etc/ntp/step-tickers
ntp.ctr.columbia.edu
sundial.columbia.edu
ntp2.mpis.net
#
```

Establishing a host-based firewall

Even though there is a hardware firewall that shields the LAN from all but `ssh` traffic, a host-based firewall increases the security of the system significantly. In general, it is better to have multiple lines of defense. There is always the possibility that someone finds a way to defeat the firewall firmware of the Linksys router. The external firewall does not filter outgoing packets, which the host-based firewall can (and will) be set up to do. Attacks can also come from the LAN through malicious programs or potentially through the laptop, which might get compromised while online somewhere else. The tcp wrapper is not sufficient to prevent exploits of the `ssh` daemon. One of my machines at work, inadvertently exposed on port 22, was compromised with a `ssh` exploit despite properly configured tcp wrappers [17]. The difference between tcp wrappers and a host-based firewall is that the latter prevents the packets from reaching the `ssh` daemon in the first place, while tcp wrappers instruct the daemon to reject certain connection requests. Without the firewall, the packets from unauthorized hosts reach the daemon and can get through its defenses by exploiting buffer overflow bugs.

As you continue the setup of your system, you will need to update the firewall rules. I show here the final configuration:

```
# cat /etc/sysconfig/ipchains
#
:input ACCEPT
:forward DENY
:output ACCEPT
# leave the commented-out accounting rule in if needed
#-A input -s 0/0 -d 0/0 -p tcp -l
#
```

```
# accept ssh connection from my machines at work
-A input -s 1.2.3.6. -d 0/0 22 -p tcp -y -j ACCEPT
-A input -s 1.2.3.7 -d 0/0 22 -p tcp -y -j ACCEPT
-A input -s 1.2.3.8  -d 0/0 22 -p tcp -y -j ACCEPT
#
# and all connections from my private network
-A input -s 192.168.12.0/24 -d 0/0 -j ACCEPT
-A input -s 192.168.22.0/24 -d 0/0 -j ACCEPT
#
# also accept all stuff from DNS
-A input -p udp -s 0/0 -d 0/0 53  -j ACCEPT
# also accept all stuff on the local interface
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
#
# but reject all other incoming connection from outside
-A input -p tcp -l  -s 0/0 -d 0/0  -y -j DENY
#
# allow the time servers
-A input -p udp   -s 128.59.64.60  -d 0/0 123 -j ACCEPT
-A input -p udp   -s 128.59.59.177 -d 0/0 123 -j ACCEPT
-A input -p udp   -s 216.152.230.3 -d 0/0 123 -j ACCEPT
#
# but reject the rest on udp
-A input -p udp -l  -s 0/0 -d 0/0 0:2049 -j DENY
#
# kill web bug reports to doubleclick.com
-A output -s 0/0 -d 208.184.29.0/24 -j DENY
-A output -s 0/0 -d 199.95.206.0/24 -j DENY
#
# finally, the masquerading rule for the internal VmWare network.
-A forward -s 192.168.22.0/24 -d 0/0 -j MASQ
```

The firewall allows `ssh` requests from my two main machines at work (denoted by 1.2.3.x), all connections from my private networks, and all from the local interface. It denies and logs (with the "`-l`" in the first two "`DENY`" lines) all other requests on a tcp port with the "SYN" bit set (the –y flag), which requests a new connection to be established. It does not filter packets from an already established connection such as a ssh session. In this way I can see who is trying to log into my machine through ssh from an unauthorized host. I can also see if there are any connections on other ports that make it through the external firewall (which should be closed). The firewall also blocks most low-numbered UDP ports, and logs the denied requests. We need to allow the traffic from the name server on port 53 and the time servers on port 123. Because UDP is a connection-less protocol, there is no equivalent to the "SYN" bit as in TCP packets, and there is no way to distinguish established connections from connection requests. That is why we cannot block the high-numbered UDP ports. The highest blocked port 2049 is the NFS server port, which needs protection. In my setup there are no open ports above that limit. If you are running special services such as a MySQL server, you may need to expand the list of blocked UDP ports.

The last line is the "masquerading" rule for the Vmware network.

The two remaining "`DENY`" lines are there to prevent an outright invasion of privacy by the company DoubleClick Inc. (http://www.doubleclick.net) [20], which arms web advertisements with

so called Web Bugs [21]. Those can track your movements around the Web, and this information, in addition to raising privacy concerns, may also be useful to an attacker. The 208.184.29.0/24 class C network has the current address of `ad.doubleclick.net` (208.184.29.130), where most of DoubleClick's web bugs report back to. The firewall stops packets to all addresses on that class C network, because DoubleClick may change the IP addresses of that host name. The other "DENY" line blocks packets to the hosts `doubleclick.net` and `doubleclick.com`; again the whole class C network is blocked.

Setting up the tcp wrappers

The tcp wrapper offers yet another line of defense against unauthorized connections. The configuration files (`/etc/hosts.deny` and `/etc/hosts.allow`) look like this:

```
# cat /etc/hosts.deny
ALL:ALL


# cat /etc/hosts.allow
sshd,sshdfwd-X11: 1.2.3.6
sshd,sshdfwd-X11: 1.2.3.7
sshd,sshdfwd-X11: 1.2.3.8

sshd,sshdfwd-X11: 192.168.12.0/255.255.255.0
```

The "deny" file specifies that all connections are denied unless explicitly allowed in the "allow" file. My 3 machines at work, already seen in the firewall configuration file, can use sshd, and are allowed to use the ssh X11 forwarding feature. Any machine on the home network can use sshd and forward X11 traffic. Note that the Vmware network is not mentioned in the allow file. The Samba server running for the 192.168.22 network ignores the tcp wrapper, and the masquerading is handled on a kernel level, bypassing the tcp wrapper. The fact that the virtual machines have no access rights with the tcp wrapper prevents them from accessing services under wrapper control on the host machine, another layer of protection in case a virtual Windows machine gets infected by a virus.

Reviewing and changing configuration files

There are several configuration files that need to be adjusted in order to implement the security requirements of the system.

Again, with ssh as the only way in from outside, the sshd configuration files are among the most important ones. From the default settings, I modified the configuration (`/etc/ssh/sshd_config`) so that

- ssh protocol version 2 is the only one allowed;
- authorized hosts (there are only three of them outside my home network) are required to have a pre-established host key in the system area (`/etc/ssh/ssh_known_hosts2`); an entry in a user's (my) `$HOME/.ssh/known_hosts2` file will not suffice;
- I also changed to "`PermitRootLogin no`", so in order to get root access, I have to log on as

myself, then use `su`;
- I enabled the "Banner" line so that before the login, a login warning is printed. More about this further down.

I did, however, choose to allow "`PasswordAuthentication`". The alternative would be the use of ssh keys. That, together with the `ssh-agent`, might allow logins from my unattended machine at work, in addition to the risk that the ssh key file might get copied from my machines (my desktop and my laptop), allowing an attacker a brute-force password cracking attack. My judgement call is that a password file on my home machine is more secure than a ssh key file on the machines at work.

Here is the `/etc/ssh/sshd.config` file:

```
# This is the sshd server system-wide configuration file.  See sshd(8)
# for more information.
Port 22
Protocol 2
#ListenAddress 0.0.0.0
#ListenAddress ::
HostKey /etc/ssh/ssh_host_key
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
#
# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
IgnoreUserKnownHosts yes
StrictModes yes
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
#PrintLastLog no
KeepAlive yes
# Logging
SyslogFacility AUTHPRIV
LogLevel INFO
#obsoletes QuietMode and FascistLogging
RhostsAuthentication yes
#
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
RhostsRSAAuthentication yes
# similar for protocol version 2
HostbasedAuthentication yes
#
RSAAuthentication yes
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
PermitEmptyPasswords no
```

```
# Uncomment to disable s/key passwords
#ChallengeResponseAuthentication no
# Uncomment to enable PAM keyboard-interactive authentication
#PAMAuthenticationViaKbdInt yes
# To change Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no
# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes
#CheckMail yes
#UseLogin no
#MaxStartups 10:30:60
Banner /etc/issue.net
#ReverseMappingCheck yes
Subsystem    sftp   /usr/libexec/openssh/sftp-server
```

The commented-out "ReverseMappingCheck" entry defaults to "yes", which requires that all machines connecting to the sshd daemon must have a name associated with them. Machines on the private network *must* have a name as well, or the access will be denied. Addresses obtained from the LAN-internal DHCP server (which is provided by the Linksys router) do not have a name entry, and would not be able to connect to the ssh daemon. I have to give them a name in /etc/hosts file, discussed next.

The /etc/hosts file

The `/etc/hosts` file can assign names to IP addresses, which can also override a DNS entry. I use the file to assign names to IP addresses on my private network, which of course do not have official DNS entries. I find it easier if I can refer to my machines by name. With the sshd_configuration file shown in the previous chapter, a machine *must* have a name associated with its IP address. I added the last two lines to the file, which allow my laptop to connect, and also enable connections from the main machine itself:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1          localhost.localdomain localhost

192.168.12.1       home
192.168.12.100     laptop
```

The inittab file

On any UNIX system, the init process has the ID 1 and is the "mother" of all other processes in the system. Its behavior is controlled by the /etc/inittab file, which controls default run levels, shutdown procedures, and many more aspects of the init process. For our purposes here, there are two weaknesses that need attention. One is the line with a preceding comment

```
# Trap CTRL-ALT-DELETE
```

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

This allows the system to be rebooted by pressing the ctrl-alt-delete keys, and we do not want just anyone to be able to do that. So we disable that line by placing a "#" in front of the line, commenting it out.

The other problem is that the RedHat standard inittab file does not require a password if you boot into single user mode, but gets you root access without any further checks. (You boot into single-user mode by typing "linux 1" at the lilo boot prompt). This can be fixed by adding a line after the "si::sysinit" line, so the section reads

```
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
~~:S:wait:/sbin/sulogin
```

I believe that Redhat ships that inittab file in order to provide a "back door" into the system in case you forget the root password but have physical access to the system. This added line will make sure that you must enter the root password if you boot into single-user mode.

That said, it can still be defeated easily enough. An attacker could enter

```
linux init=/bin/sh
```

at the lilo prompt, and get access. But this trick seems not to be widely known, so the line that we added to the inittab file adds some measure of additional security. Even so, keep in mind that "physical access to a machine means root access".

Gnome configuration file

For the same reasons why we turned off the ctrl-alt-delete shutdown method, we want to disable the "System" menu in the Gnome login dialog, which would enable any user to shutdown or reboot the machine. Edit the file /etc/X11/gdm/gdm.conf, and change the line

```
SystemMenu=true
```

to

```
SystemMenu=false
```

and the menu will not be there. While you edit the file, remove the actual commands given in the file to shutdown or reboot the machine,

```
HaltCommand=/usr/bin/poweroff
RebootCommand=/sbin/shutdown -r now
```

to

```
HaltCommand=
RebootCommand=
```

Syslogd and Log file Rotation

In case of a breakin, the log files are the most important documents that can help establish what happened. The log files are handled by the syslog facility, which is controlled by a file `/etc/syslog.conf`. I did not make changes to that file, but you might see the need to change some entries (for example, if you have a different machine for your logs). There is an extensive man page available for syslogd.conf (type "man syslogd.conf").

The log files are rotated in predetermined intervals. If you would continue writing to the same file, it would become very long, and would eventually exceed the 2GB file limit or fill up the available space on the /var disk, where those logfiles usually go. The rotation of log files is controlled by the file `/etc/logrotate.conf`. I changed the default settings to keep 24 weeks worth of log files, and to compress the rotated files. Being text files, they compress nicely to very small sizes.

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 24 weeks worth of backlogs
rotate 24

# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own lastlog or wtmp—we'll rotate them here
/var/log/wtmp {
monthly
create 0664 root utmp
rotate 1
}

# system-specific logs may be also be configured here.
```

The HTTP server

The server is protected against outside access by the external firewall blocking port 80, the host-based firewall blocking all HTTP requests from any non-LAN IP address, and, finally, the `/etc/httpd/conf/http.conf` configuration file. In the file that ships with the standard Apache server, there are several occurrences of

```
Order allow,deny
Allow from all
```

I changed that to

```
Order deny,allow
Deny from all
Allow from 192.168.12.0/255.255.255.0
```

so that only LAN IP addresses are authorized to access pages on this server. For more information, see http://www.apache.org.

Samba servers

With the installation of the Vmware virtual machine software, there are two Samba servers present on the system, one serving only the 192.168.22 network for the virtual machine(s), and the "main" server serving the LAN network. The latter one is started on demand only.

There are lots of comments and explanations in the respective `/etc/samba/smb.conf` and `/etc/vmware/vmnet1/smb/smb.conf` configuration files. Additional information can be found in the Samba HOWTO document [22].

Entries to watch for are the "socket address" and "interfaces" lines, which restrict the server to listening on the particular network only.

The LAN server configuration file reads

```
hosts allow = 192.168.12.0/255.255.255.0 127.
socket address = 192.168.12.1
interfaces = 192.168.12.0/255.255.255.0
bind interfaces only = yes
```

while the Vmware configuration is

```
hosts allow = 192.168.22.0/255.255.255.0
socket address = 192.168.22.1
interfaces = 192.168.22.0/255.255.255.0
bind interfaces only = yes
```

NFS Server

As stated above, the NFS server is started manually on demand only, when I need to access large files hosted on the main machine from my laptop, which wouldn't fit on the smaller local disk. The NFS server's behavior is controlled through the `/etc/exports` file.

```
# cat /etc/exports
/data  laptop(ro)
```

In order to start the NFS server, I need to start the portmapper first, and then the NFS service:

```
/etc/rc.d/init.d/portmap start
/etc/rc.d/init.d/nfs start
```

To shut down again, I stop the services in reverse order:

```
/etc/rc.d/init.d/nfs stop
/etc/rc.d/init.d/portmap stop
```

Setting up login warnings

For legal reasons, it is important that the system displays a login banner. Consult your lawyer for a good wording for your purpose. There are two different "banners", /etc/issue.net, which is displayed before the user logs in, and /etc/motd, which is displayed after a successful login. The behavior is controlled by the sshd_config file with the options

```
PrintMotd yes
Banner /etc/issue.net
```

I think that it makes more sense to display a warning *before* someone logs in and can still reconsider, so I use the "Banner" entry. My machine announces

```
This Computer System is for authorized use only. Any or all uses of
this system and all files on this system may be intercepted,
monitored, recorded, copied, audited, inspected, and disclosed to law
enforcement personnel. By using this system, you consent to such
interception, monitoring, recording, copying, auditing, inspection,
and disclosure.

By continuing to use this system you indicate your awareness of and
consent to these terms and conditions of use. DO NOT LOGON if you are
not an authorized user.
```

Locking down sensitive files

Once an attacker has gained root access to a machine, he or she controls your system, and can essentially do anything. You can make the life of an attacker (and the automated attack scripts) significantly harder by setting the "i" ("immutable") attribute for sensitive files using

```
chattr +i <filename>
```

This is not shown by `ls -l`, you need to use `lsattr` to see it. It acts on the file system level and prevents the file from being changed beyond the protection offered by the mode bits. Set for a directory, it prevents files from being created or deleted, which would modify the directory file (but existing files can be modified unless protected by the "i" attribute themselves). In my experience, the chattr command is very rarely used in installation scripts. This gives the possibility of renaming the chattr and lsattr commands to some bogus name, for example, /usr/X11R6/bin/xcttr and /usr/X11R6/bin/xlttr, which might not be easily identifiable to a hacker, who will need to put extra effort in to get hold of the utilities.

I usually lock down all `*.conf` files in the `/etc` tree using

```
find /etc/ -name "*.conf" -exec chattr +i {} \;
```

and

```
find /etc/sysconfig -type f -exec chattr +i {} \;
find /etc/sysconfig -type d -exec chattr +i {} \;
```

This protects all files and also directories, as discussed above (you need to change the files first

before you lock down the directories).

In addition, I protect `/etc/hosts`, `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/ssh/*`, `/etc/hosts.deny`, `/etc/hosts.allow`, `/etc/inittab`, `/etc/sendmail.cf`, `/etc/profile.d/*`, `/etc/cron.daily/`, `/etc/cron.weekly`, `/etc/cron.monthly`, `/etc/exports`, `/etc/fstab`, and so on.

Be careful with `/etc/resolv.conf`, though. When using DHCP, this file may be overwritten when the interface is brought up, and must not have the "i" attribute in this case. Making /etc/shadow "immutable" means that a user cannot change his password himself, but since I am the only user on my home system, this is not a problem.

It is also possible to lock down all binaries in `/usr/bin`, `/bin`, `/sbin`, and `/usr/sbin`, and also the system libraries to prevent trojan programs from being installed.

Setting up Tripwire

After this step, most of the system files are in a stable enough state to configure Tripwire. The details of the installation and setup process are widely available on the Web (see, for example, [23]).

After the initial installation and first check pass, expect hundreds of lines of false alarms, a lot of them about missing files that are explicitly listed in the policy file. It will take a few iterations to get rid of the false positives. Add whatever you think is necessary to the policy file (in my case, I added the Vmware configuration files and kernel modules).

Once satisfied with the amount of output generated by a `tripwire --check` command, I capture the output in a file, which I lock down and use as a baseline to detect differences with future output of the check. I use some bogus names to make that file less suspicious-looking to an intruder:

```
tripwire --check > /root/.aabbccdd.txt 2>&1
chmod 500 /root/.aabbccdd.txt
chattr +i /root/.aabbccdd.txt
```

I then run a daily cron job with the unsuspicious-looking name `/etc/cron.daily/tmp0watch`, which is a script that runs the tripwire check (by default, there is a `tmpwatch` script in that area, so a file called `tmp0watch` does not stand out). I capture the tripwire output, make a `diff` to the reference, and count the number of lines reported as different. On my system, a benign report differs from the reference in 6 lines (some lines have the date, some specify changing log filenames), which results in 18 lines of output from the `diff` command. I flag anything more as suspicious:

```
#!/bin/sh
REF=/root/.aabbccdd.txt
NEW=/root/.caa.txt
DIF=/root/.cbb.txt

#run tripwire, capture output in a new file
/usr/sbin/tripwire --check > $NEW 2>&1
```

```
# get the differences to the reference
/usr/bin/diff -b -B $REF $NEW > $DIF 2>&1

# how many lines are different?
LINES=`/bin/cat $DIF | /usr/bin/wc -l`

# 18 are ok (some dates, file names, etc) but more, ring the alarm
[ $LINES -gt 18  ] && {
echo -n "****   Tripwire reports differences  ****"
/bin/date
bin/cat $DIF
}
/bin/rm -f $DIF
```

This way of flagging suspicious activity is convenient because it allows keeping the tripwire database stable and just increase the limit on suspicious lines for a while, until I find the time for a thorough review. Rebuilding the database too often might result in losing information about genuine problems.

Once satisfied, I make a backup of all sensitive tripwire files on a CD-ROM.

Reviewing log files

I installed logcheck (http://www.psionic.com/abacus/logcheck) to monitor the log files. At its core, it is a relatively straight-forward script that will check the logfiles for suspicious lines, plus a few support utilities. It comes as a gzip'ed tar ball; I installed the latest version 1.1.1. The installation was simple, just unpack the tarball, and run "make linux":

```
# tar xfv logcheck-1.1.1.tar.gz
logcheck-1.1.1/
logcheck-1.1.1/src/
logcheck-1.1.1/src/logtail.c
logcheck-1.1.1/systems/
logcheck-1.1.1/systems/sun/
logcheck-1.1.1/systems/sun/logcheck.sh
logcheck-1.1.1/systems/sun/logcheck.hacking
 < more lines deleted>
# cd logcheck-1.1.1
# make linux
  <output deleted>
```

By default, this package wants to be installed in /usr/local/etc/. Unless there is a compelling reason to change that, install it there. You would need to change the main script if you choose a different area. You end up with a few files in the installation directory:

```
# ls -l /usr/local/etc/
total 36
-rw-------    1 root     root         1037 Jan 21 22:17 logcheck.hacking
-rw-------    1 root     root         1485 Jan 24 01:28 logcheck.ignore
-rwx------    1 root     root        10648 Jan 21 22:52 logcheck.sh
```

```
-rw-------    1 root     root          407 Jan 21 22:17 logcheck.violations
-rw-------    1 root     root           14 Jan 21 22:17 logcheck.violations.ignore
drwx------    2 root     root         4096 Mar 15 00:01 tmp
#
```

In the logcheck.sh main script, there is a line that specifies which user will get the reports per email. I changed the line

```
SYSADMIN=root
```

to read

```
SYSADMIN=my_real_email
```

because I want to receive the notifications in my regular email. If I need to go to the root account to read email first, there is a chance that I don't check the reports often enough. There are a few more configuration parameters which you need to modify only if you chose a different installation area.

Further down in the same script is a section that determines which logfiles are to be monitored.

```
# Linux Red Hat Version 3.x, 4.x
$LOGTAIL /var/log/messages > $TMPDIR/check.$$
$LOGTAIL /var/log/secure >> $TMPDIR/check.$$
$LOGTAIL /var/log/maillog >> $TMPDIR/check.$$
```

This looks ok for my system, but if you have a different configuration, you must change those lines to reflect your setup.

You can also choose to add more log files, say, the httpd error log, by appending

```
# Linux Red Hat Version 3.x, 4.x
$LOGTAIL /var/log/messages > $TMPDIR/check.$$
$LOGTAIL /var/log/secure >> $TMPDIR/check.$$
$LOGTAIL /var/log/maillog >> $TMPDIR/check.$$
$LOGTAIL /var/log/httpd/error_log >> $TMPDIR/check.$$
```

Keep in mind, though, that this will probably require extensive adjustments in the other configuration files to cut down the false positives.

By symlinking the logcheck.sh file to the /etc/cron.hourly area, cron runs the script once an hour.

```
# ln -s /usr/local/etc/logcheck.sh /etc/cron.hourly/
```

Logcheck has the policy to "report unless told to ignore", so in the beginning you will receive lots of false error messages. Take note of the ones which you don't want to receive and add them to the `/usr/local/etc/lockcheck.ignore` file. Over time, I added the following lines to the file:

```
kernel.*: scsi : aborting command due to timeout
kernel.*: SCSI bus is being reset
kernel.*: SCSI host 0 abort
kernel.*: h*: irq timeout
kernel.*: h*: ATAPI reset complete
```

```
syslogd 1.4.1: restart.
ntpd*: synchronisation lost
session closed for user*
```

The first five lines filter out reports about SCSI errors when a CD-writing operation times out, which happens occasionally when trying to erase a defunct CD-RW disk (when the disk has scratches, etc). I also filter out reports of the syslog daemon restarting on a log rotation, and the "synchronization lost" error of the ntp daemon. Finally, I don't want to get notified if I log off from the system.

Vmware considerations

A word about running Vmware. I highly recommend to use "host-only" and not "bridged" networking in your virtual machine. "Bridged" networking assigns a second IP address to the Ethernet interface. Your virtual machine becomes a client on your network in its own right, and is equally vulnerable and forces you to secure yet another machine. It also has the effect of switching the Ethernet interface to promiscuous mode, which is almost always a bad thing, because finding an interface in that mode is usually a sign of a packet sniffer at work. If your interface is always in promiscuous mode, you will not find this suspicious. Running an internal private network is preferable.

Setting passwords in the BIOS setup and /etc/lilo.conf

Most PC systems allow you to enter the BIOS setup, usually by holding down the F2 or Delete key after powering on. This will give you access to low-level parameters set on the motherboard. For our purposes here, we will concentrate on the security-related ones. You can usually set passwords for different levels of access. You can require a password to actually boot a system, and a different one to access the BIOS setup page again.

In any case you should protect the BIOS setup itself with a password. If you don't, anyone with physical access to your machine could lock *you* out of your own machine, by just rebooting, accessing the BIOS setup, and establishing passwords. (This seems to be a popular prank that colleagues at work like to play).

If you also set a boot password, you will prevent others (for example, your children) from booting the machine, but be aware that the unattended machine will not reboot automatically after a power failure or a crash. If you want the machine to reboot after such an event, you should not set such a password.

Keep in mind, though, that while these passwords slow down an intruder who gains physical access to the machine, it will not protect your data from a determined adversary. Many motherboards lose their setups if the battery is removed for a few hours, and in any case, the disk can be installed in another system and be read there. But it will be enough to keep your cleaning woman and your baby-sitter from booting and accessing your system.

Also consider tuning the list of boot devices, so that the system cannot be booted from a floppy or a CD.

An additional option is to arm lilo with a password, which must be entered before lilo boots the system. This is really only effective if you disable the floppy and CD-ROM as boot devices in the BIOS. Also, since the password is sitting in clear text in the /etc/lilo.conf file, you must prevent others from seeing the file, or you must store it on a floppy and remove it altogether after you ran lilo.

You have several options here, which you should choose carefully. You can set a global password in the "preamble" of the lilo.conf file,

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
linear
default=linux
password=SecRet
```

You can add the "restricted" keyword to allow a standard boot without password, but require a password if the user enters boot parameters, for example, to boot into single user mode. If you decide that you want an unattended reboot after a power failure, you should consider this option.

Finally, you can set different passwords for different images, or decide to set only passwords for some images and not for others, letting, for example, your children boot Windows, but not Linux. Refer to the lilo.conf man or info pages for a complete description (type "man lilo.conf" or "info lilo.conf".

Testing the System

After the system has been set up, it is time to test it thoroughly. We will perform the following checks:

- We will demonstrate that the "immutable" attribute for a file is effective;
- we will change a file that is under tripwire surveillance and verify that the tripwire cron job alerts us to this change properly;
- we will port-scan the system from the inside and from the Internet;
- test that I can login from an authorized host (and get the login warning);
- verify that I *cannot* login from an unauthorized host, and that logcheck informs me of such an event;
- make sure that the web server is not accessible from outside the LAN;
- check that the suid bit is not honored and special device files are not recognized on the file systems which are mounted with the nosuid and nodev options;
- verify that no executables can be run from a floppy (test the noexec option);

- verify that the "vgetty" answering machine software does not permit dial-in logins.

Testing the "immutable" attribute

This is just a simple test to show that this attribute works as advertised – this attribute does not seem to be widely known, so I decided to demonstrate its effectiveness. I just show the commands I gave (as root) in sequence; I ended each input to the file with ctrl-D.

```
[ root]# cat > testfile
hello
[ root]# cat >> testfile
line 2
[ root]# cat testfile
hello
line 2
[ root]# chattr +i testfile
[ root]# lsattr testfile
---i--------- testfile
[ root]# cat >> testfile
bash: testfile: Permission denied
[ root]# chmod 777 testfile
chmod: changing permissions of `testfile': Operation not permitted
[ root]# chattr -i testfile
[ root]# chmod 777 testfile
[ root]# cat >> testfile
line 3
[ root]# cat testfile
hello
line 2
line 3
[ root]#
```

This shows that the file cannot be changed in any way while the immutable attribute is set; not even its permissions can be changed. This is a very effective protection against changes of the file.

Verify that tripwire reports changed files properly

For this test, we change a file which is "watched" by tripwire. In the /etc/hosts.deny file, we change the line

```
ALL:ALL
```

to

```
ALL : ALL
```

in order to trip off the tripwire check. We don't do anything else; we just let the daily cron job run the tripwire check and see if this error is reported. Here is the mail, with a few empty lines and some irrelevant lines removed

```
/etc/cron.daily/tmp0watch:
****   Error: Tripwire report differences  ****
Thu Mar 14 04:14:41 EST 2002
8c8
```

```
< Wrote report file: /var/lib/tripwire/report/home-20020122-002159.twr
---
> Wrote report file: /var/lib/tripwire/report/home-20020314-040924.twr
14c14
< Report created on:          Tue 22 Jan 2002 12:21:59 AM EST
---
> Report created on:          Thu Mar 14 04:09:24 2002
27c27
< Command line used:          tripwire --check
---
> Command line used:          /usr/sbin/tripwire --check
61c61
< * Critical configuration files   100            1      0      2
---
> * Critical configuration files   100            0      0      1
68,69c68,69
< Total objects scanned:  31202
< Total violations found:  3
---
> Total objects scanned:  31332
> Total violations found:  1
80c80
< Rule Name: Critical configuration files (/etc/cron.daily)
---
> Rule Name: Critical configuration files (/etc/hosts.deny)
84,86d83
< Added:
---
> "/etc/hosts.deny"
```

Port-scan the system from the inside and from the Internet

For the first tests, the machine is still not connected to the network, although the `eth0` interface has been brought up, and the Vmware internal network is running.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:60:97:B7:0D:C1
          inet addr:192.168.12.1  Bcast:192.168.12.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:371 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes: (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:10 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2336 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2336 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:395957 (386.6 Kb)  TX bytes:395957 (386.6 Kb)
```

```
vmnet1    Link encap:Ethernet  HWaddr 00:50:56:01:00:00
          inet addr:192.168.22.1  Bcast:192.168.22.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:369 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

This is a scan from the machine of its own interface, which shows the ssh, the Web server, and X11 ports:

```
# nmap 192.168.12.1
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on home (192.168.12.1):
(The 1538 ports scanned but not shown below are in state: closed)
Port       State      Service
22/tcp     open       ssh
80/tcp     open       http
443/tcp    open       https
6000/tcp   open       X11
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

And a scan of the loopback interface (127.0.0.1):

```
# nmap 127.0.0.1
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1538 ports scanned but not shown below are in state: closed)
Port       State      Service
22/tcp     open       ssh
80/tcp     open       http
443/tcp    open       https
6000/tcp   open       X11
Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

A scan of the Vmware interface:

```
# nmap 192.168.22.1
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.22.1):
(The 1537 ports scanned but not shown below are in state: closed)
Port       State      Service
22/tcp     open       ssh
80/tcp     open       http
139/tcp    open       netbios-ssn
443/tcp    open       https
6000/tcp   open       X11
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

This shows the Samba server port 139, which allows the virtual machine to access the hosts file systems.

At this point I connected the system to the Internet, and port-scanned from "the outside". This is what an attacker would see:

```
# nmap -O 1.2.3.11
Starting nmap V. 2.54BETA30 ( 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
Interesting ports on martins.machine.home.net (1.2.3.9):
(The 1547 ports scanned but not shown below are in state: closed)
Port       State      Service
22/tcp     filtered    ssh
80/tcp     filtered    http

No exact OS matches for host (test conditions non-ideal).
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i586-pc-linux-gnu%D=1/15%Time=3C448D3C%O=-1%C=1)
T5(Resp=Y%DF=N%W=800%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=800%ACK=S%Flags=AR%Ops=)
T7(Resp=Y%DF=N%W=800%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
Nmap run completed -- 1 IP address (1 host up) scanned in 15 seconds
```

That means that the configuration was successful. None of the LAN-internal ports are visible on the Internet. The fact that port 80 shows up here as "filtered" has nothing to do with my home setup. The HTTP port is filtered upstream by my cable provider as a way to enforce compliance with an agreement in the service contract not to run Web servers.

If you have no machine from which you can run an "external" port-scan, there are several Web-based services available which can perform such a scan of your IP address. It is a judgement call if you want to do this. You give away possible vulnerabilities of your system, and practically advertise your system to the web site operator, so make sure you don't ask a hacker site for a scan. An example of a scan service is http://www.dslreports.com/scan, which is given as a reference here. Take your own decision.

Logging in from an authorized host

The next sequence shows me logging in from my machine at work, which is authorized to open a ssh session. Apart from testing that it works as advertised, it shows that I get the login warning as it should be. The "homeip.txt" file holds my home system's current IP address.

```
[purschke@work ~]$ ssh `cat homeip.txt `

This Computer System is for authorized use only. Any or all uses of
this system and all files on this system may be intercepted,
monitored, recorded, copied, audited, inspected, and disclosed to law
enforcement personnel. By using this system, you consent to such
interception, monitoring, recording, copying, auditing, inspection,
and disclosure.

By continuing to use this system you indicate your awareness of and
consent to these terms and conditions of use. DO NOT LOGON if you are
```

```
not an authorized user.

purschke@1.2.3.11's password:
Last login: Wed Mar 13 20:56:28 2002 from 1.2.3.6
[purschke@home ~]$
[purschke@home ~]$ exit
logout
Connection to 1.2.3.11 closed.
```

That worked as it should. Now we test, again from the authorized host, that logcheck notifies me if I mistype the password (or someone else tries to log in from my unattended workstation). Here I deliberately give the wrong password:

```
[purschke@work ~]$ ssh `cat homeip.txt `

This Computer System is for authorized use only. Any or all uses of
this system and all files on this system may be intercepted,
monitored, recorded, copied, audited, inspected, and disclosed to law
enforcement personnel. By using this system, you consent to such
interception, monitoring, recording, copying, auditing, inspection,
and disclosure.

By continuing to use this system you indicate your awareness of and
consent to these terms and conditions of use. DO NOT LOGON if you are
not an authorized user.

purschke@1.2.3.11's password:
Permission denied, please try again.
purschke@1.2.3.11's password:
Permission denied, please try again.
purschke@1.2.3.11's password:
Permission denied (publickey,password,keyboard-interactive,hostbased).
```

Not unexpectedly, the login attempt fails. What's more interesting is whether or not logcheck picks up on that. Promptly, at the full hour, the logcheck cron job runs, and sends me the following mail:

```
Security Violations
=-=-=-=-=-=-=-=-=-=-
Mar 13 20:57:35 home sshd(pam_unix)[23599]: authentication failure; logname= uid=0
euid=0 tty=NODEVssh ruser= rhost=1.2.3.6  user=purschke
Mar 13 20:57:37 home sshd[23599]: Failed password for purschke from 1.2.3.6 port
51751 ssh2
```

Trying to login from an unauthorized host

I then log on to another host at work, which is *not* authorized to access my home system, and try to login from there:

```
purschke@anotherhost: ~> ssh 1.2.3.11
```

Nothing happens. Since the firewall rule is "deny", the ssh client doesn't receive any feedback, and times out eventually. We now have to wait for the next full hour for the cron job to run again, and

to see if we get a report. A while later, it has arrived:

```
Security Violations
=-=-=-=-=-=-=-=-=-=
Mar 13 21:31:59 home kernel: Packet log: input DENY eth0 PROTO=6 1.2.3.20:4963
192.168.12.1:22 L=60 S=0x00 I=43696 F=0x4000 T=49 SYN (#11)
Mar 13 21:32:02 home kernel: Packet log: input DENY eth0 PROTO=6 1.2.3.20:4963
192.168.12.1:22 L=60 S=0x00 I=43711 F=0x4000 T=49 SYN (#11)
Mar 13 21:32:08 home kernel: Packet log: input DENY eth0 PROTO=6 1.2.3.20:4963
192.168.12.1:22 L=60 S=0x00 I=43713 F=0x4000 T=49 SYN (#11)
```

What does this say? We denied incoming packets on interface eth0. The protocol was 6 (you can look it up in /etc/protocols, 6 is TCP, the other frequently seen UDP protocol is number 17). The packet originated from my unauthorized host at work, denoted as 1.2.3.20 on port 4963, connecting to port 22 on my 192.168.12.1 machine. (Except perhaps for the very last field, that's really all we wanted to know.)

Let's briefly go through the other fields: The length of the packet was 60 bytes. The "S" denotes the "Type of Service", which was 0 (this field can contain 4 bits, none of which is set here, which can give routers hints how to treat this packet. The bits mean "Minimum delay", "Maximum throughput", "Maximum reliability", and "Minimum cost", and obviously, at most one bit at a time is allowed, because the goals are mutually exclusive). The "I" denotes a counter at the sending host, which increases with each packet it sends. The "F" value holds the "flags" in the 3 most significant bits, and the "fragment offset" in the other bits. The flags are 0,1,0 where the "1" means that this packet must not be fragmented, and the right 0 means that this is the last (and in this case, the only) fragment of the packet – hence the fragment offset is 0. The "T" field is the "time to live" of that packet; it will be forwarded only so often, and has 49 more stations, or hops, to go. It has arrived at its destination with a comfortable margin. Each forwarding station decrements the value until the packet either reaches its destination or is discarded when the counter reaches 0. The packet had the SYN bit on. The final field "(#11)" is potentially more interesting again, because it denotes the rule in ipchains which discarded the packet. List the rules with "ipchains –L", and count them. Rule number 11 in my setup is

```
DENY      tcp  -y--l-  anywhere          anywhere          any ->   any
```

If you go back to the firewall configuration file, this is the line that became rule number 11:

```
# but reject all other incoming connection from outside
-A input -p tcp -l  -s 0/0 -d 0/0  -y -j DENY
```

Verify that the web server is not accessible from the Internet

We already saw in the port scans that port 80 is "filtered". Because it is easier to show, I use the wget utility here, which will access a server and download a file. In this case, we try to get hold of the "index.html" file. First I show a positive test to verify that this test gives the right answer when used from the LAN. This is from my laptop on the home network:

```
$ wget http://192.168.12.1
```

```
--21:43:41--  http://192.168.12.1/
         => `index.html'
Connecting to 192.168.12.1:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
    0K                                                     @ 247.72 KB/s
21:43:41 (82.57 KB/s) - `index.html' saved [761]
```

Now I logon to a machine outside, and first make sure that wget works on that machine and gives a positive test if I access a regular open Web server:

```
[lan]$ wget http://www.netscape.com
--03:54:26--  http://www.netscape.com/
         => `index.html'
Connecting to wwwproxy.somehost.de:80... connected!
Proxy request sent, awaiting response... 200 OK
Length: unspecified [text/html]

    0K ......... ......... ......... ......... .......... @  25.87 KB/s
   50K ......... ......... ....                           @ 134.32 KB/s

03:54:30 (35.39 KB/s) - `index.html' saved [76783]
```

That demonstrates that we will indeed get an answer if we access a responsive Web server. Now, finally, the attempt to connect to my home web server from that machine:

```
[lan]$ wget http://1.2.3.11
--03:58:06--  http://1.2.3.11/
         => `index.html'
Connecting to wwwproxy.somehost.de:80... connected!
Proxy request sent, awaiting response...
Proxy request sent, awaiting response... 504 Gateway Time-out
04:00:07 ERROR 504: Gateway Time-out.
```

This shows that the web server is indeed only accessible from the LAN.

Check the nodev and nosuid mount options

I specified the nodev and nosuid options on the mount options for the two "user" file systems /data and /home, and the two CD-ROM's in my system. We need to check now that the suid bit is not honored and special device files are indeed not recognized. For this test, I create a directory on the /data file system, and copy the device special files for the CD-ROM's (hdc and scd0) there, as well as a shell with the suid bit set:

```
 # ls -l
total 512
brwxrwxrwx   1 root    disk     22,   0 Aug 30  2001 hdc
brwxrwxrwx   1 root    disk     11,   0 Aug 30  2001 scd0
-rwsr-sr-x   1 root    root      519964 Mar 13 22:13 sh
```

I then cd, as myself, to that directory and see if I can use the special files. First I show the behavior

with the “regular” device special file in the /dev directory:

```
$ eject /dev/scd0
$ eject -t /dev/scd0
$
```

(CD tray opens and closes). Now I try to do the same using the block special file on the /data file system:

```
$ eject /data/purschke/stest/scd0
eject: unable to find or open device for: `/data/purschke/stest/scd0'
$
```

Next I run the shell with the suid bit set. I try to create a file in the /etc/ directory. With the suid bit honored, I would be able to create that file. Here is what happens:

```
$ /data/purschke/stest/sh
sh-2.05$ touch /etc/xxxxx
touch: creating `/etc/xxxxx': Permission denied
sh-2.05$
```

This shows that the mount options are effective. I don’t show it here, but I also tested the same with a CD mounted, and also the /home file system.

Testing the “noexec” mount option

Because binaries are not normally kept on floppy disks these days, I added the noexec option to the floppy mount options. This should prevent any executable from being run from the file system on the floppy. I copied /bin/sh to a floppy. Here is what I get:

```
$ mount /mnt/floppy/
$ mount | grep floppy
/dev/fd0 on /mnt/floppy type vfat (rw,noexec,nosuid,nodev,user=purschke)
$ ls -l /mnt/floppy
total 508
-rwxr-xr-x    1 purschke users    519964 Mar 13 22:39 sh
$ /mnt/floppy/sh
/mnt/floppy/sh: Permission denied.
$
```

This shows that the “noexec” option works as expected.

Verify that the “vgetty” answering machine software does not permit dial-in logins.

I used the minicom terminal emulator to dial my home from my office and see if I am able to login. I am not; I get disconnected after a short moment:

```
Welcome to minicom 1.83.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Aug 28 2001, 15:09:33.
```

```
Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
OK
at dt 9 555 5555
CONNECT 300 NoEC
CONNECTION LOST
NO CARRIER
```

I also tried from a Windows HyperTerminal, with the same results. As soon as the vgetty recognizes a dial-in attempt other than a fax transmission or an actual voice call, it hangs up.

This failed intrusion leaves an entry in the syslog (I made several attempts), which logcheck reports later:

```
Security Violations
=-=-=-=-=-=-=-=-=-=
Mar 15 18:23:38 home vgetty[18309]: failed A_FAIL dev=modem, pid=18309,
caller='none', conn='', name=''
Mar 15 18:39:15 home vgetty[18535]: failed A_FAIL dev=modem, pid=18535,
caller='none', conn='', name=''
Mar 15 18:45:24 home vgetty[18563]: failed A_FAIL dev=modem, pid=18563,
caller='none', conn='', name=''
Mar 15 18:51:47 home vgetty[18567]: failed A_FAIL dev=modem, pid=18567,
caller='none', conn='', name=''
Mar 15 18:53:45 home vgetty[18574]: failed A_FAIL dev=modem, pid=18574,
caller='none', conn='', name=''
```

Maintaining your system

Now that you have performed all the steps to secure your system, you should think about the maintenance work that you need to invest in order to keep your system alive and secure. You should think about the following points:

- how to perform backups
- kernel updates
- a strategy how to install security upgrades, bug fixes, and updated packages
- how to get notified of patches, security alerts, and new developments

Backups

Performing backups can be a boring and also potentially complicated task. I personally have suffered 3 catastrophic disks failures during my professional career, and the saying "not *if*, but *when* your disk fails" is still valid. Disk sizes of tens of Gigabytes have outgrown traditional (and generally too expensive) tape backup devices, and the capacity of CD-RW media as well. Those are often too slow to be routine backup media even for modest amounts of data. They can, however, be very useful to back up crucial data which do not change very often (I use it to back up the family photo album regularly, which grows, but old data don't change). With disk prices steadily falling, I decided to invest $180 for a second large backup disk. On top of the weekly "small" backups of the

photo album and the modified files in my home directory to CD-RW, I make a full backup of all disk partitions to the backup disk once a month, on the first Saturday night of a given month. I bring home and install the disk only for the backup and otherwise store it at work, so an accident (say, a fire, spilled coffee, or lightning) will not destroy the disk in use *and* the backup. If you are comfortable with opening your PC and temporarily hook up a disk, this is a very convenient way to make backups.

Some years ago my attitude was that the system comes, after all, from some installation CD, so how hard can it be to restore the system from the installation media, and backup only the "user" data? I was wrong. It took weeks until I had the system back exactly where it had been when the disk crashed. There are always some customizations that you have not recorded, some tweaks that you forgot about, some data (unfortunately in my case, a key to a commercial piece of software among them, which took a long time to replace) which are lost forever. Backing up *everything* will get your system back up in the exact way it had been in much shorter time, for not very much money.

One problem you will encounter sooner or later is that with all your mp3's, digital video, and other large multimedia files, backup files are *big*. Usually, backups are performed with the tar ("tape archiver") program, which, despite its name, is mostly used to produce disk archives of a directory or a whole file system. It has a convenient "z" flag to automatically produce a gzip-compressed tar file, and allows using other compression programs such as bzip2 as well (with the "`--use-compress-program bzip2`" option). For example, in order to backup my /home/purschke directory to a file /data/backup/purschke.tar.gz, I would use

```
$ cd /home/purschke
$ tar cfz /data/backup/purschke.tar.gz .
```

This will backup up my whole directory, together with all subdirectories, to the file. "c" means "create an archive", "f" means "not on tape, but on a file", "z" means "gzip-compress the archive". Then comes the archive filename, and then the directory to be archived. It's convenient to cd there and start the archive from a relative path (the dot "."), because you can later "displace" the whole tree when you restore it. If you have the absolute directory path in the archive, it can be more complicated.

This will work only if the size of the resulting archive, after compression, is below the file size limit of 2GB, which is often not the case (some file systems, such as the ReiserFS, can handle larger files, but in general we should stick to the 2GB limit). There is no easy way to tell whether or not an archive will stay below this limit, because you can't tell beforehand how much compression you will achieve. If the size of all your files is below 2GB, it will work. If it's slightly more than that, say, 2.5 GB, you take a chance. If it is much more than that, the archive will be too large.

In order to find out how much data you have in a directory, total, use the "du –s -m" command, which will tell you the total in units of Megabytes:

```
[purschke@home ~]$ du -s -m /home/purschke
5672   /home/purschke
[purschke@home ~]$
```

This is clearly too much for just one archive file. We have two options now. We can archive individual subdirectories, which is really error-prone and tedious. Or we can ask tar to send its output to the standard output, so we can run it through gzip and split and produce chunks of smaller files. Tar has capabilities to produce a multi-volume archive, but this has several problems, which we should better avoid. I'm showing here a script that will produce a "fragmented" archive, backup.sh:

```
#! /bin/sh
BACKUPDIR=$1
BACKUPNAME=$2
PWD=`pwd`
cd $BACKUPDIR
tar --create  --one-file-system  . | gzip -3c | split -b 2000m - ${BACKUPNAME}.
cd $PWD
```

Instead of using the "short" options of tar such as "cf", I used the "long" options, which are easier to read. We give two parameters to the script, the directory to archive, and the name of the archive. The gzip -3c instructs gzip to use compression level 3 and to run in "filter" mode, reading from standard in and writing to standard out. The dash (-) in the split command instructs split to read from standard input again. Also note the trailing dot in the filename, which will create filenames like archive.tar.gz.aa, archive.tar.gz.ab, and so on.

Instead of the command before

```
$ cd /home/purschke
$ tar cfz /data/backup/purschke.tar.gz .
```

I would now give

```
$ backup.sh /home/purschke /data/backup/purschke.tar.gz
```

The –one-file-system option in the script specifies to remain in one file system, so if I backup the root directory "/", tar will not traverse other file systems mounted under "/", such as /usr. Also note that this produces a relative-path archive as described before. The size has been set to 2000MB, which is below the limit of 2GB (which is 2048MB).

For the following demonstration, I have reduced the fragment size to 100MB, and picked a (smaller) subdirectory /home/purschke/mp3, which holds 2592 MB of data. Here is the command:

```
# du -s -m  /home/purschke/mp3
2592   /home/purschke/mp3
# ./backup.sh /home/purschke/mp3 /backup/b1/mp3.tar.gz
# ls -l /backup/b1/
total 1803880
-rw-r--r--   1 root    root    104857600 Mar 16 21:15 mp3.tar.gz.aa
-rw-r--r--   1 root    root    104857600 Mar 16 21:17 mp3.tar.gz.ab
-rw-r--r--   1 root    root    104857600 Mar 16 21:19 mp3.tar.gz.ac
-rw-r--r--   1 root    root    104857600 Mar 16 21:21 mp3.tar.gz.ad
-rw-r--r--   1 root    root    104857600 Mar 16 21:23 mp3.tar.gz.ae
-rw-r--r--   1 root    root    104857600 Mar 16 21:25 mp3.tar.gz.af
-rw-r--r--   1 root    root    104857600 Mar 16 21:27 mp3.tar.gz.ag
-rw-r--r--   1 root    root    104857600 Mar 16 21:29 mp3.tar.gz.ah
```

```
-rw-r--r--   1 root     root     104857600 Mar 16 21:31 mp3.tar.gz.ai
-rw-r--r--   1 root     root     104857600 Mar 16 21:33 mp3.tar.gz.aj
-rw-r--r--   1 root     root     104857600 Mar 16 21:34 mp3.tar.gz.ak
-rw-r--r--   1 root     root     104857600 Mar 16 21:36 mp3.tar.gz.al
-rw-r--r--   1 root     root     104857600 Mar 16 21:38 mp3.tar.gz.am
-rw-r--r--   1 root     root     104857600 Mar 16 21:40 mp3.tar.gz.an
-rw-r--r--   1 root     root     104857600 Mar 16 21:42 mp3.tar.gz.ao
-rw-r--r--   1 root     root     104857600 Mar 16 21:44 mp3.tar.gz.ap
-rw-r--r--   1 root     root     104857600 Mar 16 21:47 mp3.tar.gz.aq
-rw-r--r--   1 root     root      62717952 Mar 16 21:48 mp3.tar.gz.ar
#
```

Restoring data from the fragmented archive is painless. We use gunzip –c to send the uncompressed fragment files to its standard output and pipe the output into tar:

```
# cd /home/purschke/mp3
# gunzip -c /backup/b1/mp3.tar.gz.* | tar --extract --same-owner --same-permissions
```

Since we usually perform the backups as root, these options will restore the ownership and the permissions of the files properly.

Kernel updates

Every so often, a new stable kernel is released, and you will think about a kernel upgrade. Such an upgrade can be very disruptive if you get it wrong, and there may be third-party packages (such as VmWare) which build kernel modules that have to be re-made as well. I usually skim the changelog files for the kernel (go to http://www.kernel.org, and you see the latest stable version and a link to the changelog), and decide if there is anything I use and think is important. In this case, with the new kernel 2.4.18, there was nothing obvious that would affect my home machine, but there were several bug fixes for the "eepro100" Ethernet card driver [24]. My laptop uses this driver, so I upgraded, and I like to keep all my systems at the same level.

Download the kernel source from http://www.kernel.org, and unpack the source tree. To do that, cd to `/usr/src`. You will see some kernel directories, and a "linux" symlink to the current one (in my case, 2.4.16 before the upgrade):

```
# ls -l
total 12
lrwxrwxrwx   1 root     root          12 Dec  4 20:47 linux -> linux-2.4.16
drwxr-xr-x  14 573      573         4096 Jan 23 20:18 linux-2.4.16
drwxr-xr-x  14 1046     101         4096 Oct 16 20:06 linux-2.4.9
drwxr-xr-x   7 root     root        4096 Sep  6  2001 redhat
```

The file names in the tar archive of the new kernel start with a linux/ directory (see the discussion of relative-path archives in the previous chapter), so you will need to remove your linux symlink temporarily. After unpacking the archive, you end up with an actual "linux" directory:

```
# rm linux
rm: remove `linux'? y
# tar xfz ~/linux-2.4.18.tar.gz
# ls -l
```

```
total 16
drwxr-xr-x   14 573     573         4096 Feb 25 14:37 linux
drwxr-xr-x   14 573     573         4096 Jan 23 20:18 linux-2.4.16
drwxr-xr-x   14 1046    101         4096 Oct 16 20:06 linux-2.4.9
drwxr-xr-x    7 root    root        4096 Sep  6  2001 redhat
```

You should then rename the new directory to linux-2.4.18, and re-establish the symlink, which now points to the new kernel source tree:

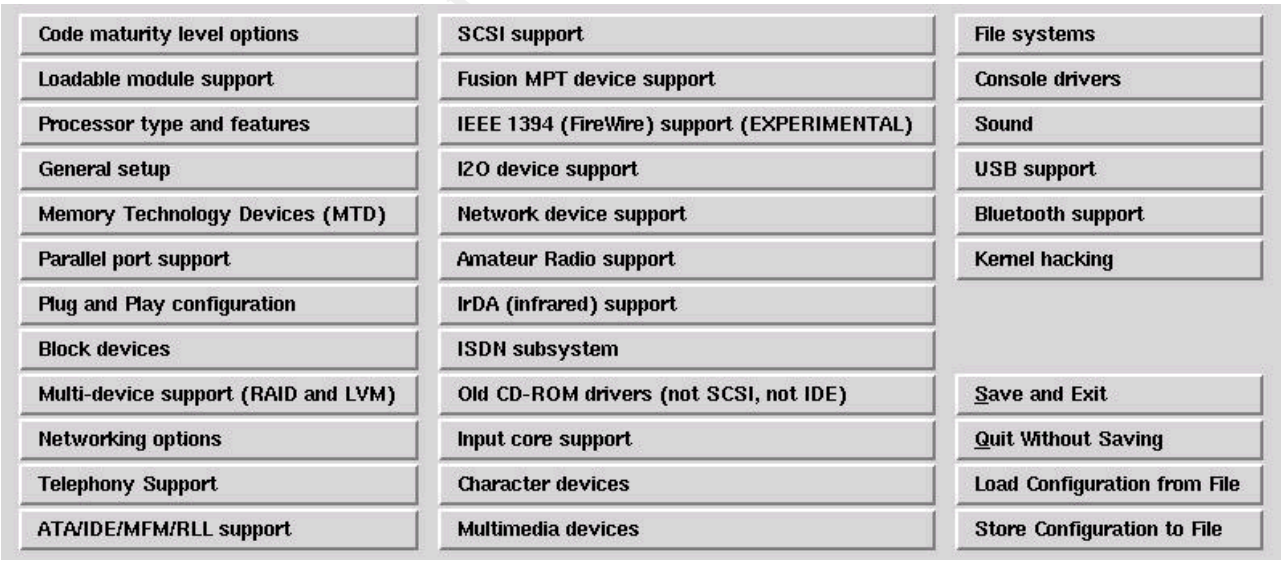
```
# mv linux linux-2.4.18
# ln -s linux-2.4.18 linux
# ls -l
total 16
lrwxrwxrwx    1 root    root          12 Mar 14 22:14 linux -> linux-2.4.18
drwxr-xr-x   14 573     573         4096 Jan 23 20:18 linux-2.4.16
drwxr-xr-x   14 573     573         4096 Feb 25 14:37 linux-2.4.18
drwxr-xr-x   14 1046    101         4096 Oct 16 20:06 linux-2.4.9
drwxr-xr-x    7 root    root        4096 Sep  6  2001 redhat
```

This allows you to easily switch between kernel versions, if needed.

The existing kernel configuration is kept in a hidden file .config in the old linux-2.4.16 directory. We change directories now to the new linux/ area. By copying the old configuration file to the new kernel directory, we do not need to go through the configuration procedure from scratch:

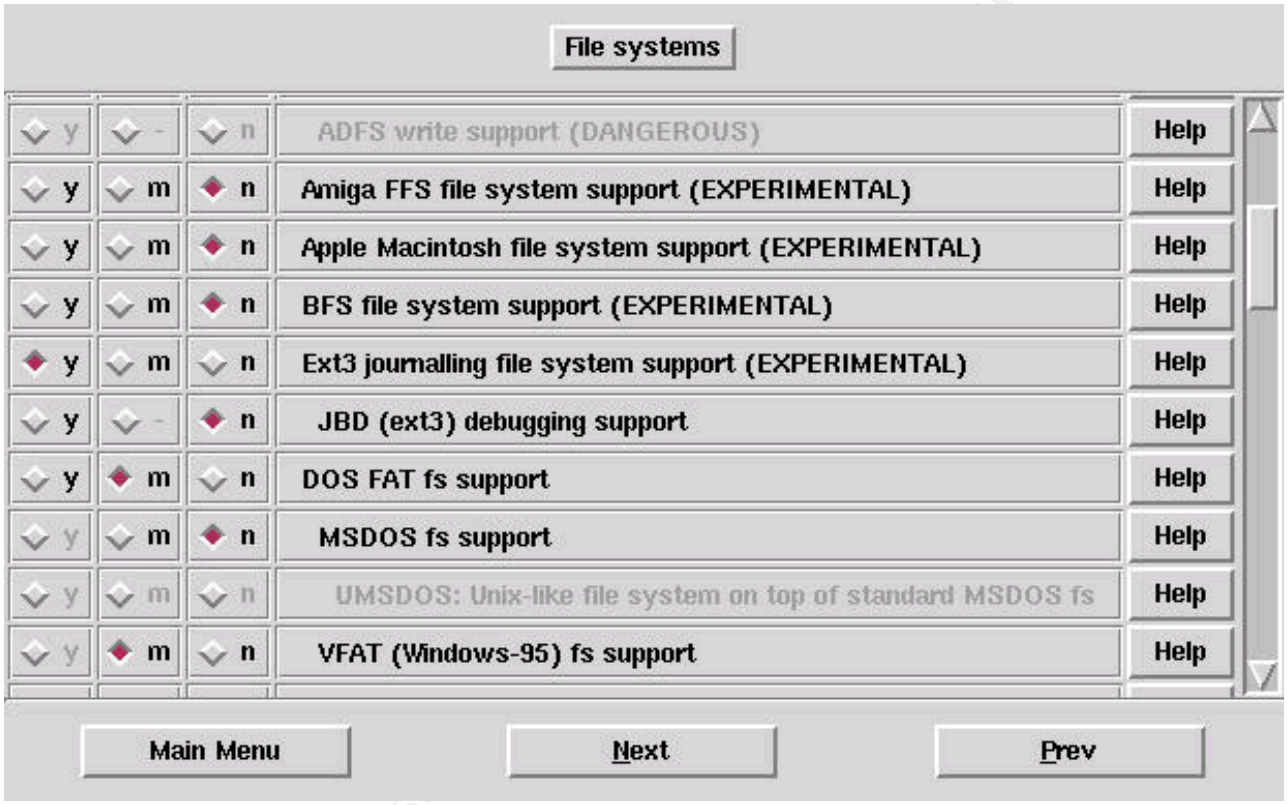
```
# cd linux/
# cp ../linux-2.4.16/.config .
```

This just puts the configuration file in place. In order to actually configure the kernel and prepare it for compilation, we still need to run “make xconfig” (which I will do next) or “make config” for a text-only version. Here is a screen shot of the graphical configuration utility:

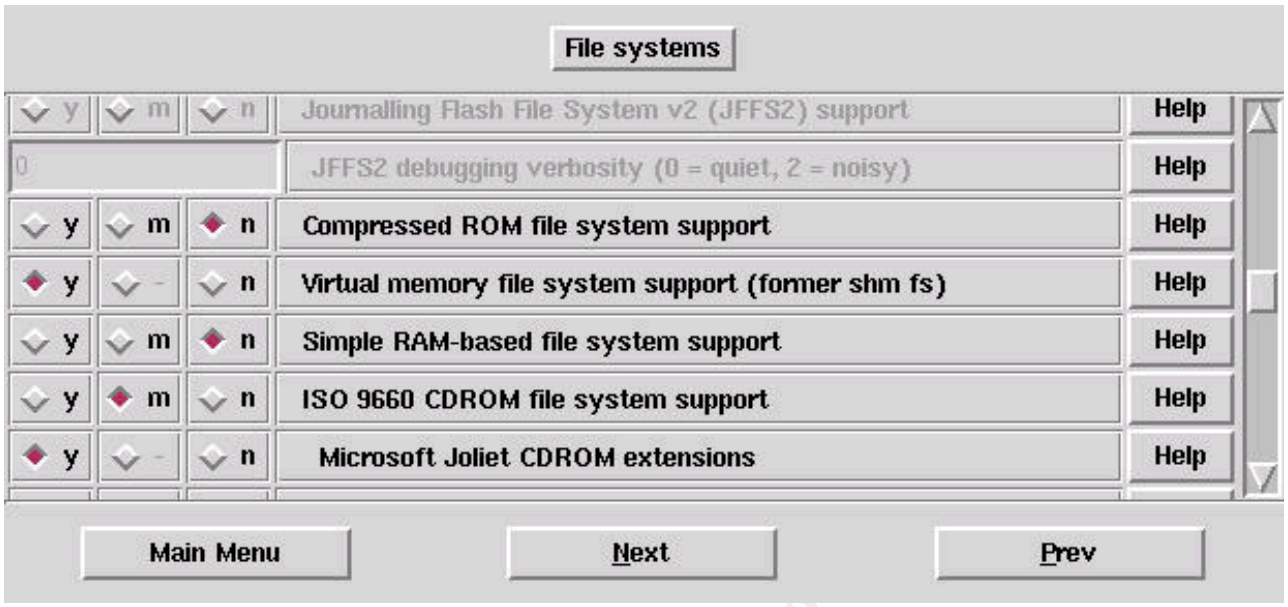


This allows you to tweak your kernel the way you like it, which I did when I set up the original customized kernel. Since we re-use the configuration from the old 2.4.16 kernel, most settings

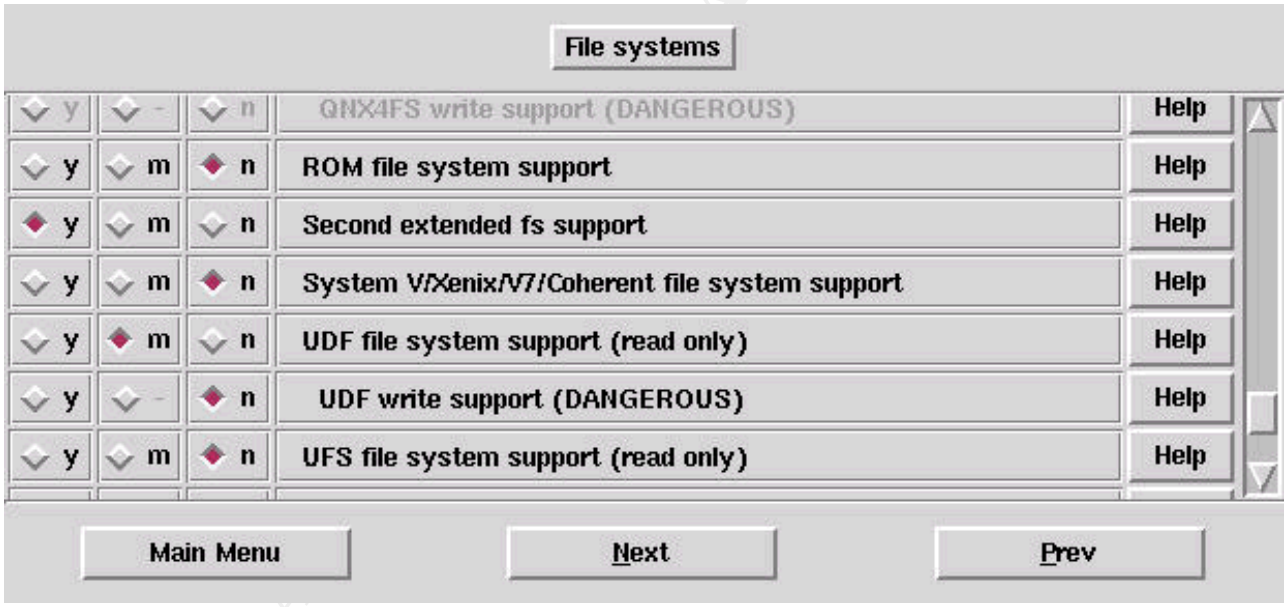
should be the same as before. The "y" options mean "yes, we want to add that driver to the kernel", while "m" means "we want to make that driver a module", which can be loaded later and on demand only. For good measure, we will look at some of the most important settings (we don't really expect them to be wrong, just checking). First, we need to make sure that the kernel can read the disks. That means that the kernel must have support for the "ext3" file system, and it *must not* be a module; the kernel itself must be able to read the disk. (This is not strictly true, there are ways to load modules at boot time from a ramdisk, but this is beyond the scope of this discussion).



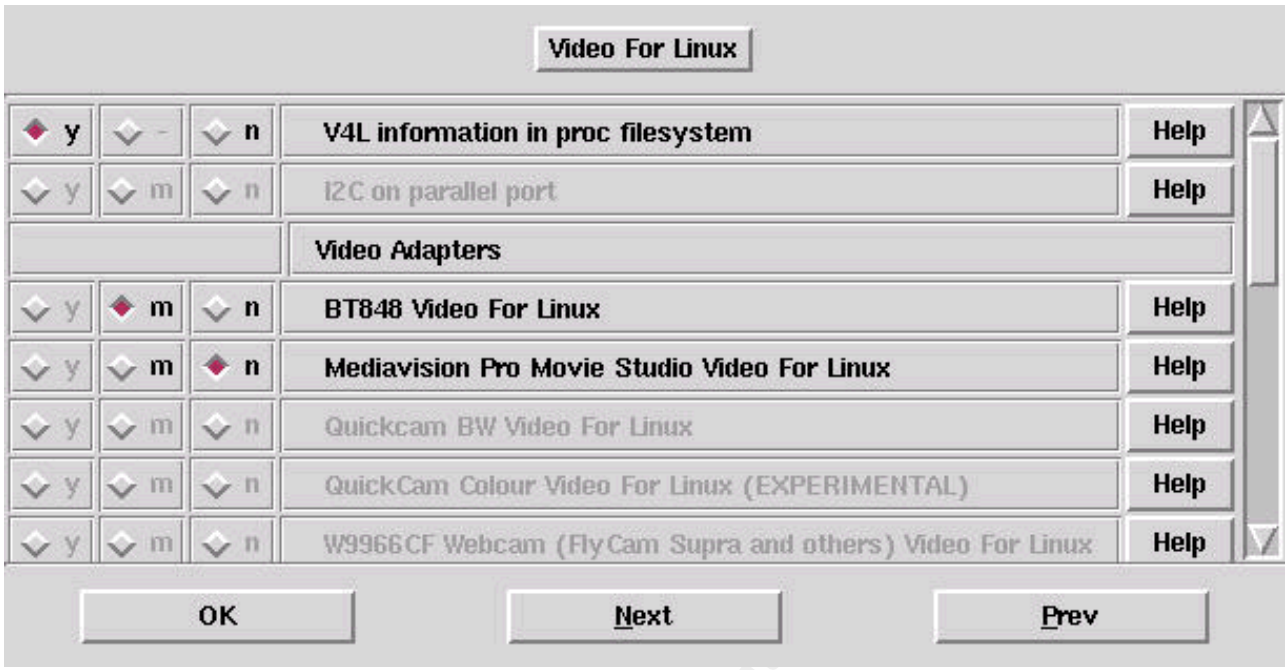
Further down in the same menu, we also check that the CDROM ISO9660 file system is supported, and that the "Joliet" extensions are enabled (we can make this a module).



Along the same lines, we also check the "ext2" file system, and also the "UDF" file system, which allows to read DVD disks.



The TV card is a "BT848" card, which I check next:



These are some crosschecks that the transfer of the configuration worked ok.

When we are done, we click on "Save and Exit" on the main panel. That sets up the kernel tree for compilation. The all-in-one command to compile everything and also install the new modules (not yet the kernel) is

```
# make dep bzImage modules modules_install
```

That is the same as running "make dep" (generate all dependencies, essentially figure out what else needs to be compiled), "make bzImage" (that builds a bzip2-compressed kernel), "make modules" (build the modules), and "make modules_install" (install the modules).

We usually want a bzip2-compiled kernel because it compresses to smaller sizes than gzip. If you want a gzip-compressed one, use "zImage" instead of "bzImage".

That compilation will take some time now, depending on how fast your machine is. When it's done, we copy the kernel to the /boot area, copy the file System.map there as well (giving it a new file name with the version):

```
# cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.18
# cp System.map /boot/System.map-2.4.18
```

Then we symlink the System.map to our version-specific System.map-2.4.18 file:

```
# cd /boot
# ln -sf System.map-2.4.18 System.map
# ls -l System.map
lrwxrwxrwx    1 root     root       17 Mar 14 22:55 System.map -> System.map-2.4.18
```

We are almost there. We now edit /etc/lilo.conf, and add the new kernel:

```
$ cat /etc/lilo.conf
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
linear
default=linux

image=/boot/vmlinuz-2.4.18
       label=linux
       read-only
       root=/dev/hda5
       append="hdd=ide-scsi"

image=/boot/vmlinuz-2.4.16
       label=linux-2.4.16
       read-only
       root=/dev/hda5
       append="hdd=ide-scsi"

image=/boot/vmlinuz-2.4.7-10
       label=orig
       initrd=/boot/initrd-2.4.7-10.img
       read-only
       root=/dev/hda5
       append="hdd=ide-scsi"

other=/dev/hda1
       optional
       label=win98
```

You can see that the old kernel 2.4.16 is still in there, and can be booted at the lilo prompt as "linux-2.4.16". This is a safety belt in case something went awry with the new kernel. After a few weeks without any problems with the new kernel, I will remove that kernel, remove the source tree, and the modules of version 2.4.16.

All that's left is to run lilo to install the new kernel with the boot loader setup:

```
# lilo
Added linux *
Added linux-2.4.16
Added orig
Added win98
#
```

Updates

With a RedHat system, you have your system backed by an actual company, which offers services to keep your system up to date automatically, and at the latest patch level. If you bought the installation CD's, you are entitled to use the services of the "Red Hat Network". If you downloaded the installation CD's from the Web, you can still purchase the service. This works best if you have a

standard RedHat installation without many customizations. The idea is that an "update agent" polls a RedHat server periodically for new updates and patches, decides if they are relevant for the system, and downloads and installs them automatically. You can customize many aspects of how this is done. I don't use the service myself, but I know people who maintain their system this way, and are very satisfied with it. You can find more information here [25].

If you are uneasy about automated updates, or don't want to pay for the service, you can still poll the RedHat Web site (http://www.redhat.com/support/errata/rh72-errata.html) yourself, and download and install the patches. If you establish a routine when you do that, say, every Friday night, you can keep track of all updates that are posted.

A perhaps better alternative is AutoRPM [26], which I use. It polls a list of ftp sites for upgrades. It is highly customizable, and can be a bit intimidating to a novice user, mainly because of the extremely terse documentation, which assumes that you are already familiar with the terminology used in the program. There is a HOWTO [27] available, which offers a few tips, but does not add much value for the beginner beyond what the man pages say.

However, you can use AutoRPM almost out of the box, and do not need to change very much. If you install AutoRPM itself as a RPM, there is essentially only one configuration file that you need to adjust. AutoRPM installs in the /etc/autorpm.d/ directory, and the configuration file is /etc/autorpm.d/pools/redhat-updates. In there, you'll find a whole list of ftp sites to poll for updates. Strip that down to just one line

```
# cat /etc/autorpm.d/pools/redhat-updates
ftp://updates.redhat.com/${RHVersion}/${Lang}/os
```

So this becomes the only site which is polled for updates.

In another file, etc/autorpm.d/redhat-updates.conf, you should find the lines

```
# Don't upgrade kernel packages...
   Regex_Ignore ("^kernel-");
```

This instructs AutoRPM to leave the kernel alone (ignore all kernel updates), which is what we usually want, as discussed before. Check that those lines are actually there.

The RPM installs a script in the /etc/cron.daily directory which runs autorpm once a night. By default, it will only download updated RPM's, that is, only newer versions of RPM's that you have installed on your system. It will notify you about the availability of *new* packages, without downloading them. By default, it will not install anything, which is usually the way you want it. You will receive mail if there are updates pending, and you can decide to install them at a convenient time. You could modify the configuration to automatically install new patches, if you prefer to have everything done on autopilot. I certainly like control over what gets installed.

If you decide to update, you run autorpm interactively and can use AutoRPM commands (list, install) to manage the new RPM's to be installed. Here is the transcript of an autorpm session, where I query which updates have been downloaded. I then decided to install the redhat-config-network update:

```
# autorpm
```

```
          -= AutoRPM Tip ('set tips off' to disable) =-

You can enable FTP hash marks while downloading files in interactive mode:
  set ftp_hash on

  136 RPM(s) waiting to be installed/updated/removed Interactively

AutoRPM@home> list update
[Update] XFree86-4.1.0-15
[Update] ghostscript-6.51-16
[Update] jadetex-3.11-4
[Update] kde-i18n-German-2.2.2-2
[Update] kdoc-2.2.2-1
[Update] koffice-1.1.1-2
[Update] perl-5.6.1-26.72.3
[Update] redhat-config-network-0.9.10-2
[Update] tetex-dvips-1.0.7-38.2
[Update] ttfonts-ja-1.0-7
AutoRPM@home> install redhat-config-network-0.9.10-2
  Installing Packages...
Preparing...              ########################################### [100%]
  1:redhat-config-network  ########################################### [100%]
  Package redhat-config-network-0.9.10-2 (Upgraded from 0.9-1)
    Deleted /var/spool/autorpm/redhat-config-network-0.9.10-2.noarch.rpm
AutoRPM@home>
```

Getting notified about Patches, Security alerts, and new Developments

In order to become aware of new threats and to be able to react in a timely manner (recently, the "zlib overflow bug" was all over the news [28]), it is important to subscribe to mailing lists or read newsgroups. This can lead to a daunting amount of information, and you can end up with more mail in your inbox than you can handle, so be careful which lists you subscribe to. For a RedHat system, you find a nice selection of mailing lists at http://www.redhat.com/mailing-lists/, and you should consider subscribing to the "redhat-announce-list" (http://www.redhat.com/mailing-lists/redhat-announce-list/index.html). You will be notified of new products, patches, and updates, and it is Redhat-specific. There are many more mailing lists if you want to get more involved. There is also a good Web site at http://www.linuxsecurity.com/advisories/index.html, which keeps you up-to-date with the latest security flaws that have been discovered.

Being Prepared: Building your own Bootable Forensic Toolkit CD

If the worst happens and your system gets compromised, the damage can be limited by having the right tools prepared in advance. I would like to introduce here a toolkit that I made, based on another project of mine from a few years back, which allows you to create custom-made bootable CD's in an easy fashion [29]. I used my project to create a bootable "forensic toolkit" CD with the tools needed to perform a thorough analysis of a potentially compromised system. You can download the CD image from my Web site if you trust my tools, or you can use the provided

scripts to build your own customized toolkit in about 30 minutes.

The way it is set up, the CD serves two purposes. When mounted as a regular CD in a running system, it has a `lib/` and a `bin/` directory with the tools and the necessary shared libraries to run them. If you put the directories in the front of the `$PATH` and `$LD_LIBRARY_PATH`, you can be sure to have "clean", tamper-proof utilities. This will defeat "trojan" programs in your system that may have been altered not to show certain information, such as certain processes. If your system has been rendered unusable or unbootable, or if you suspect that the kernel or the kernel modules have been altered, you can boot your system from the CD and have all the tools available to do the forensics, attempt a recovery, or just salvage user files. In that case, the above-mentioned `lib/` and `bin/` directories become `/usr/lib` and `/usr/bin` in that system.

Here is what you see if you burn the ready-made CD image and mount the CD:

```
# mount /mnt/cdrom
# ls /mnt/cdrom
bin  boot  boot.catalog  lib  ramdisk.tar.gz  TRANS.TBL
# ls /mnt/cdrom/bin/
awk      cp     egrep   gzip     ls     mkswap  ps       strings   vi
bunzip2  date   fdisk   ifconfig lsattr mount   rm       su        w
bzip2    dd     fgrep   init     lsmod  mv      rmdir    sync      which
cat      df     file    insmod   lsof   netstat rmmod    tar       who
chattr   diff   find    killall  ltrace nm      route    test
chgrp    dig    ftp     last     md5sum od      script   top
chmod    du     gdb     ldd      mkdir  passwd  sh       TRANS.TBL
chown    dump   grep    lilo     mke2fs pgp     shutdown tripwire
compress e2fsck gunzip  ln       mknod  portmap strace   umount
#
```

Of course your personal collection of tools may be different. I also keep the tripwire database and configuration files on my personal version of this CD, so that I can perform a tripwire check with the original tripwire database.

When we prepend our directories to the `PATH` and `LD_LIBRARY_PATH`, we see

```
# export LD_LIBRARY_PATH=/mnt/cdrom/lib
# PATH=/mnt/cdrom/bin:$PATH
# which ltrace
/mnt/cdrom/bin/ltrace
# ldd `which ltrace`
      libdl.so.2 => /mnt/cdrom/lib/libdl.so.2 (0x40018000)
      libc.so.6 => /mnt/cdrom/lib/libc.so.6 (0x4001c000)
      /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
#
```

Only the loader `/lib/ld-linux.so` is still taken from the potentially altered file system; all utilities in the above list and their shared libraries are taken from the CD.

The image of the CD can be found at

http://www.phenix.bnl.gov/~purschke/RescueCD/bootcd_1.3_forensics.iso.gz

Summary

It is very important to secure your home machines, especially when you upgrade to a broadband connection. This article describes the steps I took to harden my Redhat Linux 7.2 machine. No two setups are the same, and the threat levels may differ, but for most home systems connected to a broadband Internet service, I recommend

- get and configure a hardware firewall/router;
- configure a host-based firewall, and engage the tcp wrapper;
- upgrade to the latest versions of the software;
- shutdown unneeded services, in particular xinetd;
- modify the configuration files to achieve the desired level of security;
- lock down sensitive files with the chattr +i command;
- install and configure tripwire, and establish a cron job which generates reports regularly;
- install and configure logcheck to review your log files several times per day;
- port-scan and otherwise probe your system regularly to make sure that your defenses are up, and that you get alerted to such events.
- Set up a backup strategy.
- Set your system up for some form of automated upgrades.
- Subscribe to newsgroups of mailing lists to get notified of new developments and patches.
- Download the "forensics" CD image and burn the CD, or download the project files and make your own CD.

Appendix A

This is a list of all the installed packages in my system, organized the way you will see it in the selection GUI at install time. Keep in mind that this reflects my choice of what I find essential, but it can give you some guidance what to install.

------- Amusements/Games

Xboing	kdegames	Trojka
Xgammon	xpat2	Xpuzzles
Xjewel	fortune-mod	Gnome-games
Chromium	freeciv	gnuchess
Maelstrom	xbill	xboard
Xpilot		

------- Amusements/Graphics

Xv	kdetoys	xloadimage
Xsri	xscreensaver	xmorph

------- Applications/Archiving

cpio	tar	unzip
zip	cdrecord	rmt
sharutils	dump	

------- Applications/Communications

mgetty	efax	pilot-link
lrzsz	minicom	dip
mgetty-voice		

------- Applications/Databases

db3	utils	

------- Applications/Editors

Psgml	vim-common	vim-minimal
Gedit	abiword	emacs
emacs-nox	emacs-X11	hexedit
Xemacs		

------- Applications/Engineering

bc	Gnuplot	

------- Applications/File

File	Bzip2	fileutils
Findutils	Gzip	slocate
perl-Digest-MD5	stat	ncompress

------- Applications/Internet

Urlview	im	mutt
Kdenetwork	lokkit	htdig
Finger	ftp	gq
Htmlview	openldap-clients	rsh
Stunnel	talk	telnet
Traceroute	wget	whois
Balsa	links	netscape-common
netscape-communicator	pan	pine
Curl	tcpdump	rsync
Lynx	plugger	slrn
Openssh	openssh-askpass	openssh-askpass-gnome
openssh-clients	mozilla	mozilla-mail
mozilla-psm		

------- Applications/Multimedia

Xpaint	MPEG	multimedia
Kdegraphics	Gnome-audio	netpbm-progs
Aumix	Desktop-backgrounds	dia
Ee	Extace	ImageMagick
Awesfx	Cdda2wav	cdp
Cdparanoia	Gnome-media	mikmod
mpg321	Playmidi	sox
sndconfig	Timidity++	kdemultimedia
vorbis	Xawtv	xmms
xmms-gnome	Gimp	xsane

gimp-devel	Grip	playmidi-X11
xcdroast	Xmms-skins	

------- Applications/Productivity

gnome-pim	Gnumeric	kdepim
ical	groff	

------- Applications/Publishing

ghostscript-fonts	Groff-perl	mpage
pnm2ppa	Psutils	a2ps
ghostscript	Gphoto	sgml-tools
tetex-fonts	Tetex	tetex-afm
tetex-dvilj	Tetex-dvips	tetex-latex
jadetex	Tetex-xdvi	xpdf
texinfo	Gv	

------- Applications/Sound

bladeenc		

------- Applications/System

control-panel	Tksysv	modemtool
Autorun	Dosfstools	hdparm
Parted	Setserial	netconfig
Setuptool	Syslinux	iproute
Procps	Psmisc	console-tools
Time	Which	kudzu
Pciutils	Timeconfig	mkxauth
Usermode	hwbrowser	locale_config
Gnorpm	gtop	bug-buddy
gnome-utils	lm_sensors	kdeutils
Magicdev	mkisofs	bind-utils
Gnupg	firewall-config	nmap
Rdate	redhat-config-network	statserial
rp3	sane-frontends	sysstat
samba-common	samba-client	apacheconf
Tripwire	mt-st	rdist
ucd-snmp-utils	dialog	tamago
Arpwatch	screen	sudo
Mtools	cdrdao	linuxconf
gnome-linuxconf		

------- Applications/Text

docbook-dtd40-sgml	perl-SGMLSpm	docbook-dtd31-sgml
docbook-dtd41-sgml	docbook-dtd30-sgml	diffutils
Ed	gawk	grep
Less	sed	textutils
m4	nkf	pspell
Aspell	aspell-de	sgml-common
Openjade	docbook-style-dsssl	docbook-utils
docbook-utils-pdf	indent	dos2unix
unix2dos		

------- Development/Debuggers

Lsof	Gdb	ltrace
Memprof	Strace	

------- Development/Languages

gnome-objc	Perl	python
pygtk	Pygnome	pygtk-libglade
pygnome-libglade	Tcl	tk
tix	Tkinter	librep
rep-gtk	Rep-gtk-gnome	rep-gtk-libglade
umb-scheme	Guile	cpp
php	Php-imap	php-ldap
dev86	Gcc	gcc-g77
gcc-objc	Gcc-c++	expect
pygtk-devel	Tclx	

------- Development/Libraries

kdelibs-sound-devel	Libxml2	libxslt
gnome-objc-devel	Kdelibs-devel	libxml2-devel
lesstif	Popt	PyXML
4Suite	Python-xmlrpc	rpm-python
audiofile-devel	Bzip2-devel	cyrus-sasl-devel
db1-devel	E2fsprogs-devel	esound-devel
expat-devel	Freetype-devel	gd-devel
gdbm-devel	Glib-devel	glibc-devel
gmp-devel	Gnome-core-devel	gnome-pim-devel
gsm-devel	Krb5-devel	krbafs-devel
libao-devel	Libghttp-devel	libgtop-devel
libjpeg-devel	Libmng-devel	libogg-devel
libstdc++-devel	Libtermcap-devel	libtiff-devel
libungif-devel	Libvorbis-devel	Mesa-devel
ncurses-devel	Netpbm-devel	openldap-devel
openssl-devel	ORBit-devel	pam-devel
pciutils-devel	Kudzu-devel	pcre-devel
pilot-link-devel	Python-devel	qt-devel
rpm-devel	Slang-devel	newt-devel
Xaw3d-devel	XFree86-devel	gtk+-devel
gnome-games-devel	Zlib-devel	libpng-devel
imlib-devel	Gnome-libs-devel	control-center-devel
gdk-pixbuf-devel	Libxml-devel	libglade-devel
alchemist-devel	Db2-devel	libcap-devel
pump-devel		

------- Development/System

kernel	Headers	

------- Development/Tools

Kdevelop	Jdk	make
Autoconf	Automake	binutils
Bison	Byacc	cdecl
Ctags	Cvs	diffstat
Doxygen	Flex	gettext

Glade	libtool	njamd
Patch	pmake	qt-designer
Rcs	rpm-build	cproto
ElectricFence		

------- System/Base

Common	licenses	

------- "System Environment/Base"

libgnomeprint11	Basesystem	glibc-common
Mailcap	Redhat-logos	setup
Filesystem	Chkconfig	e2fsprogs
Eject	Losetup	mingetty
Mktemp	net-tools	pwdb
shadow-utils	Ntsysv	termcap
Crontabs	Logrotate	MAKEDEV
Info	Dhcpcd	man
Raidtools	Redhat-release	rootfiles
Kbdconfig	Dev	mount
Mkinitrd	Lilo	mkbootdisk
Mouseconfig	Tmpwatch	utempter
Pam	Authconfig	passwd
SysVinit	Rpm	util-linux
Initscripts	Ipchains	iptables
Quota	vixie-cron	anacron
Alchemist	Chkfontpath	dateconfig
libgnomeprint15	gnome-print	GConf
scrollkeeper	nss_ldap	pam_krb5
rhn_register	rhn_register-gnome	up2date
up2date-gnome	yp-tools	shapecfg
ttfm		

------- "System Environment/Daemons"

xinetd	Xinetd	ppp
bdflush	Iputils	at
procmail	Sysklogd	gpm
openldap	Sendmail	apmd
XFree86-xfs	Esound	ntp
ORBit	Portmap	fam
autofs	nfs-utils	nscd
pidentd	Rusers	rwho
sendmail-cf	tcp_wrappers	ucd-snmp
rwall-server	talk-server	telnet-server
ypserv	Samba	apache
mod_dav	mod_perl	mod_ssl
tux	gpm-devel	inews
mod_bandwidth	mod_put	mod_throttle
pump	Openssh-server	

------- "System Environment/Kernel"

devfsd	Ksymoops	hotplug
modutils	Kernel	

------- "System Environment/Libraries"

pythonlib	Kdelibs	libgal3
kdelibs-sound	Arts	SDL
SDL	gnome-audio-extra	glibc
bzip2-libs	Cracklib	db1
db2	db3	gdbm
glib	Pcre	slang
newt	Libtermcap	Canna-libs
libstdc++	Ncurses	openssl
readline	Words	cracklib-dicts
cyrus-sasl	cyrus-sasl-md5	cyrus-sasl-plain
krb5-libs	Zlib	expat
freetype	Gmp	libpng
XFree86-libs	VFlib2	gtk+
libjpeg	Libtiff	gdk-pixbuf
Wnn6-SDK	Xaw3d	Mesa
audiofile	Libcap	libmng
libungif	Libxml	netpbm
imlib	gnome-libs	libglade
qt	libtool-libs	gal
gdk-pixbuf-gnome	gtk-engines	imlib-cfgeditor
libghttp	libgal7	libgtop
libole2	Librsvg	libunicode
oaf	Bonobo	gnome-vfs
eel	Gtkhtml	gnome-core
libogg	Libvorbis	libao
smpeg	compat-libstdc++	krbafs
lockdev	Libesmtp	sane-backends
gd	Mm	w3c-libwww
ncurses4	openssl096	readline2.2.1
XFree86-compat-libs	db3-devel	db31
ImageMagick-perl		

------- "System Environment/Shells"

bash	Ash	tcsh
sh-utils	Mc	

------- System/Libraries

libaa1	libxmms1	

------- System/Servers

cups		

------- "User Interface/Desktops"

kdebase	Kdeadmin	switchdesk
gmc	Switchdesk-gnome	control-center
sawfish	gnome-applets	nautilus
nautilus-mozilla	kde-i18n-German	kdeartwork-locolor
koffice	Switchdesk-kde	vnc
sawfish-themer		

------- "User Interface/X"

urw-fonts	ttfonts-ja	XFree86
XFree86-100dpi-fonts	XFree86-75dpi-fonts	XFree86-ISO8859-15-100dpi-fonts
XFree86-ISO8859-15-75dpi-fonts	XFree86-tools	XFree86-twm
xinitrc	XFree86-xdm	gqview
gdm	Rxvt	

------- "User Interface/X Hardware Support"

Xconfigurator		

------- X11/Utilities

xfstt		

Appendix B: PGP and GPG

We have briefly seen the use of the pgp –w command that allows you to "wipe" a file so that its contents cannot be recovered. There are many more aspects to PGP that are relevant to keeping computer systems secure (as well as securing your day-today communications per email), so let me summarize some important points here.

There are two main free implementations of encryption software, the "classic" PGP, and the open-source GPG (GNU Privacy Guard). Since PGP had been contested in court because of a patent issue with the (recently expired) RSA patent, GPG was developed as a patent-free implementation of PGP's functionality, plus some enhancements [30]. The PGP source code, originally written by Phil Zimmermann [31], has been bought from Phil by a company called Network Associates. The program itself has been freely available (in binary form, not open source) for non-commercial use up to version 6.5.8. Recently, Network Associates announced that the company would discontinue the product [32], which makes it unclear what the future of the classic PGP will be. I have used PGP for many years, and Network Associates' PGP was readily available for the usually less technically inclined Windows users, which increased its acceptance. With the product discontinued, and with the problem of a binary-only distribution raising suspicions about back doors, I currently evaluate the upgrade path to GPG. As far as I can tell, GPG offers the same functionality as PGP, except for the "wipe" feature.

The basic idea behind PGP and GPG is what's called "public key encryption", which means that instead of just one key, you have a key pair which consists of a private component, and a public component, usually called public and private keys for short. You freely distribute your public key, which can be used by anyone to *encrypt* data. Only you with your private key can *decrypt* the data again. The public key is not enough to decrypt. This solves all kinds of problems associated with the key distribution in symmetric key scenarios, where the same key can encrypt as well as decrypt. (PGP and GPG support this symmetric encryption mode as well, for example, to encrypt the lilo.conf file if it contains a password).

In practice, you generate a key pair (matching public/private keys), give the public key to your friends, which can then send you encrypted email (or files in general).

I show here a session where I generated a new key pair for GPG. The corresponding command to do the same for PGP would be “pgp –kg”.

```
$ 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: Warning: using insecure memory!
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) 4096
Keysizes larger than 2048 are not suggested because
computations take REALLY long!
Are you sure that you want this keysize? y
Okay, but keep in mind that your monitor and keyboard radiation is also very
vulnerable to attacks!
Requested keysize is 4096 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)
Key does not expire at all
Is this correct (y/n)? y

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: Martin Purschke
Email address: purschke@home.net
Comment: GPG key
You selected this USER-ID:
   "Martin Purschke (GPG key) <purschke@home.net>"

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.
```

```
+++++.++++++++++++++++++++++++++++++++.+++++..+++++..++++++++++.++++++++++++
(output deleted)
```

You can upload your public key with your email address to a key server (for example, http://pgp.mit.edu), where many mail clients can automatically retrieve a key that matches a given email address. If you want to send encrypted mail to bob@internet.net, he can give you his public key directly, or you can query the key servers if a public key for that address is on file, and download it. Once you have Bob's public key, you can encrypt files for him to read, for example a file foo.txt, by typing

```
gpg -e -r bob@internet.net foo.txt
```

This creates a binary file foo.txt.gpg, which only Bob can decrypt. Because binary files are notoriously hard to send by email, there is an "ASCII armor" option that produces plain ASCII output. Use

```
gpg -ea -r bob@internet.net foo.txt
```

This time, you get a foo.txt.asc file that should be fine with any mail client.

Be aware that only Bob can now decrypt the file. If you keep a copy of the encrypted file, it's useless for you, because you lack the required secret key (Bob's) to decrypt it. If you plan to destroy the original file, be sure to add your own email address or name to the list of people who can decrypt it. I would use

```
gpg -ea -r bob@internet.net -r purschke foo.txt
```

and would now be able to decrypt it myself (there is a preferences file where you can set this "encrypt to self" option by default).

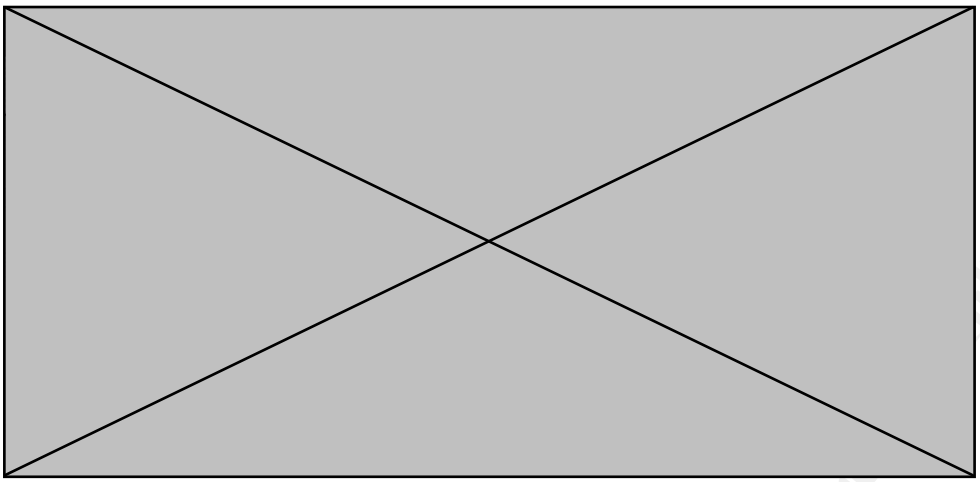
In order to read an encrypted file, you just type

```
gpg foo.txt.asc
```

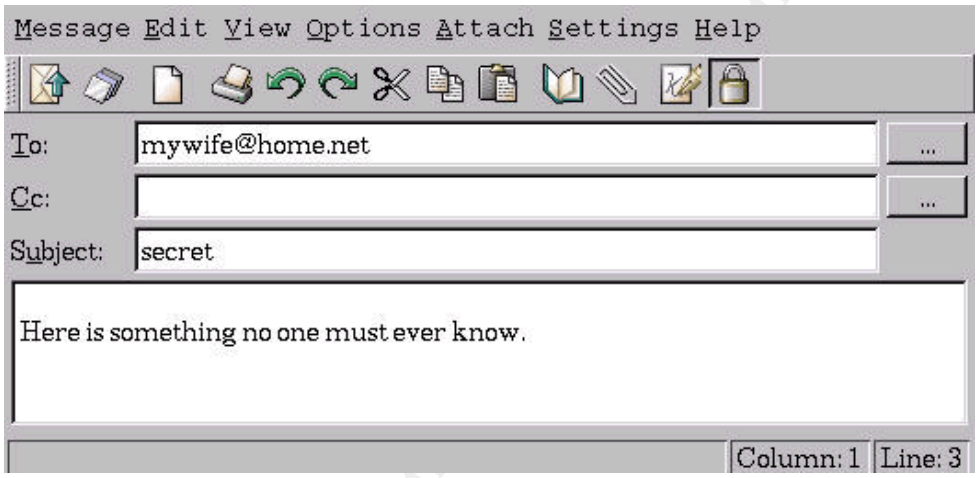
gpg will ask you for your passphrase to unlock the private key, decrypt the file, and create the file foo.txt. There are lots of options to modify the behavior of the program, for example, send the decrypted data to standard output (if you just want to read it on the screen). You should read the manual if you use this program frequently.

The only (albeit serious) problem that remains is the proof of authenticity for a given public key. Anyone can generate a public key in my (or anyone else's) name and upload it to a key server or distribute it by other means, and an unsuspecting other person would wrongly believe that he or she is securely communicating with me. PGP and GPG come with a whole framework of a "Web of Trust", where people you know and trust can vouch for the keys of others they know, and so on. In practice, I have usually obtained keys from people I know personally, and the programs come with a mechanism that allows to *fingerprint* a key. The fingerprint is a short sequence of numbers which readily fit on a business card or can be read on the phone, and give you have another, independent means of communication which will prove that the key is authentic.

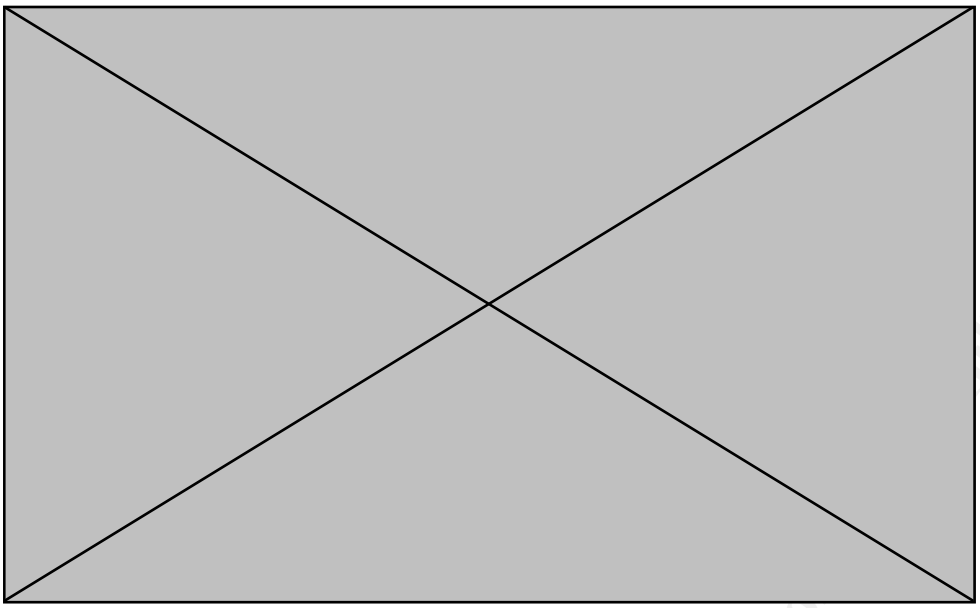
One of the few mail clients that provide a seamless integration of pgp/gpg and keyserver interaction into the mail program is kmail (another one is mailcrypt for emacs). As an example, I compose a message to my wife in kmail:



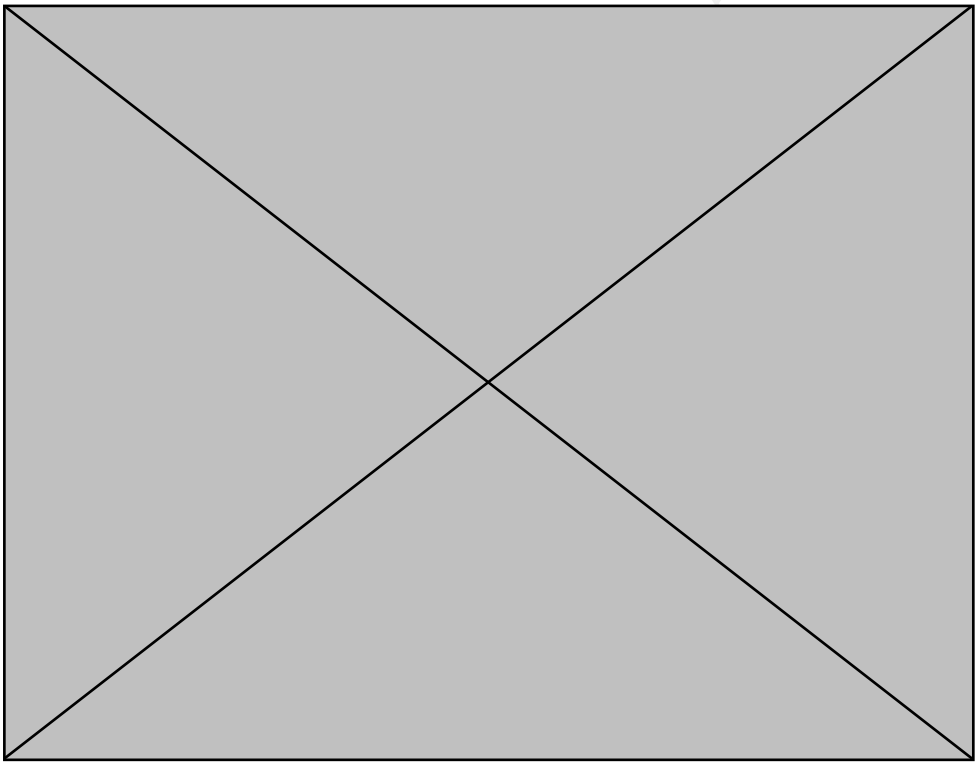
I then click on the "open padlock" icon on the right:



I get the encrypted mail in a new window, which I can then send. If the email address has a matching public key, it is used to encrypt the message:



kmail can be configured to use a variety of PGP and GPG versions:



In light of the recent discussion about strong encryption capabilities in web browsers, where key sizes of 128 bits are considered strong, you may wonder what a key size of 1024, 2048, or, as I requested in the session above, 4096 bits means. The key sizes in the symmetric encryption in the