Contents

Part I

About this book

Preface

I'm a techie by trade and not an experienced book writer. The end result of this is that many of the rules and traditions used in book writing have probably been omitted. Never-the-less, I've attempted to keep all jargon to a minimum and laid things out as clearly as possible.

In the first parts of this book we'll be looking at common tasks performed by SysAdmins. Ideally, this should help readers to understand some of what their IT department do on a daily basis as well as demystifying the system a little. In recognition of the fact that not every reader will be interested in getting hands on experience with systems management, the practical section has been kept distinct from the final part[1].

The final part examines the business case for utilising Linux. In view of the fact that many readers may not wish to actually manage their own systems, this final part has been kept distinct so that it can be read without requiring knowledge gained from earlier areas of this book.

Although in this book we'll be using the console, a wide range of graphical interfaces are available for those who feel more confident with a point-and-click interface. These range from Window managers such as KDE and Gnome to Web-based interfaces such as Cpanel, Plesk and Webmin. Most experienced SysAdmins, however, will opt to utilise the console due to the speed at which tasks can be completed!

This book has been written with Servers in mind, although Desktops are occasionally referred to. Although Linux does run perfectly well on a Desktop for most users, the reality is that most businesses will wish to continue using Microsoft^{TM} Windows® on their workstations, at least for the time being.

[1] In other words, you should be able to read the final part without needing to read the preceding parts!

What this Book Is and Isn't

I think it's important to explain exactly what this book is and isn't. That way all confusion surrounding this issue should, hopefully, be avoided allowing all arising confusion to be (quite rightly) blamed on the content!

This Book Is

This book *is* intended to help businessmen and women understand the basics involved in running and maintaining a Linux based server. It should help you to understand why some things are necessary and therefore gain a better insight into how your IT department earn their money!

Of course, if you are running a small business, it may be that you are the first point of contact for any IT issues. In that case, this book should help you become more comfortable using the Operating System that runs the vast majority of servers.

Hopefully, this book may also help you to appreciate the role that IT plays in your business.

This Book Isn't

This book *isn't* intended to teach you how to become a Linux System Administrator (SysAdmin). Understanding the day to day intricacies of running any server (UNIX, Linux or Windows) is something best left to your IT department. It is their job to remain acquainted with the results of each release cycle, and their experience ensures that they will usually resolve issues far more quickly than any non-professional enthusiast can. Just as using Microsoft^{TM} Windows® on a Desktop does not prepare you to become a Windows SysAdmin, reading a book will not prepare you to become a Linux SysAdmin either.

This book also won't attempt to teach you to understand the sense of humour possessed by the average SysAdmin, but should hopefully help you to understand some of the jargon that they use.

Part II

Becoming Familiar with Linux

Chapter 1

What is Linux?

In the strictest sense of the word, Linux actually refers to the kernel underpinning the rest of Linux based Operating Systems (OS). In practice, however, when someone says 'Linux' they are usually referring to the OS as a whole (known formally as GNU/Linux). Linux runs on your server just as Microsoft^{TM} Windows® probably runs on your desktop machine (though Linux will happily run here too!).

Linux runs on a wide range of devices, and the odds are very high that you have interacted with a Linux based system regularly without knowing about it. Whilst Microsoft^{TM} Windows® holds a monopoly position on desktop machines, this is far from the case in other areas of computing.

Most servers in use today won't be running Windows®, but may instead be running Linux, UNIX or a BSD variant. Most competent SysAdmins (if given the choice) will shy away from running Windows® on servers because there are far more suitable tools for the job.

In addition to servers, Linux is also used to run a wide range of embedded devices. Your home router, the network switches in your business, and even the set-top box connected to your TV are likely to be running a version of Linux specially customised to their intended task!

1.1 History

Linux was devised and created by Linus Torvalds in 1991 as a personal project. The kernel has consistently grown since then, and in 2009 consisted of 370MB of source code (not all needs to be compiled for any given system!). The system was originally called Freax as Torvalds considered it too egotistical to name his creation after himself, a coworker renamed the project to Linux without permission from Torvalds and the name eventually stuck.

The source code for the Linux kernel was first released in 1991 under a license prohibiting commercial distribution, but version 0.99 was released under the GNU General Public License in December 1992.

1.2 Mascot

Microsoft^{TM} Windows® has it's flag (and as of version 8 it's tiled logo), BSD has it's devil, and Linux of course has it's own mascot in the form of Tux. The name Tux is a derivative of "Torvalds' UniX". The mascot is instantly recognisable to those familiar with Linux, and has played the starring role in a wide number of applications created by Open Source Advocates.

1.3 A Free Operating System

There are two interpretations of the word 'free', and it seems best to approach these head on;

1. Free: No Cost
2. Free: Very few or no Restrictions

Linux is a Free Operating system in the latter sense. This double entendre is often phrased in the following manner "*free as in beer, or free as in speech*". Whilst it is true that many versions of Linux are available at no cost, the freedom refers to the fact that the system is released under a 'copyleft' license which imposes very few requirements on the user.

Linux is released under the GNU General Public License, which allows users to distribute copies to whomever they like. The main restriction being that any distribution must be accompanied by the source code (or an offer of) so that each user can enjoy the same freedom of use.

Paid copies of Linux are available, although most versions are distributed for free with the option of a paid support contract (which forms the business model of Red Hat® Linux).

1.4 Why Run Linux?

There are a wide number of reasons for running Linux, some select it due to the ease with which it can be customised, others due to cost. For most, though, Linux is chosen due to it's reliability (especially in the world of servers). There will be those who will only ever run Linux, but as with anything in business it's important to select the correct tool for the job.

Rather than list the multitude of reasons for running Linux, let's instead look at the few reasons why people instead opt to run Microsoft^{TM} Windows® on servers. Whilst Windows® does run on a wide number of servers worldwide, it is my opinion that the reasons for doing so usually boil down to one (or a combination) of the following reasons;

1. The server needs to run an application that is only available on Windows®
2. The decision was made by a SysAdmin who's only familiar with that system[1]
3. The decision was made by a manager who knows very little about Operating Systems but recognises the name Microsoft^{TM}[2]

The first is, of course, completely understandable. Every modern business has applications that are deemed 'mission critical', and some of these may only run on Windows®. The reality of business is that whilst there may be alternatives available, the cost involved in investigating and then migrating to a new system can sometimes be entirely prohibitive.

Reason 2 is somewhat understandable. It'd be unwise of a SysAdmin to recommend a system that they are unfamiliar with, if only because they will be unable to effectively justify the choice. There are those, however, that are entirely unwilling to step outside their comfort zone and so will only consider what is easiest for them and not what may serve the business best. You need to be able to identify these situations so that you can ensure that the needs of the business are the primary concern.

Reason 3 is an unfortunate reality. The days when the Head of IT had a direct line to the CEO are all but gone[3]. Decisions are often now made by those who have very little understanding of the systems they are discussing, and so the technical merits of any system is largely forgotten. Those who are not experienced with System Administration are also more prone to believing the marketing used by various companies, even when the reality of the system in use is entirely contradictory.

[1] Often known by the somewhat derogatory term 'WinAdmin'

[2] There's an old(ish) adage that "No-one was ever fired for choosing Microsoft^{TM}"

[3] Experience is already showing that this has come at a cost - Projects are more likely to experience overruns, both financial and in terms of deadlines. Sadly, IT is now only seen as important when something has gone wrong!

Microsoft^{TM} Windows® certainly has it's place, but it is this authors opinion that it is not on the server (reason 1 excepted).

Of course, Linux isn't the only other option. Apple^{TM} also make a server Operating System although this is not nearly as widely used as Windows®, let alone Linux or UNIX!

In the past, UNIX was *the* server operating system[4]. UNIX systems are, generally speaking, quite expensive and SysAdmins often needed additional training in order to manage the different flavours available. Linux approaches that somewhat, although there can be large differences between each of the distributions.

We'll be examining the business case for Linux more thoroughly in the final part of this book.

[4] In reality, UNIX is not one OS - many manufacturers (IBM, HP etc) make their own flavours

Chapter 2

Different Types of Linux

You will be aware, of course, that there are numerous version of Microsoft^{TM} Windows®, the same is true of Linux based systems. The reason we say 'Linux Based' is that Linux refers to the kernel (in Windows® it's ntoskrnl). The kernel being the layer of code that sits between your physical hardware and the rest of the Software on your machine.

Different versions of Linux are normally known as Distributions (or Distros for short), there are far too many to list here, but many are based on very similar grounds so we will instead attempt to list the common bases. Even those listed have (in some cases) spawned their own derivatives!

2.1 Debian Derivatives

Called Debian derivatives because their underlying structure is based on the of the completely open-source Debian distribution (infamous for not including non-free code in their builds and repositories[1]). Examples include

- ▷ Ubuntu
- ▷ Linux Mint
- ▷ AstraLinux
- ▷ Knoppix

Debian derivatives (and indeed Debian itself) usually utilise the package manager '*apt*'.

[1] A repository is a server where software is stored for download and install by a particular distributions package-manager. Often called repo for short. Will be discussed in more depth in "Installing Software"

2.2 RPM Based Systems

RPM (or RedHat Package Manager) based systems are, as the name suggests, based around the use of RPM's for installing and managing software. In the past this often proved troublesome as the result of what many call "Dependency Hell" where required software is unavailable.

Thankfully, that is largely an issue of the past due to the package manager '*yum*', which operates in a similar way to Debian's *apt*.

Examples of RPM based systems include

- ▷ Red Hat Linux
- ▷ CentOS
- ▷ Mandriva
- ▷ Fedora
- ▷ PCLinuxOS
- ▷ SUSE

As popular as Debian based systems are in the FOSS[2] community, RPM based systems are often preferred by big business. This isn't so much a reflection on the flavour of Linux, more to do with the fact that RedHat are currently the leading Linux supplier for Enterprise. Indeed in the fourth fiscal quarter of 2012, RedHat finally broke the $1bn barrier for revenue income.

2.3 Gentoo Based Systems

For a long time, Gentoo remained this authors distro of choice. Gentoo based systems utilise the '*portage*' package manager. Unlike Debian and RPM based systems, all packages are compiled from the source code (unless you specify any) allowing your SysAdmin to specify options which optimise the resulting software for your machine.

Gentoo has never been for the weak of heart, but does (in my opinion) provide some fantastic performance when utilised correctly.

Examples of Gentoo based systems include

- ▷ Funtoo
- ▷ Gentoox

[2] FOSS - Free and Open Source Software. Free as in speech rather than Free as in beer!

- ▷ Tin Hat Linux
- ▷ Chromium OS
- ▷ Google Chrome OS

Many, sadly, consider Gentoo to be too complex and so opt for alternative systems. Whether or not Gentoo is suitable for use in business, as with any other decision, is ultimately dependant on the needs of the business. Unless performance is absolutely critical, many businesses will probably find other distributions a better fit.

2.4 Others

There are many other flavours of Linux out there, some with common ancestors. They've not been included purely because the likelihood of you coming across them in use (in business at least) are limited. However for posterity's sake, those excluded include

- ▷ Pacman based systems
- ▷ Slackware based systems
- ▷ A vast range of Embedded systems (does your router run DDWRT?)
- ▷ Puppy Linux

Chapter 3

Which Type of Linux should you use?

There's a very good chance when provisioning a Linux based system that your IT Department has already selected the Linux Distro that they feel is most suited. This chapter, however, is intended to help you understand how to make these decisions. Whilst all Distros run the Linux kernel, many are built with a specific task in mind. With the wide variety of Distro's in the Linux ecosystem, it would be impossible to advise exactly which flavour is best suited, but it is possible to examine some of the assessments that need to be made before making a selection.

It's important to note that many distros maintain two versions of each release - one for Desktops and one for Servers. Although based on the same underlying software, the differences between the two can often be many. The easiest example is, of course, that many of the server focused systems will not run a GUI by default (although this doesn't stop one from being installed!)

3.1 Common Assessments

There are of course decisions that must be made whether selecting a system to run on a server or on a desktop. Those are listed here in order to avoid unnecessarily duplicating the information!

3.1.1 Familiarity

One reason for selecting a particular distro is familiarity. If your IT department are particularly used to a certain distro (for example Ubuntu) then their experience and familiarity with this system may well contribute to a decision to

utilise that distro. So long as other needs are not compromised as a result, this is often a wise choice to make - the experience a SysAdmin has with any given system will help to ensure that they can resolve common issues far more easily.

Of course, the catch with this reason is that you need to be sure that it's not being allowed to override important Business needs.

3.1.2 Security Considerations

A particular distro may be considered on the basis of security. Whilst Linux is not generally considered insecure by default, there are packages that can be installed to harden the security. Some distros already have solutions such as SELinux and AppArmor built into them, saving the SysAdmin from needing to install these himself.

Of course, it's more than possible to add these packages onto systems that do not include them in the base build, and any good SysAdmin will insist on manually configuring the applications themselves, regardless of whether or not they were included in the base build.

3.1.3 Frequency of Upgrades and Length of Support

Updates to each distro's repositories are handled by the distros themselves (although if necessary a SysAdmin can always obtain an upgrade from further up the software development chain[1].

Most systems receive regular updates, but it is the upgrade cycle that is of particular concern to the SysAdmin. Will the version being installed be obsolete in 6 months, or is it a Long Term Support (LTS) release? Generally speaking, systems deployed in Business will usually only utilise LTS releases to ensure that updates are received for as long as possible. Only once the support life is coming to an end for the installed systems will an actual upgrade be considered (and in some cases a decision will be made *not* to upgrade - just as in the Windows® world).

3.1.4 Use Requirements

This, of course, is a section that should not need to be written! The selection of a distribution should be based around what that system is intended to do. If selecting for a Desktop that's going to process and view a lot of multimedia content, it's usually considered wise to select a distribution that is aimed towards multimedia usage.

[1] Otherwise known as upstream

Ultimately, however, you should be able to customise any Linux distro to do exactly what you need. The point, of course, is that careful selection of the right distribution can lessen this overhead tremendously.

3.1.5 Does the Network need to be Homogeneous?

From a security perspective, it's usually considered wise to ensure that a network is heterogeneous. That is to say, not every system connected should be running the same Operating System and Software stack. This helps to ensure that if malware does affect your network, it won't necessarily take down every connected system.

You could, of course, opt to use the same distro for every machine and then simply install the software each user needs. This, however, is a very Windows® approach. If selected correctly, it should be more than possible to install different distros on different machines (with each selected to best suit the users needs) without any additional management overhead.

As ever, the decision must ultimately be based upon the needs of the business.

3.1.6 Package Availability

If there's a key bit of software that may be crucial to your business, it's worth considering whether or not every distribution has it available in it's repos. Whilst most distributions have a wide range of software in their repos, there may be certain niche applications that aren't covered. You can, of course, compile the software from source or install via other methods, but if a particular distro has your software in it's repo then updates will be far, far easier to apply.

3.2 Server Systems

Given that the server is Linux's traditional stomping ground, there are a wide range of Distributions focused on this particular arena. As with any business critical infrastructure, the final decision is best left to those who will be responsible for maintaining the system and handling any issues which may arise - namely, your IT department!

Generally speaking, only Server orientated distros will be considered. Whilst it's possible to run a Desktop orientated version of Linux as a server, it is usually far easier to utilise something already aimed at servers!

3.2.1 Reliability Requirements

Whilst all server operating systems need to be as reliable as possible, some businesses have far more stringent needs. If the business operates in such a way that even 1 hour of downtime per year could have severe financial implications, this needs to be seriously considered[2].

3.2.2 Support Requirements

This can be a very important decision when selecting a distro to use. If a business reasonably believes that it is unlikely to need support from the vendor, then it may opt to select CentOS over RedHat (the two systems being extremely similar), or indeed may choose a completely different distro.

3.3 Workstations

Although this book is primarily focused on Server based systems, it would be wrong to completely exclude the wide variety of Desktop focused distro's available. As with servers, your IT department is best placed to make an accurate assessment of which best meets their needs.

3.3.1 User Training

An important consideration when selecting any system that end-user will be directly interacting with: will the users need re-training? If so, how much will they need?

Whilst migration to any new system will require a little training, the level of required training can often be reduced through intelligent selection of Desktop distributions. What are your users currently used to using? If they are most familiar with Windows®, then a distribution utilising the KDE Window Manager may be a wise move.

If your users are more familiar on other systems, then perhaps a system using the Gnome Window Manager may be the choice to make.

There's no reason why alternate Window Managers cannot be installed on any system (you can even do it on Windows® if you know how!), but this consideration can easily reduce the post-installation configuration overhead.

[2] In truth, if this affects your business you should be running at least one redundant server to help mitigate the risk.

3.3.2 Working Environment

What type of environment are your users working in? Are they able to work from home, do they use PC's or Laptops? All need to be considered when planning *any* new system (Linux or otherwise). Assessment of this information can be absolutely vital when calculating what software your users should be using, if they are going to perform more complex[3] tasks like connecting to a VPN, the interface needs to be as user friendly as possible.

3.3.3 Security Requirements

What security requirements does your business have? Do the hard drives on workstations need to be encrypted by default, do users need to work in a very closely watched environment? All are easily configurable on any Linux distro, but if your business has higher security needs you may want to consider security centric distro's such as the US Department of Defence's Lightweight Portable Security (LPS).

[3] To the user at least

Chapter 4

Know your Rights

It's very important that you understand how permissions work on a Linux (or any UNIX) server. But it's also equally important to understand how your SysAdmin will assign rights, whether in a system sense or a physical sense. In this chapter we'll explore the permissions system used on Linux as well as explaining the decisions a SysAdmin will generally make when calculating which privileges to assign a user.

4.1 User Permissions

4.1.1 Need to Run Basis

If you contacted your SysAdmin now (please don't, you'll make him/her very uncomfortable!) and asked for the root login for the server, he should *at least* ask why you need them. A lot of SysAdmins will in fact deny your request without asking why (tact and diplomacy is often not widespread amongst IT Professionals!). You may be the owner of the company, you may be his boss, but security of the server is *his*[1] job. He's denying the request for good reason - with those details you could do *anything* on the server.

A basic principle of security is that you only give each user what they actually *need* to do their job. No-one should be given higher privileges than is absolutely necessary, and in some cases should only be given these privileges for a short amount of time.

[1] or her

4.1.2 Root Privileges

Root is to the UNIX/Linux world as 'Administrator' is to the Windows® world, except with a lot more power. On Windows® even Administrators are somewhat limited in what they can do, Root suffers no such limitation[2].

With root privileges there is absolutely nothing you cannot do, up to and including destroying the eeprom on the motherboard (this is *really* not a good thing to do!). It stands to reason, therefore, that root privileges are only given to those who can be trusted with them, others may be given limited rights in the form of *sudo* privileges however.

4.1.3 Sudo Privileges

Sudo is a command that will allow the user to execute commands as if they were root. Your SysAdmin can configure the system so that you have *sudo ALL* privileges (which is a little like letting you be root) or can configure specific commands that you can run using sudo.

Much like giving someone the credentials to log in as root, *sudo* privileges are only given to those who can be trusted with them!

Part of the beauty of *sudo* is that your SysAdmin can grant and remove *sudo* privileges easily and quickly (the user doesn't even need to log out!), so users can be given *sudo* access to a specific command for just as long as they need it. It also means that those users who think they have a right to run everything using *sudo* can be contained quite easily!

All attempts to use *sudo* (successful or otherwise) are generally logged in the systems authentication log which we'll be examining later in this book.

4.1.4 User Privileges

Every user should run most of their commands using nothing more than standard user privileges. These are customisable at the whim of a SysAdmin and include things like the ability to access CD drives, USB, Games and even the Internet. On a Web-server users rights are rarely customised to this extent, but the ability is there.

Of course, those users who have a need can be given *sudo* privileges and so can run certain commands as root. Most, however, are never so lucky.

The Linux equivalent (and indeed, this applies to most non Windows® systems) of 'My Documents' is your 'Home' directory. This is usually at */home/(yourusername)/*. Within this little walled garden, you are your own god. Unless root has limited

[2] Leading to the expression on one of my favourite tee-shirts - "Bow before me for I am root"

what you can do, you are free to make any changes (good or bad) that suit. What you can't do, is destroy the rest of the system and ruin the enjoyment of all the other users!

4.1.5 Group Privileges

Most permissions are assigned using groups on Linux. A user who is part of few groups will usually be able to run and access far less than one who is part of many: A user can be part of many groups, allowing easy control of who can access resources needed by multiple users. This will be examined more thoroughly when we look at file permissions, but needless to say it's never been more important to be part of the right group!

4.2 File Permissions

Everything on Linux is a file or a directory, allowing a reasonably simple Permissions structure to be used. Permissions can be assigned to the 'owner' of the file, the 'owning' group of the file and then to all others. We'll examine this in more depth in 'Meet the Console' but the basic theory is something that needs to be understood first.

This is normally represented by using three numbers denoting the permission granted to each, but can also be denoted in a far more literal way.

-rw-r–r– 1 ben backups 1964 2011-08-18 23:17 DB_info.sql

We can see from the box above that the file DB_info is owned by the user 'ben', and is owned by the group 'backups'. The owner (ben) has read/write access to the file, whilst members of the group 'backups' may only read the file. All other users have read access. This is representative of 655 permissions.

We could of course limit read/write access to the owner and read access to the group (640 permissions) which after changing would return the following result

-rw-r—– 1 ben backups 1964 2011-08-18 23:17 DB_info.sql

Or we could completely limit access to the owner (600 permission)

-rw——- 1 ben backups 1964 2011-08-18 23:17 DB_info.sql

In all of the examples so far, no-one has the right to actually *execute* the file. So if our file was a program, no-one could actually run it. For only the owner to do so, 700 permissions are required

-rwx—— 1 root root 1964 2011-08-18 23:17 DB_info.sql

Note the 'x' that has appeared. We can also assign permissions so that group users can execute the file (750 permissions), or allow all users to execute (755 permissions).

Crucially on Linux, no downloaded file will have execute permissions by default, so they must be made executable before they can run. This helps to avoid the spread of malware, although it does nothing to help users that download unknown software and then opt to run it!

Note: To delete or create a file, you need write permissions to the folder containing that file. So although you may assign 700 privileges to a file, other users will be able to delete it if the containing folder allows them to.

4.2.1 Reference Table

For easy reference the table below lists the permissions that are available

Permission	Description
0	No permissions. Cannot read, write or execute
1	Can execute, cannot read, cannot write (of limited day-to-day use as read is required to execute!)
2	Can write. Cannot read or execute (file can in effect be overwritten but not edited)
3	Can write and execute. Cannot read (in essence a combination of 1 & 2)
4	Can read, cannot write, cannot execute
5	Can read and execute. Cannot write.
6	Can read and write. Cannot execute
7	Can read, write and execute

Chapter 5

Meet the Console

Although most Linux systems have a Graphical User Interface (GUI), most SysAdmins will rarely see this when managing servers. Anything you need to do on Linux can be achieved from the console[1]. As intimidating as it may first seem, the console is not something you should be afraid of using, especially once you understand some of the features intended to make your life easier.

This chapter is best explored with a Linux system at your fingertips (though I've tried to ensure that it's not 100% necessary!). If you don't currently have access to a Linux system, you could try running in a Virtual Machine[2]. This will allow you to use a Linux system without needing to overwrite anything on your current machine!

5.1 Things you should never do

There are a number of commands bandied about by SysAdmins in jest. For safety and clarities sake, I felt it best to begin by listing them and what they do!

5.1.1 A simple rule

Let's being with a simple rule - not so much a command, but a rule that should always be observed: *Only ever use root privileges when they are absolutely necessary. Do not run as root all the time as you'll be risking the security of your system!*

[1] The console is known as the Command Line on Windows® based systems

[2] Ubuntu Server is available as a pre-built image from http://virtualboxes.org/images/Ubuntu-server. Install VirtualBox itself from https://www.virtualbox.org (the documentation sections on both sites will guide you through installing).

5.1.2 Commands you should never run

5.1.2.1 *rm -rf /*

```
rm -rf /*
```

This command will recursively delete everything on your server! Rm means Remove, whilst -rf specifies the *recursive* and *force* flags. Linux organises folders like branches of a tree, with / being the root (which is also what that particular folder is called!)

5.1.2.2 passwd file

Any command ending in *> /etc/passwd* will overwrite the file containing the systems users, including those you need to log in. Don't run these commands yourself, refer them to a competent SysAdmin before trying!

5.2 The Console

Now that we've ruled out a few commands, it's time to meet the console!

Most Linux servers will be using BASH[3]. You needn't worry too much about the various types of shell unless you are planning on actually maintaining the server yourself! To run commands we simply type the command and then press the Enter key.

5.2.1 Basic commands

Let's begin with some basic commands, each of these will do no harm but will allow you to acquaint yourself with how the shell accepts and returns information.

- ▷ List files - type *ls*
- ▷ Change directory - type *cd [directory name]* (e.g. *cd /etc*)
- ▷ Copy - type *cp [oldfile] [newfile]* (e.g. *cp /etc/hosts ~/hosts.bkup*)

So far it couldn't be simpler!

5.2.2 Comments

The shell allows you to type comments that are ignored. When interacting directly with the shell this is of very little use, it does however have real benefit when creating scripts. Comments are preceded with a hash (#) and are used extensively in later sections of this book.

You don't need to type anything preceded by a hash into the shell if you are trying the examples here, but it won't do any harm if you do!

ls /etc #this later part is a comment and will be ignored!

[3] Bourne Again SHell

5.2.3 Piping Output

We won't delve too far into this, but occasionally you will need to send the output of a command to a file, or even through another program. As complex as this may sound, it's actually incredibly easy

5.2.3.1 Directing to a file

We may want to save the output of a command to a file for a variety of reasons. Perhaps for logging, or even to investigate errors. To do so, you simply use the >. So to send the output of ls /etc to a file in our home directory we'd run

```
ls /etc > ~/ourfile.txt
```

This will run the command ls /etc but then write all output to the file in our home directory (overwriting the file if it already exists).

If we later wanted to add more data to the file, without overwriting the existing we could use a double chevron.

```
ls /etc >> ~/ourfile.txt
```

If a file doesn't exist and a double chevron is used, the system will simply create the file as though as single chevron was in use.

5.2.3.2 Piping to another command

It's amazing how often piping gets used (and misused). It allows the output of one command to be piped to another. The most obvious example would be limiting to output of ls to show only the files we want.

```
ls -l /etc | grep "host"
```

The *-l* on our *ls* command simply changes the information that ls provides. That information is then passed through *grep* which will only display lines with the word 'host' in them.

5.2.3.3 Piping Back In

So what do we do if we want to capture output now, but pipe it through to another command later? Thankfully, there's an answer for that too! We can pass the contents of a file into a command by using <. So, for sake of example, we'll run a MySQL backup and then restore it by piping back in (You will need MySQL installed to acheive this, but the example should make sense)

```
mysqldump –all-databases > mysqlbackup.sql # First we run the backup
# We could run cat mysqlbackup.sql to view the contents of the file
mysql < mysqlbackup.sql # Pass the contents of the file to the mysql client
```

The example above gives one of the most common uses for output redirection on business machines - backup!

5.2.4 Making Life Easier

Most shells now support useful features like auto-complete, this allows you type the beginning of a command and then press tab to auto-complete the rest of that command. If there is more than one possibility the system won't auto-complete, but a second press of tab will list everything available.

You can also use this function for file names, so whilst you are unlikely to use auto-complete to type *rm*, you may use it for the file name (especially useful where backups have a long file name!)

Try typing *ls /et* and then press tab once.

5.2.4.1 *grep*

In the previous section we piped output from a command through *grep*. This command is incredibly useful when searching large files for specific words (or if you need to exclude lines), it's therefore deserving of it's own section!

We previously used a simple *grep* command to display only the lines that contained the word host. But what if we wanted to go further than this and exclude some specific results? So to begin with we received the results

```
ls /etc | grep host
host.conf
hostname
hosts
hosts.allow
hosts.deny
```

But (for whatever reason) we don't want to display lines containing 'hosts'. We can pipe the output through a second *grep* call using the exclude flag (-v)

```
ls /etc | grep host | grep -v hosts
host.conf
hostname
```

Of course, *grep* doesn't just read from standard input, you can also search files directly.

```
grep "ben" *.php
```

Will search any PHP file in the current directory for the word 'ben'.

There are endless possibilities to the ways in which you can limit results with grep, but it currently supports the following flags (only those considered basic usage are included!)

Flag	Description
-c	Print only a count of how many matches were found
-H	Prints the name of the file containing the match
-i	Case Insensitive search
-l	Print only the names of files containing a match
-L	Print only the names of files containing no match
-m NUM	Stop looking after NUM of matches
-n	Print the line number with matching lines
-o	Only display the parts of the line that match
-r	Recursive: look inside directories
-v	Invert match (hide matching lines)
-w	Match only against whole words
-x	Match only against whole lines

5.2.4.2 *man*

As we've already seen, most command will accept a wide variety of input flags, and there's not always a consistent pattern between the desired end result and the argument you need to use!

However, no SysAdmin is actually expected to remember the syntax for every command they may need to run[4]. Instead we have *man*(ual) pages to remind us, although not every command has a *man* page, most do.

To access the man page for a specific command simply type *man* followed by the command, so to view the *man* page for *cp* we would enter

```
man cp
# Use q to exit the man page
```

5.2.4.3 –help

Most commands support the argument *–help* or *-h*. This will output a list of the arguments that a command accepts and will usually include basic information on

[4] Though the more you learn the more efficient you'll be.

syntax. Although not generally as detailed as the man page, the *–help* argument can be very helpful when trying to remember a specific argument.

```
cp –help
# cp doesn't support -h
# Single out the flag relating to updates using grep
cp –help | grep update
```

5.2.4.4 Wildcards

No system would be complete without the ability to accidentally wipe multiple files at once! Linux, of course, supports the wildcard. This allows you to enter part of a file name with a wildcard used to inform the system to match against anything containing the section you specified.

So for example

```
ls /etc/hos*
```

Will return any files in /etc with a file name beginning 'hos'

```
ls /etc/*p
```

Will return any files in /etc ending in a 'p'

```
ls /etc/*p*
```

Will return any files in /etc with a 'p' in their name

A word of warning: Be very careful when using wildcards with anything that actually affects files. It's easy to make a mistake and remove/overwrite files that you didn't expect to be included! It's always worth a test run using *ls* first!

5.3 Changing Permissions

5.3.1 File Permissions

We briefly examined the concept of file permission on Linux in a previous chapter. In this section we'll look at how you actually go about checking and changing permissions. As we don't want to risk breaking the system we are using, the first command will create a file that we can tweak with impunity!

```
echo “” > ourfile #create our test file
ls -l ourfile #Show the current permissions for the new file
```

You should have received output similar to the following

```
-rw-r--r-- 1 ben ben 1 2012-03-03 15:18 ourfile
```

Currently any user can read our file. We can limit read access to ourselves and the group (in this case also called ben) by assigning 640 permissions.

```
chmod 640 ourfile #Change the permissions
ls -l ourfile #Display the new permissions
-rw-r----- 1 ben ben 1 2012-03-03 15:18 ourfile
```

We may, of course, not even want to allow the group read access, so lets assign 600 permissions

```
chmod 600 ourfile #Change the permissions
ls -l ourfile #Display the new permissions
-rw------- 1 ben ben 1 2012-03-03 15:18 ourfile
```

Assuming the file was something we wanted to execute, we would need to add execute permissions. We want to keep the contents of the file private though, so we'll only assign permission to ourselves (as the owner of the file)

```
chmod 700 ourfile #Change the permissions
ls -l ourfile #Display the new permissions
-rwx------ 1 ben ben 1 2012-03-03 15:18 ourfile
```

Similarly we can apply permissions to the group whilst leaving non-group members unable to access the file

```
chmod 760 ourfile #We can execute, Group can read & write
ls -l ourfile #Display the new permissions
-rwxrw---- 1 ben ben 1 2012-03-03 15:18 ourfile
```

So as you can see, each of the numerical characters used in the command applies to different categories of user.

chmod	7	6	0	*filename*
Command to change permissions	*Owner*	*Group*	*All Other Users*	*The file to change*

5.3.1.1 The Extra Hyphen

Eagle eyed readers will have noticed that there is in fact an extra dash in the examples above. At the beginning of the permissions string is a dash indicating that the file is not a directory. If we were to run the following command, that dash would be replaced with the letter 'd'.

```
ls -l / | grep etc
drwxr-xr-x 137 root root 12288 2012-02-12 13:40 etc
```

5.3.2 Changing File Owners

5.3.2.1 User

The matter of which user 'owns' a file is of the utmost importance, in many cases they will be the only user assigned permissions to actually edit a file. So using our example above, where we currently own the file, how do we assign it to another user (in this case root).

Quite simply by using the *chown* command

```
ls -l ourfile #Display the current ownership and permissions
-rwxrw—- 1 ben ben 1 2012-03-03 15:18 ourfile
chown root ourfile #Change the owner to root
ls -l ourfile #Display the new ownership and permissions
-rwxrw—- 1 root ben 1 2012-03-03 15:18 ourfile
```

If the above example returned *"chown: changing ownership of 'ourfile': Operation not permitted"* it means that you lack the permissions to change the file to be owned by another user. In this case you would usually use root privileges to acheive this (we'll examine *su* and *sudo* shortly).

5.3.2.2 Group

Group permissions are also quite important, as it allows you to assign privileges to a group of users without needing to let any user on your system have those privileges. There are two ways to change the group ownership of a file, both of which we will now examine.

```
ls -l ourfile # Display the current permissions
-rwxrw—- 1 root ben 1 2012-03-03 15:18 ourfile
chgrp users ourfile
ls -l ourfile # Display the current permissions
-rwxrw—- 1 root users 1 2012-03-03 15:18 ourfile
```

As with our example in the previous section, if you receive the error "*chgrp: changing group of 'ourfile': Operation not permitted*" this simply means that you do not have the ability to assign files to other groups. Again, you'd normally run this as root instead.

The second method (Which will also return the above error if you lack the rights) is to specify the desired command to the *chown* command we examined in the previous section. Let's change the group root back to users;

```
ls -l ourfile # Display the current permissions
-rwxrw—- 1 root root 1 2012-03-03 15:18 ourfile
chown root:root ourfile
ls -l ourfile # Display the current permissions
-rwxrw—- 1 root users 1 2012-03-03 15:18 ourfile
```

5.3.3 Elevating our User Privileges

If you've been trying the examples above, you may have come across errors informing you that an operation was not permitted. This is because our commands were run as an ordinary user (unless you logged in as root). In some cases we may need to elevate our privileges to acheive an end result. There are three ways of doing this, all of which are afforded to you by whoever *is* root on your system

- ▷ Log out and log in as root (needs root password and is quite a hassle)
- ▷ Su (needs root password)
- ▷ Sudo (Doesn't need root password)

We won't examine the first possibility as it should be fairly self-explanatory and is something we'd normally aim to avoid doing where possible!

5.3.3.1 *su*

The command *su* allows us to temporarily switch user. To do so our user must be part of the appropriate group (usually called *wheel*) and we will need the password for the account we intend to switch to (unless we are running it as root).

When no username is specified, su defaults to the root account. So to work around our "Operation Not Permitted" issue we would do the following.

```
su
#Enter root password and press enter
chown root ourfile #It'll work this time!
exit # We don't want to stay root for any longer than we need to
```

It really is that simple! Of course, it does mean that your SysAdmin needs to give you the root password which is why *sudo* is generally the preferred method on multi-user systems. Additionally, once a user has been added or removed from the group *wheel*, they will need to log out and then back in for changes to take effect.

5.3.3.2 *sudo*

Sudo works in a similar way to *su* but allows the SysAdmin to limit what you can do. As a user with sudo privileges you can opt to run a single command, or to run the shell as root (the true root can of course prevent this!).

Once the user has been added to sudoers (/etc/sudoers) running a command as root is as simple as

```
sudo chown root ourfile
#Enter your password and press enter (not roots password!)
```

The true beauty of *sudo*, however, is that the SysAdmin can specify exactly which commands each user can run using sudo (or can simply say they can run anything). This allows the SysAdmin to specify, for example, that you can run *chown*, *chmod* and *chgrp* but not *rm*, *mv* or *cp* using root privileges[5].

Sudo also has the added benefit that a user needn't log out and then back in once they have been assigned permissions. Being based on a configuration file, changes are immediate (allowing the SysAdmin to quickly revoke privileges if necessary).

We'll look at how to assign *sudo* privileges in a later part of this book

Tidying up from our Examples

If you've been actively trying the examples in this section, you'll now have a file called '*ourfile*'. We no longer need this, so we'll remove it!

```
rm ourfile
```

5.4 Useful commands

We'll explore various commands throughout this book, but for reference I thought it best to list some of the basic commands, what they do and examples of how to use them.

[5] Which will quickly reduce the damage that the user can do!

Command	Example Call	Description
cat	*cat /etc/fstab*	Cat outputs the contents of a file to stdout (which is usually the console you are using). This allows you to check the content of a file
cd	*cd /etc*	Changes the current working directory to the one specified. If no directory is specified it will take you to your home directory.
chgrp	*chgrp backups*	Change the owning group for the specified file or directory.
chmod	*chmod 766 somefile*	Change the permissions assigned to a file or folder. Limits who can do what with the file.
chown	*chown ben somefile*	Change the owner of the specified file or directory
cp	*cp somefile somefile.new*	Copies the specified file to the specified location. If no path is specified then the operation takes place in the current directory
grep	*grep "/" /etc/fstab*	Grep searches the specified file for the specified string. In the example grep would return all lines in the file /etc/fstab which contain a forward slash.
less	*cat /etc/fstab | less*	Less allows you to scroll through a large amount of output (if *less* is not available, try *more*). Press q to close
ls	*ls /etc*	List all contents of the specified directory. If no directory specified then the current directory is used
mkdir	*mkdir mydir*	Create a directory, can be specified with the -p flag to specify that any non-existing directories required to make the specified directory should also be created (e.g. mkdir -p mydir/anewdir/testdir would create all three at once)
mv	*mv somefile somefile.new*	Moves the specified file to the specified location. If no path is specified then the operation takes place in the current directory
nano	*nano somefile*	My text editor of choice, not available on all systems but can easily be installed.
pwd	*pwd*	Prints the path to the directory you are currently in. Useful if you've lost track!
rm	*rm somefile*	Remove a file, can also remove directories when the recursive flag is specified (-r). Should be used with caution!
su	*su*	Switch User - usually used to become root
sudo	*sudo somecommand*	Temporarily use root privileges to run a command
wget	*wget http://foo.com/bar.zip*	Wget is a command line utility for obtaining data from web/ftp servers. Any file that can be downloaded through your browser can also usually be obtained with wget. This command is also useful for triggering web-based scripts.
ln	*ln -s myfile /home/myfile*	ln creates links to files. In the Windows world these are known as 'shortcuts'. On Linux it's possible to create different types of links, but the example shows the creation of a symbolic link, the closest approximation to the Windows shortcut.

Part III

Installing Software

Chapter 6

Introduction

So we've now had a brief introduction to the shell, although the most Linux distros do come with a wide base of software installed, it's quite likely we'll need to install software at some point. In this part we look at the ways in which we can do this, however use of the package manager differs between different distros so a chapter has been devoted to each.

The final chapter of this part explains how we compile software from source. This can be of use when a particular piece of software isn't in our distro's repositories, and works in the same way across nearly all distributions.

Before proceeding, consider whether you are connected to an actual server or are using a virtual machine[1]. If the latter then it's safe to try the examples in this book, but if it's the former then you **must** consult your SysAdmin before proceeding: we will be installing, updating and removing software!

6.1 Introduction to Package Managers

Why do we use package managers? There's more to it than the convenience of needing one command to install almost *any* piece of software.

Package managers also take care of dependencies (more on that shortly) and will notify us when an update is available. Put simply, they make life much, much easier for the SysAdmin.

There is, however, a small catch. Because it takes a little while for software to appear in the repositories, the versions available through a package manager are often older than those available to compile from source. This doesn't necessarily pose a problem, but can be a minor cause of headaches when the version in the repos lacks a feature you need.

[1] If you're simply reading this book without trying the examples, proceed!

6.2 Dependencies

Most software depends on the presence of other software (or libraries) in order to function correctly. Part of the aim of package managers is to identify and install those dependencies as well. When we compile from source in the final section of this chapter, it will be our responsibility to locate any dependencies and install those as well.

Occasionally, a package manager will report that it cannot install a dependency. This may be because the version needs clashes with another piece of software on the system, a broken package or simply that the dependency isn't available in the repo's[2].

We'll look at how you resolve this issue for each of the package managers.

6.3 Which Package Manager Are You Using?

If you're unsure as to which package manager your system uses, try each of the following until you get a meaningful response!

```
sudo apt-get
sudo yum
sudo emerge
```

Explanation:

- ▷ If apt-get returned a response other than "command not found" then you are running a Debian based system.
- ▷ If yum returned a response then you are running a RPM based system
- ▷ If emerge returned a response then you are running a Gentoo based system.

If none of the above worked, then unfortunately this book doesn't contain a section dedicated to your package manager. The chapter on compiling from source code should still apply to your system however.

[2] In the past on RPM based systems this was often referred to as "Dependency hell"

Chapter 7

Debian Based Systems - Apt

When running a Debian based system, the package manager is called 'Apt'. Whilst many of the other package managers replicate the functionality of *apt* it is still generally considered the most powerful and user-friendly of the systems.

Apt will only work correctly if you run it with root credentials (using *su* or *sudo*), this is because we will be installing software into areas of the system that ordinary users cannot write to.

In this chapter we will be installing Apache's mod_perl using Apt.

7.1 Find the package name

We may, of course, already know what the package is called. Sometimes, however, we may not know and so need to search for the name of our package.

```
sudo apt-cache search perl
[Enter your password if prompted]
```

The above command will probably have returned a multitude of packages (Perl is quite heavily used). As we know the package is a module for Apache, we can trim the results down a bit by using *grep*.

```
sudo apt-cache search perl | grep -i "Apache"
```

Note: The 'i' in the above grep command simply informs grep to perform a case-insensitive search.

We should still have received quite a number of results, but it should now be far easier to see the package we want - *libapache2-mod-perl2*

7.2 Install the Package

Now that we know the name of the package, we can install it!

```
sudo apt-get install libapache2-mod-perl2
[Enter your password if prompted]
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed: apache2 apache2-mpm-worker apache2-utils
apache2.2-bin apache2.2-common libapache2-reload-perl libaprutil1-dbd-sqlite3 libaprutil1-
ldap libbsd-resource-perl libdevel-symdump-perl
Suggested packages: apache2-doc apache2-suexec apache2-suexec-custom
The following NEW packages will be installed apache2 apache2-mpm-worker apache2-utils
apache2.2-bin apache2.2-common libapache2-mod-perl2 libapache2-reload-perl libaprutil1-dbd-
sqlite3 libaprutil1-ldap libbsd-resource-perl libdevel-symdump-perl
0 upgraded, 11 newly installed, 0 to remove and 142 not upgraded.
Need to get 4,285kB of archives. After this operation, 13.8MB of additional disk space will
be used.
Do you want to continue [Y/n]?
```

You may also receive output stating that some packages we automatically installed and are no longer required, you can safely ignore this for now!.

Press Y to continue and Apt will fetch all of the listed packages and install them. Any extra package-dependant steps you may need to take post-install will usually be listed so that you can see them.

7.3 Upgrading Packages

You don't generally update a single package with apt. Instead Apt will find updates for everything that is available and will then install those.

Note: Don't run this on your server without consulting your SysAdmin. Production servers are not generally updated until the update has been tested to ensure nothing breaks!

To run an update, you simply run

```
sudo apt-get -u update
[Enter your password if prompted]
```

As with installing you will be shown a list of packages that will be updated and then will need to confirm that you actually want to proceed.

7.4 Removing Packages

We've now install mod_perl. What do we do if we decide it's no longer needed? We simply tell Apt to remove it

```
sudo apt-get remove libapache2-mod-perl2
```

A list of all packages to be removed will be generated (if there are extra packages displayed it's because they rely on the one that you are removing).

Press Y to confirm the removal and Apt will remove the package.

7.5 Resolving Dependency Issues

This section is here more to explain what your SysAdmin will do when dependency issues arise. It's *not recommended* that you attempt this yourself as it becomes very, very easy to break things.

You will know when you've hit a dependency issue because Apt will tell you which dependencies couldn't be satisfied and why. From there on you have a few options

1. Check the available packages list is up to date
2. Add a Repository containing the package
3. Manually install the dependency yourself
4. Instruct Apt to install without the dependency

Option 1 is the very first thing you should try. Run the command

```
sudo apt-get update
```

This will update the package list, once this is complete try to install your package again. If this fails then move onto the following options.

Option 2 is reasonably simple to acheive, but is package dependant. If you can find a repository with the dependency in (a web search does wonders for this!) the repository will normally publish instructions on how to add them to your system.

Option 3 is the route that you should take if the previous two have failed, and unless it's clear that it's not necessary most SysAdmins will proceed to locate the source code for the dependency (Google is a great help here!) and then will compile and install the dependency (see later in the book for instruction on this).

Option 4 is something that should be avoided where possible. Depending on what you are installing, failing to install dependencies could have a wide range of ramifications. The most likely, however, is that the software you are installing simply won't work.

However, to instruct Apt to install without dependencies we simply run

```
sudo apt-get libapache2-mod-perl2 –nodeps #Install libapache2-mod-perl2 but ignore depen-
dancies
[Enter your password if prompted]
```

Again, it cannot be understated that you must consult your SysAdmin before running any command that installs or removes software, but especially so if you intend to use –nodeps.

Chapter 8

RPM Based Systems - Yum

Many RPM based systems now use *yum* as their package manager as it resolves the historic issues associated with satisfying dependencies when installing from a single RPM package. It is also possible to install software using the rpm command, but that falls outside the scope of this book as satisfying dependencies in this way can become very complex and is therefore best left to your SysAdmin!

We'll be installing Apache's mod_perl in this chapter.

8.1 Find the Package Name

Installing software with *yum* isn't difficult, but it can sometimes be a challenge to work out the name of a package in the repositories. Thankfully, *yum* supports the search command, so we can list packages that may be of interest.

```
sudo yum search perl
[Enter your password if prompted]
```

This will, almost inevitably, have returned an incredibly long list of packages. So let's filter the results down a little with *grep*.

```
sudo yum search perl | grep -i "Apache"
```

We should now see fewer results, and can see by reading though that the package we want is mod_perl. You may notice that there is a .i386 on the end of the package name, this is indicative of the processor architecture that you are running[1].

[1] At this stage it can safely be disregarded

8.2 Install the Package

Now that we know the name of our package, we can instruct *yum* to install it.

```
sudo yum install mod_perl
```

You won't usually be asked to confirm the action, but will see output as yum processes the dependencies and begins to install the packages.

8.3 Upgrading Packages

Yum allows you to update individual packages[2] or to update the entire system. We'll update mod_perl (although we've probably just installed the latest version!)

```
sudo yum update mod_perl
```

Yum should now check for updates and install them, or will tell you that the package is already the newest version.

8.4 Removing Packages

So we've installed mod_perl, but what do we do if it's no longer required? We use yum remove.

sudo yum remove mod_perl

The package will then be removed, along with any packages that were dependant on the one we've specified.

8.5 Resolving Dependency Issues

As with any package manager, Yum will try to resolve all dependencies for you. However, it will occasionally fail to satisfy one and will abort the installation process with an error.

Unlike *apt*, *yum* doesn't include a –nodeps option because of the potentially destructive nature of running with this flag. However, it is possible to use rpm to install using –nodeps. A better option, however, is to find the dependency (A Google search is often helpful, as is rpmfind.net) and then manually install it.

[2] Although they may have dependencies which also require updating.

You can either download the source for a dependency and manually install it (we'll be doing this in a later chapter) or you can download and install a RPM containing the dependency.

If the latter then having downloaded the RPM you need to issue the following command

sudo rpm -i /path/to/downloaded/rpm

You should *never* use the –force or –nodeps option without first consulting your SysAdmin. Both have the potential to break the system. Using –force will tell RPM to ignore any errors or conflicts, if the package you are installing conflicts with something essential on your system you will experience serious errors!

Chapter 9

Gentoo Based Systems - Portage

Gentoo based systems generally use a package manager called *Portage*. Unlike other package managers Portage compiles software from source code by default. This allows SysAdmins to specify compilation flags tailoring the software to their hardware in order to try and achieve better performance.

Specifying compilation flags isn't something we will do here, as it's advanced enough to fall outside the scope of this book.

If you receive an error saying that a package has been masked, it means it's been made unavailable for install[1]. It is possible to force an install of masked packages, but this should not be done lightly. Masked packages could potentially break your system, so in the interests of responsibility the methods for overriding a mask won't be included here!

9.1 Updating Package Lists

In order for Portage to use the latest package versions, a list of all available packages needs to be downloaded. It's considered impolite to run this command more than once a day, and doing so may lead to a temporary block on your IP preventing you from downloading this information from the Gentoo repositories

```
sudo emerge –sync
```

[1] Usually as a result of bugs or other issues

9.2 Find the Package name

As with any other package manager, the actual names of packages aren't always obvious. So we will search for the package we wish to install (*mod_perl*).

```
sudo emerge –search perl
```

In this case we can see that the name of the package is in fact mod_perl!

9.3 Install the Package

To install a package we also use the command *emerge*. To install mod_perl we would simply run

```
sudo emerge -av mod_perl
```

It's good practice to always use -av. These flags specify that emerge should be verbose about what will be installed as well as requiring that you want to continue. You could instead run

```
sudo –pretend mod_perl
```

This will run a simulation of the install without actually making any changes on disk. You'll then be able to see exactly which dependencies would be installed.

9.4 Upgrading Packages

Generally speaking we would update the entire system. Before you can perform an update you'll need to ensure that you have run *emerge –sync* recently.

9.4.1 Update a single package

```
sudo emerge –update -av mod_perl
```

9.4.2 Update the entire system (including dependencies)

```
sudo emerge –update –deep -av world
```

9.5 Removing Packages

Removing software is just as simple as installing it, in Gentoo terms we wish to 'unmerge' it. So we would issue the following command

```
sudo –unmerge -av mod_perl
```

Again we've used the -av flag so that we can ensure that we aren't going to be removing any software that we may need but relies on the software we are removing. In this case, there shouldn't be any.

9.6 Resolving Dependency Issues

Portage is generally *very* good at handling dependency issues. There may be occasions, however, when it becomes the SysAdmin's responsibility to resolve issues. Most common causes are either that a dependency has been masked or that a dependency clashes with another package on the system.

Depending on the reasons for the package being masked, it may be appropriate to unmask it and re-run the install. It may, however, also be appropriate to simply download the source and compile by hand.

If the dependency clashes with another package, things become more complicated. You need to examine which other packages rely on the software[2] in order to assess whether it can be removed. Failing that, a web-search may yield helpful results.

9.7 Useful Flags

The commands that make up portage accept a wide number of flags, for easy reference though the most useful of the *emerge* flags are listed below.

[2] By this we mean the already installed software causing the clash

Flag	Description
–unmerge	*Remove a package*
-a	*Ask for confirmation before performing the action*
-v	*Be verbose in output*
–pretend	*Simulate an install*
–fetchonly	*Download the source code but don't install it yet*
–search	*Search Package Names*
–searchdesc	*Search package descriptions*
–update	*Perform an update*
–deep	*Include dependencies (used for updates)*
–newuse	*You've specified new use flags*
–depclean	*Removed orphaned dependencies*

Chapter 10

All Systems - Compiling from source

10.1 Introduction

To the uninitiated, the idea of compiling software from source code can sound incredibly daunting. In reality, it's seldom an issue so long as all the dependencies are met. In this chapter we'll be looking at how we usually compile software from source, be aware that some programs have their own way of doing things. Most will include a file called "INSTALL" which you can read once you have extracted the package.

For this example we'll be downloading and installing a package called s3fs.

10.2 General - Which Package to Download

Many projects will host a number of packages to download. Any package marked as a source tarball[1] should do what you need, but try to ensure that you are downloading stable software (i.e. not alpha, beta or developers build).

For the purposes of our example however, we will be downloading a tarball containing Version 1.19 of S3Fs.

```
sudo -s
cd /usr/local/src
wget http://s3fs.googlecode.com/files/s3fs-1.19.tar.gz
exit
```

Retrieving the file was that simple! We can now move onto the next section where we will see how to extract the tarball ready for compilation.

[1] The filename usually ends in .tar.gz or .tar.bz

10.3 Extracting the package

In our example we have downloaded a tarball containing version 1.19 of S3fs. It's been saved to /usr/local/src with a filename of s3fs-1.19.tar.gz. We can see, from the filename that this is a Gzipped tarball so we run the following commands to extract it

```
sudo -s
cd /usr/local/src # change to the directory we want to extract to
tar xvzf s3fs-1.19.tar.gz
exit
```

However, what would we do if the file was BZipped? The filename would most likely end in .bz or .bz2 so we would instead run

```
sudo -s
cd /usr/local/src
tar jxvf s3fs-1.19.tar.bz2
exit
```

Note: Recent versions of tar will automatically detect the type of tarball, so running tar xvf filename may be sufficient on your system!

This will have created a folder called s3fs-1.19. You could also tidy up and issue the command *rm s3fs-1.19.tar.gz*, but in either case we now need to move onto compiling and installing our software.

10.4 Compiling and Installing

In the next few sections we'll be configuring, compiling and installing our example software. We'll then uninstall it as we don't currently have a need for it!

Every command should be run with superuser privileges unless stated otherwise, so run the following commands first

```
sudo -s
cd /usr/local/src/s3fs-1.19
```

10.4.1 Configuring

Most software (including in our example) will include a script intended to configure a few settings prior to compilation. Some programs will require specific options to be specified, however running the command below will work in most cases (and where it doesn't the configuration script will *usually* tell you what you need to do).

```
./configure
```

If all went well the script should have completed without error. If errors were displayed you need to resolve them. Unless the script tells you specifically what needs doing, a websearch is often a very good place to start.

10.4.2 Compiling

Our example program is now configured and ready to be compiled. To do so we simply issue the following command

```
make
```

If the process completes without error, the program is compiled. We may even be able to run it from the local directory (not always possible depending on the software itself!). In the case of our example, you should be able to obtain some basic output

```
./src/s3fs #Should return the following
s3fs: missing BUCKET argument
Usage: s3fs BUCKET MOUNTPOINT [OPTION]...
```

Although it's returned an error[2], we can see that it runs successfully and so can proceed to install it system wide.

10.4.3 Installing

We now have a fully compiled piece of software needing to be installed system-wide. It's important to note here that not all software needs a system wide install. As an unprivileged (i.e. non-root) user you could compile software in your home directory and simply skip this step. You could then run the software in the manner we tried just after compiling.

To affect a system wide install we run

```
make install
```

The system will then copy relevant files to key locations. We should now be able to run the software as any user (providing permissions and groups allow that!)

[2] As a result of our not specifying required arguments

10.4.4 Running your newly installed program

We've installed S3FS on a system wide basis, we're not actually going to use this piece of software as such, but will demonstrate how to call it from the command line (if in doubt read the README file!).

You should be able to run this as a non-root user, but it will work if you are root too

```
s3fs # Should return the following
s3fs: missing BUCKET argument
Usage: s3fs BUCKET MOUNTPOINT [OPTION]...
```

Again, as we omitted required arguments an error was returned. The difference between this and our previous attempt is that our post-compilation test required us to specify a path to the program. The '*make install*' step removed this necessity, and we can now call s3fs from anywhere on the system.

We'll now uninstall the software as it's not really required.

10.4.5 Uninstalling

There are two steps to uninstalling the software. The first is to 'undo' the *make install* step, the second is to remove the source code and the compiled software from the hard drive.

```
cd /usr/src/local/s3fs-1.19
#Undo the make install
make uninstall
```

Note: Not all software supports make uninstall. Some has other methods of uninstalling (will be detailed in the documentation) or simply does not support uninstalling.

We've now removed the ability to run S3fs without specifying a path. To check simply type *s3fs* and press enter, you should receive the error "command not found". We could, however, still run by entering *./src/s3fs* and issue we will now resolve.

Our current working directory is still the s3 source directory, so lets move up to the parent directory and then remove the source

```
cd .. #Move to the parent directory
rm -rf s3fs-1.19 # Recursively remove the source directory
```

We've now removed the directory containing the source code. As our compiled version was in a sub-directory of this, that's also no longer on our hard drive.

10.4.6 Steps Combined

For quick reference, the example we just worked through is detailed in one simple list of commands below

```
sudo -s
cd /usr/local/src
wget http://s3fs.googlecode.com/files/s3fs-1.19.tar.gz
tar xvzf s3fs-1.19.tar.gz
rm s3fs-1.19.tar.gz # Tidy up
cd s3fs-1.19
./configure
make

# We won't test the compiled version this time as it was only for illustration!
make install

# S3fs is now installed system wide and can be called with s3fs
make uninstall

# No longer installed system wide, remove the source directory
cd ..
rm -rf s3fs-1.19

# Compiled and source versions no longer on hard drive!
```

Part IV

Basic System Administration

Chapter 11

Introduction

Throughout this book you've been told you should consult your SysAdmin. In this part we will be making a leap of faith and assuming that you *are* the SysAdmin[1]. Every effort has been expended in trying to ensure that you won't destroy your system by trying these examples, however be aware that you do now have the capability to do so. If you are in *any* doubt as to the ramifications of a command you **must** consult an experienced SysAdmin before trying it!

Within this Part we'll be examining

- ▷ Task Scheduling
- ▷ Checking for Malware
- ▷ User Management
- ▷ Service Control
- ▷ Basic Firewall Rules

[1] Why else would you be trying these tasks?

Chapter 12

Scheduling Tasks

A server that needs to be manually told what to do is of very little use. However, Linux (as with any good OS) allows you to schedule tasks to run, whether just the once or regularly.

12.1 Cron

Cronjobs are tasks that are scheduled to run regularly. These might include your backups, a server status email or anti-malware checks.

Cronjobs are stored in a file called the 'crontab'. In order to add a cronjob we need to log in as the user we wish the crontask to run as (unless we are root).

We then run the following command which will open a text editor

```
crontab -e
```

There's a very specific format to use when adding a cronjob, which is as follows

0	**0**	**0**	**0**	**0**	**myscript.sh**
Minute	Hour	Day of Month	Month	Day of Week	Command to run

As complex as it may look at first glance, it's actually a very simple way of scheduling cronjobs[1]. Let's have a look at the values that each will accept.

[1] Once you've managed to remember which number schedules what!

Column	Numbers Accepted	Notes
Minute	0-59	The minute of the hour you want the job to run at
Hour	0-23	The hour you want the job to run at
Day of Month	0-31	The day of the month you want the job to run
Month	1-12	The month you want the job to run in
Day of Week	0-6	0 is Sunday, 6 is Saturday

Note: You can specify multiple numbers by including a forward slash. You can also use a wildcard (*) to specify all[2].

So let's look at a few example crontabs and what they mean

```
0 0 1 1 * myscript.sh
```

This cronjob will run once a year, on the first day of the first month (1 Jan) at 00:00.

```
0 0 1 * * myscript.sh
```

This cronjob will run once a month, on the first day at midnight

*0 0/2 * * 0 myscript.sh*

This job will run at midnight at 00:00 and 02:00 every Sunday

12.1.1 Short-names

Certain schedules can be set using short-names. You simply define these in place of the cron scheduling string.

Entry
@yearly
@monthly
@weekly
@daily
@hourly
@reboot

```
@monthly mymonthlybackup_ script.sh
```

[2]Think carefully before using a wildcard for minute or hour though. If minute is set to a wildcard the job will run every minute that the other columns are satisfied!

12.1.2 Useful Schedules

It's often easier to understand the cron scheduling string if you can see useful schedules laid out. Below are a number of cron schedules that you may see in use

Entry	Description
0 0 1 1 *	Run once a year at midnight on Jan 1st
0 0 1 * *	Run once a month at midnight on the 1st
0 0 * * 0	Run once a week at midnight on Sunday
0 0 * * *	Run once daily at midnight
0 * * * *	Run once an hour at beginning of hour
0/15/30/45 * * * *	Run once every 15 minutes

12.1.3 Viewing all Cronjobs

There is, unfortunately, no easy way to list all cronjobs created by all users. However, the following command will display this information for you (you need to run it using root privileges)

```
sudo -s
for username in $(getent passwd | cut -f1 -d:)
do
echo $username
crontab -u $username -l
done
exit
```

This will loop through every user on the system, output their name and then list the contents of their crontab.

12.1.3.1 Viewing a specific users crontab

You need to use superuser (root) privileges to acheive this.

By calling *crontab* with the user flag (-u) you can output the contents of a specific users crontab

```
sudo crontab -u ben -l # List the contents of bens crontab
```

12.1.4 Controlling who can create Cronjobs

Of course, a SysAdmin may not want users to be able to create cronjobs. At the very least, there may be specific users who the SysAdmin explicitly doesn't

want to schedule cronjobs. There are two ways of doing this, whitelisting and blacklisting.

I've assumed that the file doesn't already exist, but if it does you can edit it using a text editor such as *nano* (*nano /etc/cron.allow*)

12.1.4.1 Whitelisting

In order to whitelist users, create the file **/etc/cron.allow**. Once this file exists, only usernames listed within will be able to create cronjobs, all others will be denied!

Whitelisting is considered the more secure way of implementing any security, but be aware that you will need to manually add any users that you might later create.

To create this file you will need to assume superuser privileges by using su or sudo

```
sudo echo "ben" >> /etc/cron.allow
```

12.1.4.2 Blacklisting

If a single user is an issue, then creating a blacklist file (/etc/cron.deny) will probably be a more effective route. To do so simply create the file and add any users you wish to block to the file.

You will need to assume superuser privileges by using either su or sudo

```
sudo echo "ben" >> /etc/cron.deny
```

12.1.4.3 Example Black/Whitelisting File

Whether you are creating /etc/cron.allow or /etc/cron.deny the syntax is the same. Simply enter the usernames that you wish to allow/block on a single line separated by spaces

```
ben john gary
```

12.1.5 Changing your Crontab Editor

On a number of systems, the default crontab editor is *vim*. As powerful as this editor may be, it's simply too complex for some users. This section will show you how to set nano as your default editor

```
sudo nano /etc/profile
# At the bottom of the file add
export EDITOR=nano
#Ctrl-X followed by Y to save
source /etc/profile #this'll make the changes apply
```

Whenever you log in the default editor will now be set to nano!

12.2 At

The *at* command allows you to instruct the system to run a command at a particular time. It's generally used to schedule a one-off command.

12.2.1 Basic Syntax

At's basic syntax is fairly straightforward. You specify the commands you want to run in a text file and then direct them into *at* whilst specifying time (and if necessary date).

```
echo "echo hello world" > ~/test_at
at 1pm < ~/test_at
```

Or to specify a date

```
echo "echo hello world" > ~/test_at
at 1pm 01/03/2013 < ~/test_at #run the task on the 3 Jan 2013
```

12.2.1.1 Time Formats

At will accept time in a number of formats.

Format	Description	Example
HH:MM	*Standard time (24 hour)*	*13:40*
AM/PM	*Standard time (12 hour)*	*01:30pm*
midnight	*Run at midnight*	*midnight*
noon	*Run at 12 Noon*	*noon*
teatime	*Run at 4PM*	*teatime*
now	*Run Now*	*now*

12.2.1.2 Date Formats

At will also allow you to specify dates, otherwise you'd need to log into the system each day to schedule a command.

Format	Description	Example
m/d/y	*Standard (American) date format*	*01/31/2012*
today	*Sets the date for today*	*today*
tomorrow	*Sets the date for tomorrow*	*tomorrow*

12.2.2 Viewing Queued *At* Jobs

With users potentially being able to queue any job, it's important to know how to find out exactly what's been queued. Enter *atq*! To see jobs that you've queued just run it as your user, to see jobs that everyone has queued run as root!

```
sudo atq
6 Sat Jan 12 13:15:00 2013 test_ at ben
2 Mon Mar 5 13:42:00 2012 test_ at ben
5 Mon Mar 5 13:15:00 2012 test_ at ben
```

The output displays when the command will run as well as who scheduled it, the first column in the list identifies the *at* queue number which will come in useful when we delete the jobs.

12.2.3 Deleting Queued *At* Jobs

If we've identified a job that shouldn't be there (either because it was added in error, or because a user is misbehaving) we can remove the job using *atrm*. To do so we need to the queue number for that particular job, which we obtained by using *atq* (in my example it's queue number 6, yours will differ!)

```
atrm 6
```

One command and the job is removed!

12.2.4 Controlling who can use *At*

There may be situations that require you to control who can schedule tasks using *at*. Just as with cron, you can use a blacklist or a whitelist.

12.2.4.1 Whitelisting Users

To whitelist users create the file /etc/at.allow. Each username should be listed on a separate line, but you will need to assume superuser privileges to create/edit the file using su or sudo

```
echo "ben" >> /etc/at.allow #Add ben to the allow list
echo "john" >> /etc/at.allow #Add john to the allow list
```

Whitelisting is generally considered a more secure way of implementing security and requires far less effort to maintain than a comprehensive blacklist.

12.2.4.2 Blacklisting Users

If you have a single user that is misusing *at*, you can block them by creating and adding them to the blacklist /etc/at.deny

```
echo "ben" >> /etc/at.deny #Add ben to the blacklist
```

12.2.4.3 Example Black/Whitelist file

The syntax for both the blacklist (/etc/at.deny) and the Whitelist (/etc/at.allow) is identical. Simply enter the username of the user you wish to block/allow on each line

```
ben
john
gary
```

Chapter 13

Malware Checks

There will always be those who claim that you don't need AntiVirus on Linux (or any other non-Windows® system). This really is a fallacy as even if the malware can't actually effect our system it'd be wrong to knowingly risk passing malware onto systems that can be infected. Depending on the needs of your business you may wish to consider a commercial anti-malware systems, but the Open source options available are also very effective.

In this chapter we'll look at how to install, configure and use ClamAV and RKHunter. We'll be installing each from source as malware software is something we want to be absolutely up-to-date (meaning the repo's are not the best source!)

13.1 ClamAV

ClamAV is an Open Source antivirus system and is installed by default on some distros. Whilst there are GUI's available, ClamAV is usually triggered from the shell, especially when dealing with servers.

In this section we will both install and use ClamAV to help secure our server.

13.1.1 Installation

The easiest way to find the latest stable release is to browse to http://www.clamav.net/lang/en/download/s
from a PC. Towards the top of the page is the line "*Latest Stable Release*", hover your mouse over the link to see the URL it is pointing to (this should look similar to http://freshmeat.net/urls/c9bfa0aa2a4b8f3dc21e37debf0b05e5).

On the console of your Linux system run the following commands

```
sudo -s
cd /usr/local/src
wget "http://freshmeat.net/urls/c9bfa0aa2a4b8f3dc21e37debf0b05e5" -O clamav.tar.gz
tar xvzf clamav.tar.gz
cd clamav #Press tab after typing clamav to autocomplete
./configure
make
make install
freshclam # Updates the Virus Database
exit #Exit sudo to assume normal user privileges again
```

13.1.2 Using ClamAV

Running *clamscan –help* will output a list of the available options, however we'll look at the most common uses.

To run a system wide scan (needs to be run using root privileges), run the following command

```
sudo clamscan –recursive -i /
```

To scan files in a specific folder you can run the following command

```
sudo clamscan -i mydir/* #This will not recurse into directories below
sudo clamscan –recursive -i mydir #This will scan any subdirectories of 'mydir'
```

These will report any suspicious files found. Where files are listed as PUA[1] this does not necessarily mean that a virus has been found. PUA's are applications that may have a legitimate use but that you may not necessarily want. Unfortunately in the case of these you need to make an assessment on a case-by-case basis.

Similarly, if you are running a webserver you will probably receive warnings about script.packed. Where these are scripts that should be there, they can be ignored. If, however, you think a script should not be there you should move (not remove) it until you are sure[2]

13.1.2.1 Useful Command Line Flags

There are, in reality, far too many ClamAV flags to remember and whilst –help will list them it can quickly become irritating to scroll up and down as you build your command. For reference, a list of the most useful flags has been included below.

Note: You should always follow flags with the location you want to scan (e.g. to scan a directory called '*mydir*' in the current directory simply specify '*mydir*')

[1] PUA - Possibly Unwanted Application
[2] At which point you can use *rm* to get rid of that pesky file!

Flag	Description
-v	Be verbose (Output extra information)
–stdout	Direct all info to stdout[3]
-i	Only display infected files
-r	Recurse through subdirectories
–remove	Should infected files be removed? Be very wary of using this as false positives will be removed as well
–move	A directory to move infected files to (directory must already exist)
–exclude	Exclude files matching the value of this
–detect-pua	Detect Possibly Unwanted Applications (defaults to No)
–scan-mail	Scan emails in Inboxes (defaults to Yes)
–phishing-scan-urls	Check for known phishing URLs (defaults to Yes)

13.1.3 Scheduling Scans Using Cron

In this section we are going to schedule a weekly scan using cron and have the results of each scan emailed to use at myname@myemail.com[4].

So, we need to add the command to root's crontab

```
sudo crontab -e #Open the crontab
0 0 * * 0 clamscan –stdout -i –detect-pua=yes –recursive / | mail -s “Antivirus Scan Results” myname@myemail.com
# Now save and exit
```

Of course, it would also be wise to update the virus database before we run our scan. So we'll add a new cronjob to run freshclam late on Saturday night (as our scan runs at midnight on Sunday).

```
sudo crontab -e
45 23 * * 6 freshclam
#Now save and exit
```

In a few simple steps we've set up a weekly scan of the whole system. Depending on the needs of your business you may wish to limit full scans to once a month and simply scan other areas (e.g. the webserver document root) weekly.

[4] Obviously you need to replace this with your own email address!

13.2 RKHunter

RKHunter is slightly different to ClamAV in that it looks for rootkits on your system. Although a rootkit can be classified as malware, it's something that's specifically aimed at allowing an attacker root level access to your system. They can be difficult to detect, and so checks need to be run regularly.

13.2.1 Installing RKHunter

As with ClamAV, the easiest way to locate the URL used to download the latest version of RKHunter is to use a webbrowser on another PC. Browse to http://sourceforge.net/projects/rkhunter/files/ and hover over the link just beside "*Looking for the latest version?*" so that you can make a note of the URL to use[5].

Now to your Linux console!

```
sudo -s
cd /usr/local/src
wget -O rkhunter.tar.gz "http://sourceforge.net/projects/rkhunter/files/latest/download?source=files"
#Replace with your URL
tar xvzf rkhunter.tar.gz
cd rkhunter # press tab after typing rkhunter to auto-complete
./installer.sh –install
rkhunter –update #Update the definition files
exit
```

Now we need to run an additional command, but you should *only* ever run this when you are confident that all files can be trusted (i.e. it's very wise to run a ClamAV scan first!)

```
sudo rkhunter –propupd
```

This command informs *rkhunter* that it should note details about key files for comparison later.

13.2.2 Running RKHunter

RKHunter will default to checking your entire system (which is exactly what we want), so there's no need to specify a path. The default settings will require you to manually interact with the system, so we are going to pass some flags to override that.

```
sudo rkhunter -c –sk #Check the system and skip keypress sections
```

[5] It should look similar to http://sourceforge.net/projects/rkhunter/files/latest/download?source=files

13.2.2.1 Useful Flags

As with ClamAV you can list all available options by using the *–help* flag. For reference, a list of the most useful flags has been listed below

Flag	Description
-c	Check the system
–cronjob	Disable colours, check the system and skip keypresses
–ns	Don't display the summary at the end of the scan
–rwo	Only show warning messages
–sk	Skip the keypress requirement
–update	Update the definitions file
–propupd	Update the local file checksums

13.2.3 Scheduling Checks using Cron

In this section we are going to add a cronjob that will run a daily scan and email the results to us at myname@mymail.com[6]. Ideally, we'd be able to simply add our rkhunter command to the crontab in the way we did with ClamAV. Unfortunately this doesn't seem to work reliably, so we need to create a small shellscript that will do what we need and then add that.

```
sudo nano /root/rkhunterrun.sh
#!/bin/bash
rkhunter –cronjob > /tmp/rkhunter.log
cat /tmp/rkhunter.log | mail -s "RKHunter Scan Result" myname@mymail.com
rm -f /tmp/rkhunter.log
# Save and exit
```

Next we'll make the file executable and add it to our crontab

```
sudo chmod +x /root/rkhunterrun.sh
sudo crontab -e
30 0 * * * /root/rkhunterrun.sh
# Save and exit
```

This scan will run daily at 30 minutes past midnight. We probably want to run an update beforehand though (we could of course just enter the command into our shell script instead of adding a second cronjob)

```
sudo crontab -e
0 0 * * * rkhunter –update
#Save and exit
```

[6] You should, of course, replace this with your own email address!

In a few simple steps we've now enabled a full daily scan with results emailed to us!

13.2.4 Regular Maintenance

One of the checks RKHunter does is to look and see if any of the key files on the system have changed. This does mean, however, that following a system update you will receive a number of warnings about changed files. You can resolve this by running the following command, however you *must only* do this if you are sure that the system is secure!

```
sudo rkhunter –propupd
```

13.3 Principles of Server Security - Recovering from Compromise

One of the core things a SysAdmin understands is that once a system has been compromised it should not be trusted. From a business point of view it's far more desirable to avoid downtime by 'fixing' issues than it is to perform a complete re-install/re-image of the system.

However, once a system has been compromised it becomes very difficult to say for certain that the system has been completely re-secured. A good SysAdmin will make an evaluation based on the severity of the compromise, the potential risks posed by not re-imaging and the time it will take to 'fix' the issue.

The basic principle is that if there's a potential to increase the attack surface, we need to mitigate.

We will look at recovering from compromise in more depth in a later part of this book. However, this section will examine the basic principles that a SysAdmin will use when evaluating the repercussions of a malware infection.

13.3.1 Low Risk Compromise

Some examples of compromise are viewed as low risk. It's possible that we may have found malware on our system, but can accept that it does not pose an ongoing risk to our server[7].

So for example, where Windows specific malware has been found in a users mailbox we would simply remove the offending file. There'd be absolutely no reason to re-image our server, because we can evaluate and answer the following

[7] Not that we wouldn't remove it!

Question	Answer	Risk
How did the malware get onto our server	*Emailed in*	*L*
Has the malware executed on our server	*No (Windows Specific)*	*L*
Is there potential for backdoors?	*No (Not Executed)*	*L*
Final Assessment	***Low Risk***	

13.3.2 High Risk Compromise

If we had discovered a rootkit on our system then we would consider it high risk, regardless of whether or not we know how it came to be on our system, the very nature of a rootkit means that it's almost impossible to say for sure that we have fully disabled it.

We'd make the following assessments

Question	Answer	Risk
How did the malware get onto our server	*Unknown*	*H*
Has the malware executed on our server	*Yes*	*H*
Is there potential for backdoors	*Yes*	*H*
Did the malware execute with superuser privileges	*Yes*	*H*
Has the malware been fully removed	*Unknown*	*M*
Final Assessment	***High Risk***	

We'd therefore have no choice but to re-image the server to ensure that all attack vectors that may have been opened were closed. Of course, we'd also need to investigate how the malware came to be on our system as that is clearly a known attack vector.

Chapter 14

User Management

One of the most common tasks a Sysadmin is likely to perform is the managing of user accounts. Some types of servers won't need so much effort in this area as ordinary users won't need to authenticate (Web servers being a good example). However, it is important to understand how users and groups are managed, especially when investigating a possible compromise.

14.1 Adding a User

A server should always have at least one (non-system) user other than root. The principles of good security say that you should only have higher privileges when you *actually* need them. In other words, you should only login as root when it's essential to do so. A good SysAdmin will use a non-privileged account most of the time, and may in fact never login as root directly (by using *su* or *sudo* instead).

Let's add a new user with the following information

Detail	**Value**
Real name	*Geoffrey Jones*
Desired Username	*geoff*
Home Directory	*/home/geoff*
Shell	*/bin/bash*
Group	*users*

The first two values are self-explanatory, so what exactly are we doing with the final three? The user's home directory is where all their files will be stored. It also may be the only area that they can edit/create files depending on group

membership and the privileges we assign them. Generally, if you fail to define a home directory the system will create /home/{username} so in this instance this could be left blank.

When a user connects to the console, a shell will be called. We can set this to a range of values including */bin/false*[1]. In this case we are setting the users default shell to the Bourne Again SHell (BASH).

A user will always be part of a group, sometimes that group will simply be the same as their username (and they'll be the only member!). We are going to add the user to the group 'users'.

So to create this user we run the following command

```
sudo -s
adduser –home "/home/geoff/" –shell "/bin/bash" –ingroup "users" geoff
# Set a new password and confirm when prompted
# Enter the users Full Name
# Enter any other information you wish to add, or just press enter to accept the default
# Ensure you confirm the information is correct
exit # We don't need root privileges any longer
```

We've now added our new user and Geoffrey should be able to log-in using the password you've just set! We can view the results by running the following command

```
cat /etc/passwd | grep geoff
geoff:x:1003:100:Geoffrey Jones,,,:/home/geoff/:/bin/bash
```

By way of explanation, the 1003 in the above result is the UID for that user. Most system-users[2] will have a UID of less than 500. There are, however, some exceptions!

The 100 represents the GID of the users primary group. We'll be adding additional groups later in this chapter.

14.2 Adding a Group

We may wish to create a new group so that we can allow multiple users permission to access a certain file or resource. There may of course already be a suitable group to use, however if there isn't we will need to add one.

For the purposes of this example, we are going to create a group called "editors"

```
sudo -s
addgroup editors
exit # We don't need to be root any longer
```

[1] Used to deny shell access, although it should not be the only defence!

[2] Generally added to allow a specific piece of software to run. Login is generally disabled for these users

We'd then need to make our new group the owner of whichever file we wish to grant access to[3].

As the group is currently empty, we'll now look at adding our user to this group.

14.3 Assigning a User to a Group

We now wish to add an existing user to our new group. To do so we will use the command *usermod*.

```
sudo -s
usermod -a -G editors geoff
exit
```

We can now confirm whether this was successful by running the following command

```
sudo groups geoff
geoff : users editors
```

You should be aware, however, that changes in group membership generally only take effect the next time a user logs in,

14.4 Allowing a user to use *su*

We've already looked at how to add a user to a group, enabling a user to use *su* simply involves adding the relevant user to the correct group. In most cases this will be '*wheel*'. So to allow geoff to use *su* we would simply issue the following command[4]

```
sudo usermod -a -G wheel geoff
```

14.5 Allowing a user to use *sudo*

We may wish to allow Geoff to use the *sudo* command. As he is not currently a sudoer, any attempts to use sudo will return the error "*geoff is not in the sudoers file. This incident will be reported.*" and an entry will appear in the system log showing that *geoff* attempted to use *sudo*.

[3] As we saw earlier in this book, this is achieved using either *chown* or *chgrp*

[4] If you receive "*usermod: group 'wheel' does not exist*" this simply means that your system either doesn't have su installed or uses a different group. Skip onto allowing a user to use *sudo*.

There are two ways in which we can enable sudo for geoff. We can either give him privileges to run anything using *sudo*, or limit him to certain commands (the latter is generally the more secure way).

You will only be able to complete these steps if *sudo* is installed[5]!

14.5.1 Limited *sudo* privileges

All sudo control is achieved through the sudoers file, usually */etc/sudoers*. We are going to enable a very basic set of commands for geoff - *ls* and *cat*.

```
sudo -s
nano /etc/sudoers
# Add the following line
geoff ALL=(ALL) /bin/ls, /bin/cat
# Save and close
```

Geoff can now use sudo to run *ls* and *cat*, but nothing else. Attempts to run another command using *sudo* will return an error stating that he lacks the privileges.

To remove the privileges, simply remove the line that we just added.

Give careful consideration to which commands you allow a user to run, as they may well be able to use them to circumvent your security. Allowing a user to run *nano* with *sudo* may seem perfectly harmless, but they can then utilise this to edit the sudoers file and grant themselves more privileges!

14.5.2 All *sudo* privileges

To allow a user to run any command using sudo, we grant them sudo all privileges. This needs very careful consideration first, however, as you will in essence be granting that user full root level access!

We simply need to add an appropriate line to /etc/sudoers

```
sudo -s
nano /etc/sudoers
# Add the following line
geoff ALL=(ALL) ALL
#Save and close
```

Geoff can now run any command using sudo.

To deprive him of this ability we need only remove the line we just added.

[5] Most distributions now have *sudo* installed by default

14.5.3 Using a Group

We don't have to list each individual user in /etc/sudoers, we can instead allow members of a specific group to use sudo[6]. To do so we add a line almost exactly the same as those we added for Geoff, but prefix the name with a percentage sign to mark that it represents a group.

As Geoff is a member of the *editors* group, we'll grant *sudo* all privileges to that group.

```
sudo -s
nano /etc/sudoers
# Add the following line
%editors ALL=(ALL) ALL
# Save and close
```

14.6 Removing a User from a Group

As we don't actually want geoff to be able to use *su*, we need to remove him from the relevant group. To do so we need to view his groups and decide which we want him to stay in

```
sudo -s
groups geoff
geoff : users wheel editors
# We want to keep Geoff in users and editors
usermod -G users, editors geoff
# Confirm whether it worked
groups geoff
geoff : users editors
```

Geoff has now been removed from the group 'wheel' and will be unable to use su.

14.7 Resetting a users password

To reset a users password, assume root privileges (using sudo if necessary) and run the following command

```
passwd {username} # i.e. passwd ben
# Enter New Password
# Confirm New Password
```

The users password has now been reset to whatever you entered when prompted.

[6] Which is, incidentally, how a default Ubuntu installation is configured - using group admin

14.8 Temporarily Disabling a Users Account

There may be occasions where we want to disable a users account without actually deleting it[7]. We can acheive this with the *passwd* command.

We'll lock and then unlock Geoff's account

```
# To lock the account
passwd -l geoff
passwd: password expiry information changed.
# Now we'll unlock it
passwd -u geoff
passwd: password expiry information changed.
```

It's good practice to do this when an account won't be needed for a while as it ensures that others cannot try and quietly use the account.

14.9 Removing a user

Rather than having unnecessary user accounts on the server, we should always remove an account when it's no longer needed[8]. To do so we use the command *userdel*. We'll assume that Geoff has left the business, and remove his account

```
sudo -s
userdel geoff # Don't run this yet!
exit
```

Geoff's username has now been removed, but his home directory and mail folders have been left in place. We could of course remove these manually, but if we are sure that those items will no longer be required we can run the following command instead

```
sudo -s
userdel -r geoff
exit
```

14.10 Removing a group

We may also opt to remove a group, perhaps because it is no longer needed. Doing so is no more complex than removing a user simply requiring us to run the following to remove the group '*editors*'.

[7] For example, when an employee is on a Long Term absence
[8] Even if we simply lock it for 6 months and then remove after that.

```
sudo -s
groupdel editors
exit
```

There is, however, one caveat. You cannot delete a group that is currently set as the primary group for a user. You first need to assign any relevant users to another group using usermod, and then can delete the group.

Chapter 15

Controlling Services

Even on a Desktop Linux system there are a number of services running, more so on a server! The most common related services are MySQL, Apache, Postfix/Qmail and perhaps even a FTP server such as ProFTPD.

Sometimes we need to restart these services, whether because they have stopped responding or because we need to force them to load a new configuration. We may also need to start a service if it isn't configured to start when the system boots or perhaps because it has stopped of it's own accord.

It's rare that we'd actually need to stop a service, unless we are explicitly trying to prevent anyone from using it for a short time.

In this chapter we will examine the two common methods of achieving this and will also examine how we configure a service to run at boot. To acheive all this, we first need to understand what is actually responsible for achieving this.

Because we are controlling system services, superuser privileges are required. Every command listed in this chapter should therefore be prefixed with *sudo* or run using *su*.

Note: If you are running a Gentoo based system, things work slightly differently to the information contained in this chapter. A quick reference has been provided for Gentoo users in the final section.

15.1 Service control - Init Scripts

Services that begin at startup are called by a dedicated script - known as an init script. Any software that can be started using an init script will usually have one in */etc/init.d*. Existence of a relevant init script does not necessarily imply that the service will start at boot however, as we first need to activate it.

An init script is nothing more than a standard shell script containing a range of functions intended to start, stop and restart the relevant service. Some may also support other commands (such as zap or reload) but the basic rule of thumb is *stop*, *start* and *restart*.

As we will see in later sections, to issue a command to an init script you simply execute the script with your desired action as the argument

```
/etc/init.d/mysql restart
```

15.1.1 Finding out which arguments are supported

Whether you opt to use the service command or call the init script directly, finding out which arguments are supported is simply a case of calling without *any* arguments.

```
/etc/init.d/apache2
* Usage: /etc/init.d/apache2 {start|stop|restart|reload|force-reload|start-htcacheclean|stop-
htcacheclean|status}
service apache2
* Usage: /etc/init.d/apache2 {start|stop|restart|reload|force-reload|start-htcacheclean|stop-
htcacheclean|status}
```

15.2 Non-Gentoo Systems

15.2.1 Direct calls to Init Scripts

Historically, all services were controlled by making a direct call to the relevant init script. At time of writing all common systems still support this[1]. Doing so really couldn't be simpler as it's identical to the way in which we'd normally run a command.

15.2.1.1 Starting a Service

To start a service we simply call the relevant init script and use the argument start. So to start MySQL we'd run the following

```
/etc/init.d/mysql start
```

[1] Although many will now suggest you use the newer *service* command

15.2.1.2 Stopping a Service

To stop a service we call the relevant init script with the argument stop. Having started MySQL we'd then run

```
/etc/init.d/mysql stop
```

15.2.1.3 Restart a Service

To restart a service (perhaps because we need it to reload it's configuration), we use the restart argument

```
/etc/init.d/mysql restart
```

15.2.2 *Service* command

The *service* command is often in use on newer systems. Whilst it doesn't prevent you from using the init scripts directly, if you do so you will receive a suggestion that you utilise *service*. It does save some typing as there's no need to enter a path to the init script. Longer term it is better to utilise the *service* command as the location of init scripts may change at some point in the future.

Again, it accepts simple arguments - the first being the name of the service the second the action you wish to perform.

If you forget the name of a service you can always list the contents of */etc/init.d* to look it up - of course, at this point you may as well run the init script directly.

15.2.2.1 Starting a Service

To start a service we simply need to use the argument start. So to start MySQL by using the service command we would run

```
service mysql start
```

15.2.2.2 Stopping a Service

To stop a service we use the argument stop. So having started MySQL we'd then run

```
service mysql stop
```

15.2.2.3 Restarting a Service

To restart a service we use the argument restart.

```
service mysql restart
```

15.2.2.4 Notes on Service

As we noted earlier, some init scripts will support additional arguments. Generally the service command will accept these, but there may be some occasions where it will not. In these instances the resolution is to directly call the init script itself. So we may encounter the following scenario

```
service apache someunusualargument
# Service reports failure
/etc/init.d/apache someunusualargument
```

It's very rare for this to occur however.

15.2.3 Enabling/Disabling Services

Generally speaking Init scripts are enabled by adding them to a folder dedicated to the relevant runlevel. A full explanation of runlevels falls somewhat outside the scope of this book, but we do need to know how to identify which runlevel to enable an init script for.

15.2.3.1 Finding the default runlevel

In general if we are configuring a service to run at boot, we want it to run in the default runlevel. This means that whenever the system boots normally our desired service should also start.

To find information on the current runlevel we simply run

```
runlevel
# The second number outputted is the current runlevel. The first is the previous
```

15.2.3.2 Listing Enabled Services

We may find that the service we wish to enable is in fact already enabled. In order to check we need to look at which init scripts are actually enabled. To do so, we simply need to list any files in the directory /etc/rc{runlevel}.d.

So to check enabled services for runlevel 2 we would execute

```
ls /etc/rc2.d/
```

Many of the returned results will usually be prefixed with a number and/or a letter. These are used to control the order in which the scripts are called (useful if a service is dependant on another).

15.2.3.3 Enabling a service for a runlevel

Having looked up the default runlevel we can now proceed to enable an init script for it. We enable init scripts by adding them in /etc/rc{runlevel}.d. So if our default runlevel is 2 we would add an entry to /etc/rc2.d/

We don't actually need to copy the init script into the directory[2], instead we create a symbolic link using the *ln* command.

So to enable the init script /etc/init.d/apache2 for runlevel 2 we would run the following command

```
ln -s /etc/init.d/apache2 /etc/rc2.d/s99apache2
```

We've prefixed apache2 with S99 to ensure that it is one of the last started services. The service Apache2 will now start whenever the system boots into runlevel 2.

15.2.3.4 Disabling a service for a runlevel

There are times when we need to disable a service. This is usually because it is no longer required, but may also be a precautionary measure whilst an critical issue is addressed.

To do so we simply need to remove the relevant symbolic link from the runlevels directory.

So to remove apache2 from runlevel 2 we simply run

```
rm /etc/rc2.d/s99apache2
```

15.3 Gentoo Based Systems

Although many of the basic fundamentals on Gentoo remain the same, there are some minor differences. The largest difference being the method for enabling or disabling a service.

[2] This is in fact a *very* bad idea as any updates to the init script won't take effect!

15.3.1 Direct Calls to Init Scripts

15.3.1.1 Starting a Service

As with other distribution types, the init scripts for all system services are usually in */etc/init.d*. We can run these directly as follows

```
/etc/init.d/mysql start
```

15.3.1.2 Stopping a Service

Stopping a service by directly calling the init script simply involves calling the init script with the stop argument.

```
/etc/init.d/mysql stop
```

15.3.2 Using the rc-service command

On non Gentoo systems we can use the *service* command to control services. On Gentoo, the command differs but the principle remains exactly the same. This is also the preferred method of controlling services as it ensures consistency if the location of init scripts does change at some point in the future

15.3.2.1 Starting a Service

To start a service using the *rc-service* command, we simply specify the name of the service and the action (start)

```
rc-service mysql start
```

15.3.2.2 Stopping a Service

To stop a service using the *rc-service* command, we again specify the name of the service and the desired action

```
rc-service mysql stop
```

15.3.3 Enabling/Disabling Services

The difference between Gentoo based systems and non-Gentoo systems should become immediately apparent in this section. Whereas we might create a symlink in /etc/rc2.d on a RedHat system, on a Gentoo based system we use the *rc-update* command to effectively create a symlink for us.

15.3.3.1 Finding Runlevels

To begin, we need to find out the names of the runlevels configured on the system. In many ways, this is far easier for the average user than on non-Gentoo systems. On Gentoo systems the runlevel is given a name such as 'default' rather than relying on the numbering schema used by other systems.

To view the available runlevels we simply run

```
ls /etc/runlevels
```

This should return the name of a number of runlevels, on a default install you will most likely see

- ▷ boot
- ▷ default
- ▷ nonetwork
- ▷ single

Each of these is actually a directory, containing symbolic links to the relevant init script. However, rather than manually adding links we can simply tell the system to do it for us.

15.3.3.2 Listing Enabled Services

To list all enabled services for a given runlevel, we use the show argument with *rc-update*

```
rc-update show runlevel
```

So to list all services enabled for the default runlevel, we would run

```
rc-update show default
```

15.3.3.3 Enabling Services

To enable the services we simply tell *rc-update* to add the service.

```
rc-update add initscript runlevel
```

So to add MySQL to the default runlevel

```
rc-update add mysql default
```

MySQL will then be started whenever the default runlevel begins.

15.3.3.4 Disabling Services

To disable a service, we simply tell *rc-update* to delete it from the desired runlevel

```
rc-update del initscript runlevel
```

So to disable MySQL from the default runlevel we would run

```
rc-update del mysql default
```

Chapter 16

Basic Firewall Rules

It's very important to get firewall rules exactly right, otherwise traffic that you don't want to be blocked will be or traffic that you do want to block will still get through. It's therefore essential that a SysAdmin understands exactly what the command they are about to issue will do, and the potential effects.

It's also important to consider how you are interacting with the server. If you are connecting in remotely using SSH, you need to think very carefully about adding any rule that may potentially prevent you connecting in this manner. If you do accidentally block all SSH traffic, how will you connect to the server to remove the rule?

Although there are other firewall solutions available, many Linux distributions ship with *iptables* as default. We'll therefore be interacting with the firewall using this command.

Within this chapter we'll be looking at

- ▷ IPChains Based Firewalls - The Basics
- ▷ Iptables Command Syntax
- ▷ Adding a rule
- ▷ Removing a rule
- ▷ Blocking Specific Hosts
- ▷ Blocking Specific Services
- ▷ Useful Commands

As the firewall is a system wide resource, every command in this chapter will require root privileges. You'll therefore need to prefix each with *sudo* or run *su* (or *sudo -s*) before commencing.

16.1 IPChains Based Firewalls - The Basics

To the average user, the topic of how a firewall works is generally very complicated. With that in mind, this section will simply give a basic oversight of the process a chains based firewall follows when deciding to allow or deny a connection.

Generally speaking, there will be three chains

- ▷ Input
- ▷ Forward
- ▷ Output

You can, of course, add more where necessary[1].

Within each chain we specify a number of rules, the system works through the rules in the relevant chain from top to bottom until it finds one that matches. If no matching rule is found, the system applies the default action (which is normally to allow the connection).

So assuming the following rule-set on the Input chain

Target	***Prot***	***Opt***	***Source***	***Destination***	
DROP	*tcp*	–	*0.0.0.0*	*anywhere*	*tcp dpt:smtp*

Any TCP connections coming into the server on port 25 will be blocked (Port 25 is the SMTP port). Connections to any other port will be allowed.

We can also assign similar rules to the Output chain to prevent our server from connecting to Port 25 on another system[2].

Be aware that where DROP has been used, the connecting system will not receive an error. The connection will simply time out; there is a method for configuring the firewall to respond and say that the connection has been blocked, but this can be helpful to an attacker and so will not be explored here.

16.2 Iptables Command Syntax

The command *iptables* will accept a wide variety of command line arguments. Many of these fall outside the scope of this book, so we'll simply explore the basic syntax that will be needed within this chapter.

The command requires the following syntax

[1] Although this falls outside the scope of this book.
[2] Though this would prevent the server from communicating with mail servers!

```
iptables -[ADI] chain rule-specification [options]
```

To append a rule, we begin with *-A*, whereas to remove a rule we being with -D. The chain will usually be INPUT or OUTPUT.

The rule specification is where we can use a variety of flags to define how the rule will work, the most common flags are

Flag	***Description***
-s	*The source IP (i.e. who's connecting)*
-p	*The protocol (tcp/udp/icmp are the most common)*
-d	*The destination address*
-i	*Interface. We use this if we have multiple network cards, but don't want the rule to apply to all*
-j	*Jump - Which action to jump to (i.e. ACCEPT or DROP)*
–dport	*Used with -p to specify a specific port*

16.3 Viewing Rules

To view rules, we simply use the *–list* argument which will return a list of all rules in a standard format. The command can sometimes take a little while to complete

```
iptables –list
Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP tcp – 0.0.0.0 anywhere tcp dpt:smtp

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

We can see from the output above the the default policy is to accept. This means that if a rule isn't matched, all connections will be allowed. Whether or not this is suitable depends entirely on the use of the server, it's a wise rule to use for any system where you cannot guarantee that connections will come from specific IP addresses and/or using specific ports and protocols.

Where this can be guaranteed, however, it may be more sensible to use a default policy of DROP. Although this can be changed, the easiest way is simply to ensure that the last rule in the chain is a global DROP statement as all permitted connections will have already matched a rule.

16.4 Adding a rule

In this section we'll look at adding rules. There are two main ways in which we can do this, we can either insert a rule at the top of the chain (so it's the first rule checked) or Append to the end of the chain so that it's the final rule checked[3].

Which you require depends on what the rule is intended to do. For example, if you are adding a rule that will drop all connections (so that the default policy becomes DROP), you'll want to append the rule. Otherwise, all connections will be dropped regardless of any rules that may have allowed some of those connections through.

Most users will always use Append unless one of the following circumstances applies

- ▷ There's a need for the rule to be the first processed
- ▷ There's already a global 'catch-all' rule at the bottom of the chain, so we need to insert the rule before that

In either case, the syntax remains more or less the same. The only change is in one command line flag, as we'll see below.

16.4.1 Inserting

To insert a rule at the top of the chain we use the *-I* argument. For our example we'll insert a rule that blocks all traffic to port 25

```
iptables -I INPUT -s 0.0.0.0 -p tcp –dport 25 -j DROP
```

The arguments used here break down as follows

Argument	***Value***	***Explanation***
-I		*Insert the rule at the top of the Chain*
INPUT		*The chain to update*
-s	*0.0.0.0*	*The source IP (in this case, any IP)*
-p	*tcp*	*The protocol to block*
–dport	*25*	*The port the client is trying to connect to*
-j	*DROP*	*The action to jump to (Effectively, block the connection)*

Any system trying to connect to our system on port 25 will now be unable to do so. Running *iptables –list* will now show our new rule in the INPUT chain

[3] In reality, we can also add rules in any position we need, but for most needs this is unnecessary

16.4.2 Appending

To append a rule we use the *-A* argument, the syntax remains the same as insertion.

```
iptables -A INPUT -s 0.0.0.0 -p tcp –dport 25 -j DROP
```

This rule would then be added to the bottom of the chain.

16.5 Removing a rule

To remove a rule we simply need to use the *-D* argument and specify as much information as we can to identify the rule. So having added a rule to block traffic to port 25, we'll now remove it

```
iptables -D INPUT -s 0.0.0.0 -p tcp –dport 25 -j DROP
```

16.6 Blocking Specific Hosts

To block a specific IP we simply specify that IP in our command. So to block the IP 211.51.63.4 from accessing any port we would run the following command

```
iptables -A INPUT -s 211.51.63.4 -j DROP
```

Running *iptables –list* should now show the following rule

```
target prot opt source destination
DROP all – 211.51.63.4 anywhere
```

16.7 Blocking Specific Services

We explored blocking specific services in our earlier examples. To block access to a specific service for all users, we would add a rule specifying the relevant port number. So to block FTP (Default is Port 21) we'd run

```
iptables -A INPUT -s 0.0.0.0 -p tcp –dport 21 -j DROP
```

We may, however, wish to block access to FTP for all systems except those we wish to explicitly authorise. This is where the distinction between Appending and Inserting becomes very important. We've just added a global rule to block

FTP to the end of the chain, so lets insert a rule at the beginning of the chain to allow FTP access to a system with the IP 192.168.1.112[4].

```
iptables -I INPUT -s 192.168.1.112 -p tcp –dport 21 -j DROP
```

Now, if a system with the IP 192.168.1.112 attempts to connect to FTP, the connection will be permitted. Any other client, however, will be denied access.

16.8 Clearing all Rules

There may be times when you want to clear all rules from the iptables chains. This could be because the server is being re-purposed without a re-install, or because you want to ensure that the firewall is not the cause of an issue being experienced. To do so, we simply use the *–flush* argument

```
iptables –flush [chainname optional]
```

If you don't specify a chainname, all chains will be cleared. If you specify INPUT then only the input chain will be cleared. So to only clear rules controlling how our server may connect to other systems (i.e. the OUTPUT chain) we'd run

```
iptables –flush OUTPUT
```

This can take some time to complete if there are a lot of rules.

16.9 Saving the current Rule-set

At time of writing, a reboot will generally clear all IPTables rules. However, there is some basic configuration that can be undertaken to ensure that rules persist. In this section we'll be looking at the two main reasons why you may wish to save rules - to allow for backing the rules up and to ensure that rules persist after a reboot.

16.9.1 Taking a backup of Rules

This task is simple to achieve and requires no adjustment to the system unless you wish to automate this step as part of a backup script. To save a rule-set to a file we use the *iptables-save* command and direct it's output to our desired file.

```
iptables-save > /root/myFirewallbackup.fw
```

This will create a save-file in */root/* called MyFirewallbackup

[4] Those who are up-to-date with their RFC reading will know that this is an internal IP address!

16.9.2 Restoring from Backup

There is, of course, little point in backing up rules if you can't then restore them! We simply need to pass the content of the backup file to the *iptables-restore* command.

```
iptables-restore < /root/myFirewallbackup.fw
```

16.9.3 Making rules survive a reboot

To instruct the system to save the iptables configuration we need to run the iptables init script with the argument *save*. This will write the configuration to */etc/sysconfig/iptables*, and will be restored whenever iptables starts.

How you configure the server to save the rules is entirely down to your discretion. Some SysAdmins may want the rules to be saved at shutdown (so that the rules are exactly the same at boot), others may wish to simply save the configuration periodically at the risk of losing a few rules if the system reboots after changes are made, but before they are saved.

To ensure that the rules are saved at shutdown, add a line containing */etc/init.d/iptables save* to the file */etc/rc.local.shutdown*. Or, to run the command regularly, simply create a cronjob to run at the interval you desire.

16.10 Useful Commands

This section contains a few IPTables rules that are used a little more regularly and can be of use to SysAdmins. Each has an explanation of what the rule does, and what it's designed to protect against.

16.10.1 Blocking TCP Timestamp Requests

A useful feature when developing network aware applications is the ability to request and inspect TCP timestamps. However, on a production server this feature is not needed and can potentially allow an attacker to identify how long the server has been running for. This can then be used to help exploit other vulnerabilities that the attacker may have discovered on your server.

We are going to globally disable the timestamps, but then to be absolutely certain we are also going to add a firewall rule to block both the request and the response

```
nano /etc/sysctl.conf
# Find the line net.ipv4.tcp_timestamps = 1 if it exists
# Change to, or create if doesn't exist
net.ipv4.tcp_timestamps = 0
# Save and close

# Now We'll add the rules
#Prevent incoming requests
iptables -A INPUT -p icmp –icmp-type timestamp-request -j DROP

# Prevent Responses
iptables -A OUTPUT -p icmp –icmp-type timestamp-reply -j DROP
```

16.10.2 Controlling Access to SSH

You may wish to control which IP's can connect to your server via SSH. You should only do this if you have at least one client with a static IP[5], and should always ensure that you have at least one other method by which you may adjust the rules if they block legitimate connections.

In order to control access, we are going to add a catch-all rule with rules to permit allowed hosts through. Because the default will be deny, this is in essence a whitelist.

NOTE: If you are managing your server using SSH, follow these steps in the order declared and double-check which IP the server see's for your client.

To begin we whitelist our current client's IP address, for sake of example we'll say the IP is 10.0.0.5

```
iptables -I INPUT -p tcp –dport 22 -s 10.0.0.5 -j ACCEPT
```

We've inserted the rule at the top of the chain to ensure that it is checked before reaching the catch-all rule which we'll now append to the bottom of the chain

```
iptables -A INPUT -p tcp –dport 22 -s 0.0.0.0 -j DROP
```

To whitelist another host, simply run the first command again using that host's IP.

16.10.3 Defending against DoS Attacks

A popular method of achieving a Denial of Service Attack is to send what's known as a SYN flood. In simple terms, when a client connects to a server, it sends a SYN packet. The server will then respond with an ACK in order to

[5] The IP address of many home and business connections changes regularly

establish the connection. By sending a flood of SYN's the attacker is in effect forcing the server to begin establishing connections that will never be used.

To help defend against this a little, we can add a number of rules that limit the number of SYN packets that can be received from each host

```
iptables -A INPUT -p tcp –syn –dport 25 -j ACCEPT
iptables -A INPUT -p tcp –syn -m limit –limit 1/s –limit-burst 4 -j ACCEPT
iptables -A INPUT -p tcp –syn -j DROP
```

16.10.4 Reducing Processing Overhead

So far we've created a number of rules, however each of these will be checked for every packet a client machine sends to the server. This isn't absolutely necessary, as the packets only need to be checked when a new connection is being established[6]. In order to reduce the processing overhead, we can add a rule that will skip rule checking for all established connections.

```
iptables -I INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT
```

This simply tells IP tables that if the connection state is set to ESTABLISHED or if the packet is related to such a connection, the connection should be permitted without further rule checking.

[6] As the connection is port specific: So if you connect to port 80, then changing to Port 22 would still be considered New.

Part V

When Things Go Wrong

Chapter 17

Introduction

An important aspect of what a SysAdmin does is problem solving. This part of the book will help to explain the basic troubleshooting steps that should be undertaken when things go wrong. Given the wide range of potential scenarios, it's important to remember that no book can provide 'cover-all' examples of what you should do. Having an experienced professional on hand is often a must here, but a number of the basic troubleshooting steps are laid out here.

Within this part, we'll be examining

- ▷ Security Compromise
- ▷ Log Files
- ▷ Disk Space Usage

Each chapter will examine the relevant topic, the aim being to highlight the steps a SysAdmin would take as well as passing useful hints where possible.

Chapter 18

Security Compromise

We previously look at how to install and configure malware defenses, however there is more to maintaining security than watching for malware. Whilst your AV and Rootkit scans may return no results, your security could still have been compromised. Whether it's an unauthorised access, or your system is being used as a SPAM relay a compromise of security is something that you should take very, very seriously.

Given the potential repercussions of a compromise, it's absolutely essential that any investigation is undertaken by someone competent and trustworthy. It's strongly advised, therefore, that you should pass the task of investigating a compromise to someone experienced in this area[1]. Whichever route you take, the key is to act quickly.

In this chapter we will look at the common forms of compromise and some of the basic steps that you need to take.

It's important to note, however, that the steps we would take are entirely based on the server and the compromise. This chapter should be taken as a guide and nothing more!

18.1 Unauthorised Access - User Accounts

When an attacker manages to log into an account that they should not have access to, that account is considered compromised. Depending on the rights and roles of that account, we may also need to consider the entire system to be compromised. Whatever the case, however, our starting assumption must always be that the integrity of the entire system is in doubt.

[1] This could be your SysAdmin or you may wish to outsource

Where at all possible, you should prevent any other user from logging in or using the server until you are confident that it is safe to do so. In some cases this may require you to boot the system into a 'rescue mode'.

18.1.1 We don't know which account was used

We may not know which account was used, and depending on the attack[2] it may not be easy to ascertain which account was actually compromised.

The very first step you need to take is to notify all users that their passwords may have been compromised, and that you will be resetting them. This *must* include resetting the root password as we cannot say for certain that root was not compromised.

We now need to reset passwords for every user on the system, so log into your server and run the following commands (we should use a different password for each user, however as a batch job it is far simpler to set a single password).

```
sudo -s
while IFS=: read u x nn rest; do if [ $nn -ge 500 ]; then echo "YOURNEWPASSWORD"
/passwd –stdin $u; fi done < /etc/passwd
# This simply corrects an issue if NFS is installed, ignore any error that might occur
passwd -l nfsnobody
exit
```

This has now reset the password for every normal user. To reset your root password run

```
sudo -s
passwd
#Enter New Password
#Verify New Password
exit
```

We also need to ensure that any SSH keys that users may have used are revoked (a key will allow a user to SSH in without providing a password)

```
sudo -s
find / -name authorized_keys
exit
```

Now work through each of the results and delete those where the file is in a directory called *.ssh*[3]

```
sudo rm -f /path/to/file/.ssh/authorized_keys
```

[2] and the skills of the attacker

[3] Whilst it is possible to instruct find to do this for us, we need to be sure we don't wipe anything important!

All authorised SSH keys should now have been purged preventing keypair logins.

Sadly, this is not likely to have been sufficient just yet. Whilst an important starting point, it's very important that we continue to investigate which user was used. Consider the following questions as they will often lead us to the answers we need

Question	How to Find Out	Notes
What time did the intrusion occur? Were any files created/modified after the initial intrusion	*find / -mtime -4 -print*	Would print a list of files changed in the last 4 days (this may be a lot!)
Were any files created by the intruder? If so what is the ownership on them	*ls -l /path/to/file*	Although the intruder could have changed ownership of the files, it may be that the account used is the one specified as owner.
How did they connect in, does that system keep logs?		If an attacker connected through SSH then you may find that the access was logged. Similarly for other methods

Depending on the attack, we may be able to find other means of identifying which account was used. However, this is an area where experience really comes into play - it's no exaggeration to say that this section could easily be a book in itself.

18.1.2 We Know Which Account Was Used

Life is made slightly easier if we at least know which account was used in the compromise. Rather than resetting passwords and de-authorising ssh keys we *may* be able to simply reset that account.

Whether or not this is appropriate depends entirely on the user account compromised. If the user is able to gain system-wide access (i.e. can use *sudo*) then the attacker has potentially had the ability to tamper with other accounts.

However, to reset the users password and deauthorise any SSH keys we run the following

```
sudo -s
passwd {username}
# Enter a New Password
# Verify the password
# Next we will deauthorise any SSH keys the user has set up
rm -f ~{username}/.ssh/authorized_keys
exit
```

We have now changed the authorisation credentials for that user. The real user will obviously need to be informed of the new password, but don't give them the details just yet!

18.1.3 Privileged account compromise

If a privileged account was compromised (i.e. root or a user with *sudo* ALL privileges) then it becomes time to consider whether the integrity of the system can really be assured. A seasoned attacker may well have patched common commandline tools in order to hide their activities. At this point, the output of *top*, *ps* and any other installed software may not be entirely accurate.

As a rule of thumb, if you've lost control of a privileged account you should *always* re-image the server. Whilst this will inevitable cause downtime, it's impossible to say for sure that the server has been effectively re-secured when an unauthorised user has had full system access.

18.1.4 Checking for Scheduled Jobs

A seasoned attacker may well have scheduled a task designed to re-compromise the server and ensure that they can continue to access your system. There are a number of ways in which they may have implemented this, including cronjobs, *at* jobs and init scripts.

If *rkhunter –propupd* was run sometime before the compromise, the output of a *rkhunter* scan can prove very useful when looking for changed files. Of course, if the attacker compromised a privileged account they may well have run an update command themselves[4].

Another area to be aware of is any script that is run when a user logs in. This varies from server to server but will probably include */etc/profile*, */etc/login* and */home/{user}/.bashrc*.

In this section we'll be checking for scheduled jobs to ensure that the attacker hasn't left a ticking timebomb on our server. The ability to identify potentially malicious scripts comes with experience and knowledge of the server[5]. Of course, if you aren't familiar with the server you can check each scheduled job individually, which whilst providing a more thorough audit is often very time-consuming.

Even if we know exactly which account was compromised, it's best practice to perform server wide checks. Although the user account may be non-privileged, it's not beyond the realms of possibility that the attacker was able to utilise a privilege-escalation vulnerability in order to gain root level access[6].

[4] But as we saw in the last section, if this is the case you should really be re-imaging the server anyway.

[5] One of the reasons why ensuring continuity in your IT department is so important!

[6] More on this shortly

Cron Jobs

In an earlier section we examined the basics of Cron jobs, including how to view all scheduled cronjobs by running the following command

```
sudo -s
for username in $(getent passwd | cut -f1 -d:)
do
echo $username
crontab -u $username -l
done
exit
```

This will have returned a result similar to the following, containg details of all scheduled cronjobs

```
glandwrl 0 3 * * 0 /usr/bin/php5-cli /home/glandwrl/backup.php -profile=#2
```

We would normally proceed to examine the output before moving onto At Jobs. However, for the purposes of brevity, we will examine the all *Cron* and *at* jobs together.

At Jobs

As we saw in an earlier section, viewing all at jobs is far, far easier than viewing all cronjobs. We simply need to run the following

```
sudo atq
```

Now that we have details of all scheduled tasks using both at and cron we can begin to look for anything suspicious.

18.1.4.1 Spotting Suspicious Scripts

We now need to look for any scheduled jobs that should not be there. Being the fulltime SysAdmin of the server becomes very helpful at this stage. For example, if you know that all sites on a webserver are PHP based, it's reasonable to consider any Perl based files as potentially suspicious.

Another potential cause of concern would arise if you recall blacklisting a user from cron but find that they have scheduled cron tasks. This could also potentially be indicative of a Privilege Escalation issue.

If you are unsure as to what a task does, try opening the command in a text editor. In some cases the target file will be a binary[7], but where interpreted

[7] So you won't be able to read it

languages (such as Perl, PHP and BASH) have been used this can prove very helpful.

Be aware that an attacker may well have made a minor adjustment to a script already present on the server and then scheduled that. So although you may believe that *php myfile.php* can be trusted, consider whether or not it's reasonable that this task has been added to a cronjob.

18.1.4.2 Checking Startup Scripts

In this section we will be examining scripts that run automatically as the result of a login/logout or at the server's startup/shutdown. If an attacker has managed to edit system wide files such as init scripts or global login/out scripts, this must be considered a sign on privilege escalation and appropriate action should be taken.

Init Scripts

Init scripts are called when the server first boots, there can be a number of different scripts called based on the runlevel being booted[8]

Viewing init scripts will vary based on the distro being run, but as a general rule of thumb the following command will list all enabled init scripts

```
sudo ls -r /etc/rc*.d
# If you are running a Gentoo based system you'll need to run
sudo rc-update show
```

Most init scripts will be shell scripts[9], and so can be viewed with a text editor such as *nano*. Again, experience and familiarity with the system is the strongest weapon in your arsenal here. You need to feel confident that you can spot init scripts that should not be there, as well as manually reviewing to ensure that minor changes haven't been made.

A failure to spot a change could mean that your system automatically recompromises itself the next time it is restarted. However, if an init script has been added/modified this is a sign that the attacker managed to assume superuser privileges and so the integrity of the system should be held in serious doubt.

Login/Logout Scripts

There are a number of scripts that will be called when a login/out event occurs. These can be customised on a per-server basis so it's important to ensure that

[8] A Linux system has a number of runlevels intended to allow SysAdmins to achieve various tasks. However, a fuller explanation of this falls far outside the scope of this book!

[9] Which may well then call a binary

you know the system well enough to be able to check any additional files. Some of these files are system wide, others can be edited by each individual user to suit their own preferences, both need to be checked to ensure there are no signs of tampering.

Global Files

The first, and most obvious, place to check is */etc/profile*. This file is parsed whenever a user logs in and is usually used to set global variables (such as $PATH and $EDITOR), being a simple shell script there is, of course, nothing to stop an attacker adding their payload to this file.

To check, run

```
sudo nano /etc/profile
```

Some older systems will also have */etc/login.default* and */etc/logout.default*. Both need to be checked, if they exist at all.

User Specific Files

Most users will have some of the following files in their home directory (i.e. */home/{$user}/*)

- .bash_profile
- .profile
- .bashrc
- .login
- .cshrc
- .tcshrc

Each of these files need to be checked, with special attention paid to the home directory of the compromised user account[10]. To do so, run the following command for each user on your system

```
nano ~{$username}/.bash_profile
#e.g. nano ~ben/.bash_profile
```

Don't forget to check for each of the files, as some or all may exist for different users. Check each file for anything suspicious, and thoroughly investigate anything that looks out of place.

[10] Assuming we know which account was compromised

Also keep in mind the name of the compromised account[11], if there is a modification to another users files this may be a sign of multiple compromises, or indeed of privilege escalation.

18.1.5 Finding Changed Files

It's possible to search the file-system for changed files using the *find* command. However, this is only of use if we know when the breach occurred, and will inevitably involve trawling through a vast amount of information. Because the server has been compromised we need to ensure that we are as thorough as possible, but due to the nature of a server files are regularly updated and changed in the course of normal operations.

To find all files modified in the last 7 days run the following command

```
find / -mtime -7 -print > /tmp/fileauditlog
```

In the above command we've directed the output to a file so that we can more easily read through it. The results are likely to number in the tens of thousands, there may even be many more if your server plays host to mailboxes so expect the search to take some time.

Reading through the results will usually take some time, and involves using the same mix of experience and knowledge as we used when searching for malicious cronjobs. You simply need to be able to spot files that either shouldn't be there or shouldn't have been modified.

You can also use *grep* on the file we created in order to exclude some results.

18.1.6 Spotting Signs of Privilege Escalation

We need to consider the possibility that a compromised non-privileged user account has been used to exploit a privilege escalation vulnerability. In a previous section we explored how to allow a user to run commands as root, however where a malicious attacker is involved we do need to consider other possibilities.

Whilst the compromised user account may not have had *su* or *sudo* enabled, because they were able to run some commands on your server it is possible they were able to exploit the system in order to bypass these controls[12]. The means of doing so could be the result of either a documented or an undocumented bug, so rather than dwelling on how, we'll proceed to look at the potential signs of unauthorised Privilege Escalation.

- ▷ Modified init scripts

[11] Assuming we know which
[12] Known in the appropriate circles as 'rooting'

- ▷ Modified files in /etc
- ▷ Modified files in another users home directory.
- ▷ Modified files in /root
- ▷ Modified files in directories owned by other users (but not those granting read/write to all users!)
- ▷ New software installed on a system-wide basis
- ▷ New services running/scheduled to run
- ▷ At/Cron jobs added for other users

There may also be other signs of a Privilege Escalation, the symptoms can be as diverse as the means. If you are in doubt, assume the worst and proceed accordingly. You may be able to test whether or not a user should be able to acheive an action by running *su* as root to switch to a non-privileged account and then attempt that action

```
# Find out if a non-privileged user can write to /etc/init.d
# Run as root (using sudo -s if necessary)
su ben
echo "" >> /etc/init.d/test
bash: /etc/init.d/test: Permission denied # Ordinary user can't do that!
```

If you are unable to perform an action as a non-privileged user then it's reasonable to assume that the attacker has managed to obtain higher privileges.

At this point, your only choice is to consider the integrity of the system to be seriously compromised and to act accordingly[13].

18.2 Malware based compromise

It may be, of course, that a compromise doesn't involve unauthorised access, but instead the execution of malware. This could be in the form of a rootkit, a worm or a Trojan horse. The most common means of infection for any system is via the users themselves.

Prevention is always better than a cure, so IT Departments will generally try to ensure that users understand that they should not be running any software which is even potentially unsafe. In reality, however, the users will generally decide that they know best and will run anything that they are able to!

[13] Usually by re-imaging

More often that not, the first time we will become aware of a malware infection is when our AV defences report one. We may get a report from a user, but this rarely occurs[14].

The steps we take are entirely dependant on the malware itself. If a rootkit has been installed then a re-image is usually a wise idea. We do need, however, to investigate so that we can answer a few simple questions.

Whilst manual investigation is required, additional information can also be gleaned by running a websearch for the malware. Most AntiVirus vendors maintain knowledge-bases relating to different strains of malware - always a useful resource!

18.2.1 First steps

The most important thing is to prevent cross-infection. As we've not investigated the malware yet, we have no idea whether or not it can self-propagate. We therefore have to assume that it can, so we need to take steps to ensure that other servers and users cannot also be infected.

If your server is running as a publicly accessible webserver, this is of the utmost importance because it will be your customers who are infected if the malware has added 'drive-by' downloads to your site(s).

You need to stop users from accessing your server, and where possible may even wish to completely disconnect it from the network pending investigation. Where the server can't be disconnected[15], you may instead wish to stop any services that pose a risk (HTTP server, FTP server etc).

The administrative cost of having to remove malware from other machines following cross-infection will usually vastly outweigh the potential costs of the downtime.

18.2.2 How did it get onto the server?

Regardless of the severity of the discovered malware, it's essential that we try to ascertain quite how it came to be on the server in the first place. It may have been installed onto the system through exploitation of a vulnerability[16]. It's also quite possible, as we've seen, that a user is responsible (deliberately or otherwise).

Whatever the case, we need to be able to take reasonable steps to mitigate the risk of further infection. This may involve updating software, educating users and even a full security audit of the system.

[14] They'll often stay quiet in the hope that the SysAdmin won't find out who ran that piece of code!

[15] For example because you are connecting to it from another location

[16] The Code-Red worm self-propagated amongst Windows® servers in just this manner.

18.2.3 Has the malware actually executed?

This is something we need to consider when our AV Defences report malware. The system will pick up any software matching it's internal signatures without considering whether or not that malware can actually affect our system. Where our server is hosting mailboxes, we may well receive numerous reports of Windows® specific malware. Whilst these aren't wanted, and should be removed, we don't need to consider the system compromised.

If, however, the malware does effect our system, we need to continue investigating.

18.2.4 What does the malware do?

We need to consider what the malware in question actually does so that we can assess the level of compromise. If we've discovered a rootkit then a re-image is almost certainly a necessity.

Unfortunately, if a piece of malware is capable of infecting a Linux server it's usually fairly certain that it has been designed to steal information. Of course, the aim of the malware may instead be to enlist your server into a botnet or as a means to installing a backdoor.

Using a websearch to find out as much information as possible about any relevant strains of malware is a must. Without this information it's impossible to accurately assess what the next course of action should be.

18.2.5 Action to take

We should, by now, have some idea of what the malware is and does. We now need to assess just how compromised the server is and whether it's possible to clean the infection without a re-image.

We generally use the following decision table

	Question	
1	*Was the infection a rootkit*	
2	*Is the malware known to install back-doors*	
3	*Was the malware installed system wide?*	
4	*Did the malware affect any files that should not be writable by the user*	
5	*Could the malware have affected any of the users files*	*Back*
6	*Can the malware be removed without removing the user account*	
7	*Was the infection as a result of a user doing something they shouldn't?*	*Consider possibility*
8	*Backup files, remove user account and re-add. Scan backed up files and restore where possible*	

There will always be occasions where the decision table above isn't suitable for the scenario. In this case, we need to follow (and document) a logical train of thought. Where possible we aim to minimise disruption by simply cleaning the infection, but due consideration must be given to the possibility that this may not be the most appropriate route.

Similarly, we don't generally want to ban users on a whim[17] but if IT policies are being deliberately ignored, sometimes disciplinary measures may need to be applied. A temporary ban with a notification to the users Line Manager as to exactly why the user was banned can often be an effective method where a user is seen to be deliberately flaunting good practice. Of course, such a decision should always be taken by the SysAdmin's line manager (or higher) with full documentation of the considerations taken and decisions made.

[17] However tempting that may sometimes be!

Chapter 19

Logs

Many of the applications and services running on a server will maintain logs intended to aid troubleshooting in case of malfunction or compromise. Whilst the location of these logs can usually be customised on a per-application basis, most will usually be within */var/log*.

Within this chapter, we'll be looking at common log locations and ways in which you can quickly parse certain log files to help identify issues. We'll also be looking at logs which can prove very useful in detecting issues, regardless of the applications in use.

It's also worth noting that whilst logs are often very useful in investigating a failure of any kind, this is not the only time that they should be checked. A good SysAdmin will know the importance of regularly checking logs in order to address issues before they arise.

Finally, a competent attacker may also have tampered with logs, so a lack of relevant entries does not necessarily mean that something hasn't happened!

19.1 Log file Contents

The actual content of log files varies based upon the application responsible. There's no real set format, largely because each system needs to log different information. To further fragment the issue, some software will allow you to customise the format of the logging output (Apache is one such application).

Assuming you have Apache running on your system, there should be some log files in one of the following locations[1]

- /var/log/apache2

[1] The location may, of course, have been customised

- /var/log/apache
- /var/log/httpd
- /etc/httpd/logs
- /etc/apache2/logs

You can use the *ls* command to view the contents of each directory (it's unlikely that all of the above will actually exist!) in order to ascertain which directory you need. For the purposes of examples, we'll be assuming */var/log/httpd*

Running the command

```
ls /var/log/httpd
```

Will usually return at least two files - *access_log* and *error_log*. If log rotation is set up then there may also be a number of historic logs in the directory.

The two were are interested in contain logging information for two types of events - Server requests and Server errors[2].

As the log entries are in the order in which they were written, it's often useful to have a reasonably accurate date/time for the event we are trying to find. This is part of why a SysAdmin will ask for so much detail when an error is reported to them.

To view the entire log, we'd use the following command

```
cat access_log | less
# View the last 500 entries
tail -n 500 access_log | less
```

You can also search the log using *grep*, so to only view entries from 28 Mar 2012 within the access log, we'd use

```
cat access_log | grep "28/Mar/2012"
```

19.1.1 Apache Access Log

We'll begin by looking at the access log; we are only interested in the last few requests, so we'll utilise the command *tail* to limit the results

```
tail -n 3 access_log # Show the last 3 requests
```

The results will normally be in the following format

[2] It should be evident from the names which is which!

Requesting IP	*Date*	*Request*	*Status*	*Size*	*User-Agent*

Where information wasn't available, a dash is normally used. The contents of this log can be customised within the Apache configuration file to contain additional information. Where errors are encountered, the Status field will generally contain an code of greater than 400.

19.1.1.1 HTTP Status Codes

For reference, below is a list of the current HTTP status error codes you may come across, and a brief description of what they mean.

Note: as a rule of thumb everything below 400 means the request was successful. 300 codes usually redirect the browser to another location (perhaps as the result of a page being moved)

Code	*Explanation*
400	***Bad Request*** *The server didn't understand the request*
401	***Unauthorised*** *The user isn't authorised to view the requested resource - Need to log in first*
402	***Payment Required*** *You will only rarely come across this!*
403	***Forbidden*** *Regardless of authentication, the user is not permitted to view the resource*
404	***Not Found*** *The requested resource doesn't appear to exist*
500	***Internal Error*** *The server encountered an error which prevented it fulfilling the request*
501	***Not Implemented*** *The server doesn't support the requested functionality*
502	***Server Busy*** *The server may be processing more requests than it can handle*
503	***Gateway Timeout*** *Where the server was waiting for another to respond, the response took too long to return or wasn't received.*

19.1.2 Apache Error Log

The contents of the error log are entirely dependant on the configuration used for Apache. The log can be made to contain basic information or very detailed information. By default, the log will normally be quite basic, but this doesn't reduce it's utility when investigating issues.

Frustratingly, the error log uses a slightly different format for dates, so you need to keep this in mind if you are intending to search the file using *grep*.

```
tail -n 3 error_log # Show the last 3 entries
```

The default format is similar to that of the access log.

Date	***Error type***	***Client***	***Error Details***	***Referer***

Whereas the date format in the access log is *dd/mm/yyyy:H:m:s timezone(28/Mar/2012:11:43:32 +0100)*, the format in the error log includes the day of the week, and uses spaces to delimit (*Wed Mar 28 11:43:32 2012*).

The log can be incredibly useful for tracking the source of issues. The way to effectively back-trace is to start at the last error and work up (making sure each error is for the same client) until you find the original cause of an error. Although errors may appear, it's often the case that they are as the result of an earlier error and you'll otherwise waste time fixing something that is working as it should.

19.2 Authentication Log

The authentication log contains authentication data from a range of applications. For example, if you have the KDE window manager installed you may see entries from *kcheckpass*. The log file will normally be */var/log/auth.log* although it's also possible that the system has been configured to log elsewhere. Some systems, in fact, will simply log authentication data to the system log instead of maintaining a separate log file.

To get an idea of the contents of the authentication log, we simply need to run

```
tail -n 5 /var/log/auth.log # Return the last 5 entries
```

Generally the log will be laid out in the following manner

Date/Time	***System***	***Application***	***Activity***

So an example entry would be

Mar 28 12:30:58	*mysystem*	*CRON[17897]*	*pam_unix(cron:session): session for user root by (uid=0)*

This example entry simply tells us that at 12:30 on Mar 28 a cron job was run by *root* (which in practical terms means that the system automatically logged *root* in and ran a command).

The log will almost never contain details of failed passwords (it'll simply log a failed attempt rather than including the submitted password)[3]. However, it will allow you to see where multiple attempts have been to log into specific accounts, which is particularly useful when checking whether or not attempts have been made to brute force the system.

Similarly, this log will also allow you to identify when users have logged in, which can also be of use when trying to identify the potential cause of a security breach (such as malware based compromise).

It's also worth noting that *sudo* will generally log to this file, whether because a user without *sudo* privileges has attempted to utilise *sudo*, or because a user has successfully used it. Example *sudo* entries are below

19.2.1 Failed Sudo entries

Mar 28 12:30:58	*mysystem*	*sudo*	*bob : 2 incorrect password attempts ; TTY=pts/1 ; PWD=/home/ben/tmpdir ; USER=root ; COMMAND=/bin/bash*

Explanation: Bob attempted to use *sudo -s* (as indicated by *COMMAND=/bin/bash*) but entered his password incorrectly (twice).

Mar 28 12:30:58	*mysystem*	*sudo*	*bob : user NOT in sudoers ; TTY=pts/1 ; PWD=/home/ben/tmpdir ; USER=root ; COMMAND=/bin/ls /var/etc*

Explanation: Bob attempted to run *ls /var/etc* but is not a sudoer.

19.2.2 Successful Sudo Entries

Mar 28 12:30:58	*mysystem*	*sudo*	*ben : TTY=pts/1 ; PWD=/home/ben/tmpdir ; USER=root ; COMMAND=/bin/ls /var/log*

Explanation: Ben successfully ran *ls /var/log* using *sudo*.

19.2.2.1 Limitations

Although *sudo*'s habit of logging all commands is very helpful, there are however some limitations. We saw in an earlier example that if a user uses *sudo -s* that

[3] Any application that does log this information should be considered a security risk

the log will state that the command was */bin/bash*. Whilst this is helpful in telling us that the user had used the *-s* argument, it unfortunately does not tell us which commands a user subsequently ran.

However, the subsequent commands will have been logged to the user's bash history file. Be aware that the user can manually edit this file, so any incriminating evidence may have been removed or altered, though this should not prevent us from at least looking.

We know the user ran *sudo -s* so we simply need to check for any commands issued between that and the *exit* command.

To check the user's bash_history we run the following command

```
#We need to be root to do this
sudo -s

nano ~{username}/.bash_history
# I.E. nano ~ben/.bash_history

# Press Ctrl+W to open search
# Type sudo -s and press Enter
# We should see something like
sudo -s
nano /etc/passwd
exit
```

We can see from the above example that the user ben has been tampering with /etc/passwd. Keep in mind that there may be more than one occurrence of *sudo -s* in the users *.bash_history* file, so simply press Ctrl+w followed by Enter to move to the next.

19.3 System log

The main system log is an incredibly useful tool when investigating issues. The location is usually */var/messages* or */var/syslog*, though this may have been customised. The contents of the messages log may sometimes duplicate that available in other log files, but often contains additional information.

Most importantly when troubleshooting technical issues, this log file contains messages sent by the kernel. This can include hardware being detected, details on kernel panics as well as any kernel level changes that have been made (insertion of modules etc.)

The format of this log is much the same as the authentication log

Date/Time	***System***	***Application***	***Activity***

Depending on the issue being investigated, you may need to seek outside advice in order to understand some of the entries.

19.4 DMesg

Although DMesg shows logging output, it's not actually a log file per se. Instead, it is a command you run to view output from the kernel. It usually reads directly from the System log, although there are other methods it can use to extract this information.

Unless the system has been adjusted to prevent this, every user will generally be able to run dmesg.

Generally we use dmesg to view anything the kernel may have output when the system booted.

To utilise dmesg we simply run

```
dmesg
```

It's also possible to use command line arguments with dmesg, however their use falls outside the scope of this book as they are not normally required for basic troubleshooting.

Chapter 20

Disk Space Usage

Although not usually something that requires an experienced SysAdmin to think too hard, the issue of excessive disk space usage is something that often trips up less experienced administrators.

In this chapter we'll be examining some of the tools that can help you locate files to move/remove. We'll then touch briefly upon the issue of 'sparse' files, something which has tripped up many an inexperienced sysadmin.

It's worth noting that the issues described here are, for the large part, exactly the same as you may find on a Microsoft^{TM} Windows® based server.

20.1 Partition Usage

Generally, when Linux is installed multiple disk partitions are created with each being mounted in a different directory. Although there is some variance, common practice is to create a separate partition for each of the following directories

- /
- /var
- /home

We will generally know which partition is full if issues have been encountered, but none-the-less it's important to know how to check usage.

```
df -h
```

The output from *df* should be fairly self explanatory, but be aware that the results may also include sparse files[1].

[1] More on these later

The command we ran above used the *-h* flag, which instructs *df* to output results using human readable sizes. However, keep in mind that some rounding will occur as a result, so for accurate results you need to omit the *-h* argument.

20.1.1 Flags to use with df

df supports a number of command line arguments. These are listed below

Flag	***Description***
-a	*Include Dummy File Systems*
-h	*Use human readable sizes*
-H	*The same as -h but uses blocks of 1000 instead of* 1024^2
-l	*Only show sizes for local filesystems (i.e. not NFS mounts)*
-T	*Print filesystem type*

20.2 Folder/File Size

The easiest way to find out the size of a given folder is to use the command *du*. So if we wish to see the total size of all folders within */var* we would run the following command

```
du -sh /var/*
```

This will output a list of all folders within /var alongside the size of their contents. If we omit the *-s* argument then all files within those folders will be listed. As with *df*, the *-h* flag instructs du to use human readable sizes.

20.2.1 Flags to use with du

du supports many more command line arguments than *df*, but then it is designed to achieve a different task. The most commonly used are listed here.

Flag	*Description*
-a	*Writes size for all files (can't be used with -s)*
-b	*Print size in bytes*
-c	*Produce a grand total*
-h	*Print sizes in Human Readable Format*
-H	*Print sizes in Human Readable Format but use incorrect power of 1000 instead of 1024*
-l	*Count hard links as many times as they are used*
-S	*Do not include size of subdirectories*
-s	*Output a summary for each argument (files/directories)*

20.3 Sparse Files

You may occasionally find that the output of *df* and *du* disagree with each other, there can be many reasons for this but often it's due to the existence of 'sparse' files. The definition of a sparse file is actually very complex[3], so for simplicities sake we'll simply say that a sparse file is one that appears to absorb far more space on disk than it otherwise should. Generally a sparse file will appear empty, although it is absorbing space.

Sparse files are often used for logging, and as a result you may find that although you've moved or removed some log files, the space is still being used. The simplest way to remedy this is to restart any services responsible for those log files (or indeed the server itself).

So if you do find that */var* has become full and have reacted by removing old log files, to fully reclaim the space you may need to restart services such as Apache.

20.4 Investigation Techniques

There are a range of methods you can use to investigate the reasons behind low disk space. Which you use depends largely on what you are actually trying to achieve; you may simply want to find large files so that you can quickly clear some space or you may wish to find the areas responsible so that you can investigate quite why free space is so limited.

The former of these is useful when you need to quickly free some space (a full disk may prevent some services from running), whilst the latter is the preferred method where possible (as the aim is to prevent recurrences). Performing the

[3] For end-users at least

latter task will, however, be made a little more difficult if you remove large files first!

Hard drive space is, however, a finite resource, so it may simply be that you either need to allocate more space, or reduce the size of files stored to disk.

20.4.1 Finding Large Files

The easiest way to find large files is to run a system wide search (or from the root of a given partition if it's a single partition at fault). To do so, we use the *find* command including the size flag.

So to search for any files large that 2 Gigabytes on /var we would run

```
find /var/* -type f -size +2097152k -exec ls -lh {} \;
```

We've specified the minimum filesize in kilobytes as it's easier to get in the habit of always specifying in this manner with find!

The command will then search for any files with a size greater than 2079152 kilobytes in any directory under /var.

Note: You may also receive a number of 'permission denied' errors if you run this as a non-root user.

A list of the files found will be output, you can then proceed to do what you consider necessary with them. You should, however, keep in mind that some files may be system critical so don't simply remove them on the basis that they are large!

20.4.2 Finding the root cause

As with anything else, it's always better to try and identify the root cause of disk space issues. It could be that a user is simply absorbing too much space, or that verbose logging has been accidentally left on for a system service. Until we locate the files in question, it's impossible to know for sure.

How you perform this investigation depends partially on your style of working, but also on how much you already know about the issue. For example, if */home* is on a separate partition and is becoming full, there's very little point in running a system wide check when we can start with */home*!

The basic procedure involves working slowly through the directory hierarchy looking for directories with large sizes. It may be that there are in fact multiple directories contributing to the issues, in these cases it's best to make a note of the directory and then look into when the structure has been fully investigated.

For our examples, we'll assume that the entire system is on a single partition. In which case we want to begin at the root of the directory tree - /

```
# We need root privileges to perform a full investigation
sudo -s
cd /
# Now that we are in the root directory, let's try and
# identify the folders we need to look into
du -sh ./*
```

Running the above commands will return output similar to the following[4]

```
6.5M ./bin
80M ./boot
4.0K ./cdrom
332K ./dev
17M ./etc
40G ./home
50M ./lib
0 ./media
0 ./mnt
65M ./opt
512K ./proc
3M ./root
6M ./sbin
800K ./sys
400M ./usr
50G ./var
```

In the above example we can see that /var is using 50 Gigabytes of space on disk. Whilst /home is using 40G this is to be expected as all user files are stored here. Generally speaking, /var stores log files and spool files[5] so we wouldn't expect a size any where near 50G under ordinary conditions.

The next step is to recurse through each level following the file sizes until we find the cause.

```
cd /var
du -sh ./*
```

We will expect similar output to that above, although it will obviously only contain directories within /var. Assuming the following output

[4] The command will take some time to complete!
[5] There are, of course, exceptions

```
4.6M ./backups
475M ./cache
4.0K ./crash
292M ./lib
4.0K ./local
0 ./lock
49G ./log
4.0K ./mail
4.0K ./opt
216K ./run
260K ./spool
106M ./tmp
```

We can see that the directory /var/log is responsible for 49 gigabytes of the space absorbed by /var. So we'll *cd* into log to find where the space is being used.

```
cd /var/log/
du -sh ./*
```

This time our output likely included a number of files, as the command used will include these[6]. We may find that a number of files are together responsible for absorbing the space, or that a single file is.

Having located the files responsible, we may need to reconfigure services to reduce the verbosity of logging, or perhaps investigate the cause of errors contributing heavily to the size of the log. The only way to know is to view the contents of the log file.

[6] Where best practices are followed, neither / or /var will directly contain files.

Part VI

The Business Case for Linux

Chapter 21

Introduction

In this Part we'll be looking at the business case for Linux. The potential cost and benefits of using Linux will obviously vary on a per-business basis. However, we'll be examining several scenarios in which you may be considering using Linux and exploring the factors that need to be considered.

Many of the options we'll explore here remain the same whichever type of system you are planning to implement. It is, however, strongly advised that you consult a professional SysAdmin in order to evaluate the costs based on your business needs, as this simply isn't something a book can accurately achieve.

Within this part, we'll be examining

- ▷ Global Factors to Consider
- ▷ Scenario Dependant Factors
- ▷ Licensing Costs
- ▷ Retraining Staff (Where necessary)
- ▷ Total Cost of Ownership

To begin, we'll first dispel a common myth: the idea that if something is free it can't be as good as the 'paid for' versions. Whilst there are many situations where '*you get what you pay for*', Open Source Software isn't one of those. The quality of each project varies, just as in the proprietary world, but many projects (especially the popular ones) offer solutions that are at least as reliable as their proprietary counterparts.

As an example, Linux is used to run systems ranging from tiny embedded systems to supercomputers. There's usually no cost for the operating system, and yet Linux is used by companies and organisations across the globe for a wide range of tasks.

For many experienced users, the proprietary alternative - Microsoft^{TM} Windows® - is considered to be far inferior for many uses. The truth in this depends entirely on the user and what they are trying to achieve, but the point is that the quality of a software solution should never be judged solely by price.

Throughout this section, the cost of the physical hardware has not been included. Whilst Linux requires less resources and so will run on a less powerful server, deliberately choosing lower-end hardware is not in any way recommended as it limits the potential for future expansion.

Chapter 22

Global Factors

There are a number of factors that need to be considered regardless of whether you are migrating data/functionality from an old server, or simply commissioning a brand new server.

Within this chapter, we will look at some of these, including

- ▷ User Training
- ▷ Product Support Costs
- ▷ Maintenance Costs
- ▷ Vendor Lock-In
- ▷ Requirement for updates

Each business may also have it's own considerations to take into account. For example, a financial institution may need to take into account additional strictures relating to their profession, whilst those processing card payments will need to consider controls such as PCI DSS. Linux systems will generally conform to both of these, but the potential costs of maintaining compliance should also be considered where applicable.

22.1 Server Usage Requirements - User Training

What exactly will the server be used for? A large proportion of your potential capital expenditure is likely to be reliant on this question. If you are planning to use the server as a Webserver then it's likely that only your IT department are likely to need training. If your IT Department are familiar with Linux this may also be avoidable.

If your business utilises a server/thin client architecture then it's possible that you may need to retrain some/all of your users. Ordinary users, however, won't need to learn the techniques we've explored in this book as they will most likely be assigned a GUI. The experience of companies around the globe also suggests that most users will not struggle with a new interface - those that do are usually the 'Windows® Power Users' who are used to being able to tweak and customise their Windows® desktops as they see fit. These users are likely to be most vocal about the change, but with a little training and support can be made just as comfortable on the new system.

It's also worth considering the issue of continuity. The cost of re-training has, historically, been used as a reason not to convert to Linux. However, at time of writing, the interface changes made in the latest versions of Microsoft™ Windows® are also necessitating the same. Perhaps there has never been a better time to convert to a new system, especially given that the interface is likely to continue to evolve in each release.

22.2 Product Support Costs

Are you going to require an ongoing support contract? It may be that your IT Department feel they can resolve most issues themselves, allowing you to instead use a 'pay as you go' support model - Paying for support on a per case basis on the occasions that you do need to seek outside help.

It may also be financially beneficial to look at outsourcing to a company other than the vendor. Whether on a pay per-case basis or a cheaper monthly contract, this will provide the re-assurance that your IT Department have experts that can be contacted when necessary[1]. The experience of many companies suggests that the original vendor is not always best placed to provide the support that businesses need, instead companies with real-life experience in running and maintaining systems should generally be your first point of call.

22.3 Maintenance Costs

Whilst a Linux server will, as a rule, require less maintenance than it's Windows® counterparts, you should consider the familiarity of your IT Department with the two systems. If your SysAdmin is likely to encounter difficulties and as a result take longer to complete a routine task, this is a cost that should be considered. This loss of productivity can, however, always be mitigated with effective training.

Microsoft™ commissioned Ideas International (IDEAS) to produce a white paper intended to compare the costs of acquiring and supporting Windows® versus

[1] Shameless Plug: Virya Technologies can provide this support on either basis

those for Red Hat® Enterprise® Linux. The white paper concluded that for the first four years, Red Hat® was less expensive than Microsoft™ Windows®. At year 4, the cost broke even and Red Hat® became more expensive to support.

However, that report also failed to include other important factors. By year 4, most large modern businesses will be looking to replace the server and will almost certainly have needed to update licenses on the Windows®servers. The report also looked solely at large Enterprises who would benefit from Microsoft™'s volume pricing discount, Software Assurance subscription and distributed payments option. For smaller businesses, this simply isn't available so the Total Cost of Ownership (TCO) of a Microsoft™ Windows® server is liable to be much, much higher!

22.4 Vendor Lock-In

Whilst the issue of Vendor lock-in is, in many ways, far less severe than it was in the 1980's[2] it's still an area that warrants serious consideration.

It's rare that it's actually impossible to migrate data to a new system, but it can be extremely cost-prohibitive. Businesses can easily become tied to solutions provided by a single vendor as evidenced by the fact that some businesses feel that they cannot use anything but Microsoft™ solutions.

Data stored in non-open standards is particularly vulnerable, as the vendor may hold patents preventing other businesses from creating solutions to extract that data.

Any aspect of Vendor lock-in puts businesses in a potentially dangerous situation. Whilst no businesses will initially plan to stop using a vendor at time of implementation, circumstances may change leading to an urgent need to migrate data.

For example, what happens if the chosen vendor drastically increases their license fees? If the increase equates to $1000 a year, but the cost of migrating the data to another solution is likely to be $10,000 the business may have no option but to swallow the increased licensing fees. Of course, if the business can't actually afford the increase this could pose serious issues, especially where the data in question is business critical. IT is an industry that has historically aimed to lock users into a specific vendor, so never underestimate the risks posed.

It's not only licensing fees that need to be considered. What happens if the vendor goes bankrupt, or is purchased by another company which decides to end support for the software[3]? Companies reliant on those solutions will receive no further updates, regardless of any security issues discovered, and will eventually

[2] When lock-in was part of the business model of the industry's incumbents
[3] As happened recently on a number of products when Oracle purchased Sun

either need to foot the cost of migrating data or continue running outdated, unsupported software.

Any business should take the sensible precaution of insisting that all data is stored using open standards, or at the very least begin implementing a cost-effective data migration plan from day one of the roll-out.

Using Open Source Software such as Linux (for the Operating System), MySQL (for the database) and Apache (for the web server) helps to mitigate the risks of Vendor lock-in. Because the source code is available, further development is always possible even if the original company ceases support. Where projects are popular (as in the case of our examples), it's highly likely that the project will be forked, and a new team will continue development.

In the case of more niche projects (such as some financial management software) the project may not be forked. However, because the source code is available, businesses always have an additional option: they can fork the project themselves by paying a developer to continue support (whether this is a full-blown continuation of the project, or simple security and bug fixes).

Additionally, where the source is available, it becomes far easier to see exactly how data is handled, easing the transition should data need to be migrated to another system.

This, in truth, is one of Linux's main strength's in the Enterprise arena. Most proprietary alternatives present a risk (no matter how small) of potential vendor lock-in, whereas the Open Source software available for Linux presents a risk so small as to be considered trivial.

22.4.1 Potential Causes of Lock-In

To summarise, some of the potential risks posed by vendor lock-in are

- ▷ Original Vendor raising prices arbitrarily
- ▷ Original Vendor ceasing business
- ▷ Support for the solution being dropped
- ▷ Inability to migrate data cost-effectively
- ▷ Potential inability to access data at all

Most of these can be addressed simply by insisting on using Open Standards (such as Open Document Format) and utilising Open Source Software where possible.

22.5 Requirement for Updates

No software is perfect when released, whether OSS or otherwise. It's also fair to say that no software can fully address every need of every business. A solution containing 99% of what your business needs may still omit a crucial 1%.

When using proprietary off-the-shelf solutions, this is often something that businesses have to tolerate. There is, of course, always the option of approaching the original vendor for a bespoke solution, but these can be very expensive.

It may also be that your business finds a bug within the software; unfortunately in many cases you will need to wait for the original vendor to create a fix and push out an update. Whilst this bug may stop your business from operating effectively, the vendor may decide that it's a low priority and so a fix may not be forthcoming for quite some time.

Some vendors even charge for non-security updates, meaning that your business will need to pay an additional amount when the bug is eventually fixed.

However, Open Source Software addresses many of these issues. Whilst approaching the original vendors may still present similar issues, the source code is publicly available so fixes can be implemented through other means. The software in question may already have a team of volunteers who create fixes and submit them to the vendor, so it may be possible to report the issue to them and receive a patch long before the vendor pushes a formal update. Even if this is not the case, businesses retain the ability to employ a developer to correct the issue. Whilst this may not seem a suitable solution for minor bugs, it's usually well worth considering in instances where the bug is preventing your business from operating as it should.

The reality is that Open Source Software is the only way in which businesses are offered this option. Closed-source software by it's very nature prohibits users from pursuing this route without reverse engineering[4] the software.

[4] Which most proprietary licenses forbid.

Chapter 23

Scenario Dependant Factors

23.1 Commissioning a brand new server

If you are commissioning a server that won't need anything migrated from an old Windows® based server, many of the potential difficulties are already overcome. Rather than needing to calculate the costs of migration, we simply need to establish potential retraining costs and potential licensing savings.

23.2 Migrating From an Old Server

If you are investigating the possibility of commissioning a new server so that you may retire an older system, there may be additional costs involved (regardless of which system you eventually use). The intended use of the system is obviously a crucial factor as it dictates what will need to be migrated.

In this subsection we'll briefly examine some of the common considerations that need to be examined. We'll also look at some of the solutions that businesses have used to help mitigate costs.

23.2.1 Databases

If your old system acted as a database server, which database management system (DBMS) was it using? There are a wide range of DBMS' available including

- MySQL
- MSSQL

- ▷ MS JET
- ▷ Oracle®
- ▷ PostgreSQL

Of these, all except MSSQL and MS JET are platform agnostic[1]. So although there will be an administrative cost in migrating the databases, there should be very little additional cost in a migration. In the case of the Oracle® DBMS, you will likely need to purchase a new license to utilise the newest version, but this is generally considered a wise move when moving to a newer system as bug fixes will likely be present in later versions.

Both MySQL and PostgreSQL are freely available, so no licensing cost should be incurred here.

Which DBMS you will want to use will depend entirely on the need of your business. Oracle^{TM} have a fearsome reputation within the large to very large enterprise sector, but many smaller businesses find that they do not require the full set of features provided and so the solutions fail on cost/benefit calculations.

Migration from one DBMS usually involves creating scripts[2] designed to move the data between, and where necessary altering formatting so that the new DBMS will accept it. Depending on the systems used, some changes may also need to be made to applications which rely on the DBMS.

23.2.2 Web Servers

Although the term 'web servers' covers a wide range of implementations, what is relevant here is which HTTP server has been used and whether the sites being served are dependant on that particular software.

Although others are available, generally speaking a web server will either be running Apache or Microsoft^{TM} Internet Information Services (IIS). The former is platform agnostic and so can be used on a wide range of Operating Systems. The latter is, as one might expect, dependant on the Microsoft^{TM} Windows® platform.

Sites may be dependant if they utilise Server Side Scripting such as PHP or ASP. In reality, it's actually the latter of these that poses the biggest problem when migrating. PHP can be installed on various operating systems, so a competent SysAdmin will usually be able to run a PHP based site on either Apache or IIS.

ASP can be utilised on platforms other than Windows® by using the ASP compatibility library for PHP, however the end result cannot always be guaranteed and there may also be a performance overhead.

[1] In other words, they will run on a variety of Operating Systems including Linux.
[2] Some have been posted online to aid SysAdmins

The costs of rewriting an ASP site in PHP could be very substantial, so where possible it will often be better to consider utilising the compatibility library and slowly migrate to PHP by means of natural wastage. So when a site needs to be replaced, the replacement would ideally be written in PHP instead.

23.2.3 Key Business Applications

There may be a need to run software that is deemed 'business critical'. You need to evaluate which platforms this software will run on, and whether it will continue to fulfill your needs. If you are unable to run that key piece of software, are there alternatives that will fulfill the new task on the new platform?

This is perhaps the only justifiable reason for running Microsoft^{TM} Windows®on a production server, especially given the wide range of Windows® only business applications currently in the marketplace. Many providers, however, are beginning to support other platforms so it's also worth asking your vendor whether they are intending to expand support in the future.

It may also be possible to use an Emulation layer such as WINE to run the software. This can only be evaluated on a per-application basis, and will not always yield satisfactory results.

This area is, perhaps, one of the key causes of vendor lock-in.

23.2.4 Mitigation: Virtualisation

Some businesses have opted to migrate to Linux based platforms whilst utilising virtualisation to run various Microsoft^{TM} Windows® dependant applications. Virtualisation consists of running one or more virtual servers on a single physical server, as the virtual machine generally believes it is running on a physical server there are very few limitations as to which systems you can run within the virtual environment.

Virtualisation also has the benefit of allowing you to very quickly restore a system in the case of a complete breakdown within the virtual machine, but does carry the risk that you will lose several services at once if your physical server fails[3].

When considering the possibility of virtualisation, it's important to note that the physical server will need far more resources available to it than if it wasn't acting as a host system. If you have three virtual machines running on the system, there needs to be sufficient RAM and Processor capacity to handle this.

Software such as VirtualBox and VMWare are considered to be excellent candidates when planning to implement virtualisation.

[3] It is possible to run Virtual machines across numerous physical servers, which will allow for some redundancy.

Chapter 24

Licensing Costs

Potential licensing costs are inherently dependant on which software a business needs in order to function. Sometimes, however, businesses do find that they are paying higher licensing costs as a result of purchasing software that achieves far more than they actually need!

A full cost/benefit evaluation should always be undertaken before making any change to IT infrastructure, but this becomes particularly important where software is deemed to be business critical.

In this chapter we'll be looking at some of the costs that can be avoided or reduced by using a Linux based server rather than one using Microsoft^{TM} Windows®.

24.1 Operating System Costs

As we discovered earlier in the book, a Microsoft^{TM} commissioned report comparing the costs of Microsoft^{TM} Windows®Server 2003 to the equivalent Red Hat® Enterprise® Linux release discovered that for large enterprises, a cost saving could be achieved for six years by opting to utilise the Linux based system. This, however, took into account support contracts as well as applying discounts that Microsoft^{TM} purposely only make available to large enterprises.

In this section we'll be comparing the capital cost of purchasing licenses to provision a single server with an Operating system. All applications, such as e-mail will be examined later in this chapter.

In order to assess base price, rather than that imposed by re-sellers the vendors currency has been used. In most cases this will be US Dollars. Volume discounts have not been applied on the basis that most Small to Medium Enterprises will not be eligible to receive these.

24.1.1 Shared Costs

There is, of course, more to the cost of the Operating System than licensing prices. You'll also need to employ someone to install the OS for you, which may require a number of hours where there's a requirement to configure services as well.

Depending on the arrangements for the server, it may be that the OS will be pre-installed for you. This is especially true where the server is being provided by a hosting provider (as is usually the case for webservers).

24.1.2 Microsoft Windows Server 2008 R2

Although, at time of writing, Microsoft^{TM} Windows® Server 8 is due to be released, licensing costs have not yet been announced. For the purposes of this book, we will instead examine the version currently available on the open market. To commission a single server, most business would use the "*Windows® Server 2008 R2 Standard*" license, due to the fact that it is available through Retail and OEM channels.

MS advertise the cost of a single Windows 2008 R2 Standard as $1,029 including 5 Client Access Licenses (CALs). Microsoft's policies dictate that every user or device which accesses or uses Windows® Server 2008 must have a valid CAL purchased for it. There are, however, exceptions based on the use of the server

- ▷ If the server is only accessed through the internet without access being authenticated or otherwise individually identified (i.e. if the system is running as a web server)
- ▷ If a Windows 2008 External Connector license has been purchased for that server (users must be external)
- ▷ Up to two devices or users may access the server for the purposes of administering it.
- ▷ If the server is acting solely as a virtualisation host (CAL's will still be required for any Windows® based virtual machines)

This license will also allow us to run a single virtual machine if we wish.

So, depending on the needs of your business and the demands placed upon the server, the cost of the capital cost of the operating system is likely to be equal to or greater than $1,029.

24.1.3 Red Hat Enterprise Linux Server

Red Hat® are currently the industry lead in Enterprise Linux systems. Many smaller businesses will opt to use another system and vendor, but these will be examined shortly.

Red Hat® base their licensing costs on the configuration of the server you intend to deploy to, for the examples used here we will opt to utilise the '*Self-Support, 2-Sockets with 1 virtual guest*" subscription. What this means is that we will need to provide our own IT support, or pay on a pay-as-you go basis, and may run the system on a server with up to 2 processors. Additionally, our license will allow us to run a single virtual machine on the system.

At time of writing the cost of this subscription is $349 per year. If we wish to add support, this rises to $799 per year.

24.1.4 CentOS

Most business which don't wish to pay for RedHat will usually opt for CentOS instead. Both systems have more or less the same code-base and so the technical differences between the two are minuscule. The main difference, of course, is that Red Hat® provides dedicated support.

CentOS can be downloaded and installed for free, so the only capital expenditure will be the provisioning of the server and the man-hours required to install the Operating System (usually very few).

24.2 Email Servers

There are a number of email servers available (known technically as MTA's[1]). The most commonly used on Microsoft™ Windows® servers is of course Microsoft™ Exchange®. In this section we'll be examining the licensing costs of various solutions in comparison with Exchange®. We won't, however, be comparing the usefulness of each as this is something that can only truly be assessed on a per-business basis. Most non-trivial email handlers, however, do not use Microsoft™ Exchange® due to better solutions being available.

It should also be noted that some Linux Server distributions will come with an email server pre-installed. There's nothing to stop you from opting to change the software in use if you desire, however.

[1] Mail Transport Agent

24.2.1 Microsoft Exchange

The licensing cost of Microsoft^{TM} Exchange® server 2010 has been used as a comparison. In order to utilise Exchange you will require

- ▷ A server License
- ▷ Client Access Licenses (CALs)

There are two types of CAL available for Exchange, however in order to reduce costs we'll assume that the Standard CAL is preferred over the Enterprise option. In order to use an Enterprise CAL, we'd first need to purchase a Standard CAL so this needs to be considered if you believe you would need an Enterprise CAL.

As with Microsoft^{TM} Windows® Server 2008, a CAL is required for each user or devices that will access the server software. So assuming your business has five users which will require access[2], you will need to purchase 5 standard CALs.

At time of writing, the costs are as follows

License	Qty	Price	Total
Exchange Server	*1*	*$699*	*$699*
Standard CAL	*5*	*$67*	*$335*
		Total	*$1,034*

Note: If you do require an Enterprise CAL each CAL costs an additional $35

So as we can see, the capital expenditure to license Microsoft^{TM} Exchange® 2010 and grant access to 5 users is $1,034. Added to our projected capital cost of purchasing a Microsoft^{TM} Windows® Server 2008 R2 License, this brings our CapEx to $2,063

24.2.2 Postfix

PostFix is a popular Open Source Email server originally created within the IBM Research labs. It's considered a popular alternative to Sendmail.

Postfix can be downloaded and installed for free, with no limit on the number of users. Most Linux distributions have Postfix in their repo's so it needn't be manually compiled from source.

[2] We've selected 5 based on the number of CALs included with Microsoft^{TM} Windows® Server 2008 R2

24.2.3 Sendmail

Sendmail is a widely used Open Source mail server, available in both free and proprietary flavours. Sendmail will also provide services and support for users of the system for a price.

Although very popular, Sendmail has been slowly losing ground to Postfix.

24.2.4 QMail

Qmail bills itself as the "second most popular MTA on the web", whether or not this is true it certainly is widely used. It's the default server used on new installs of the popular Plesk management interface released by Parallels.

Qmail is fully Open Source and can be downloaded and installed for free.

24.3 Web Servers

Both Linux and Microsoft^{TM} Windows® have a wide range of Web servers (HTTP Services) available to them. However, generally speaking, there's no licensing cost involved on either platform.

Microsoft^{TM} Windows® ships with Internet Information Systems (IIS) whilst Linux systems generally run Apache. You can, if you wish, opt to run Apache on Windows.

In technical terms, the Web services stack will usually be referred to in one of the following manners

Term	Explanation
LAMP	*Linux, Apache, MySQL, PHP/Perl*
WAMP	*Windows, Apache, MySQL, PHP/Perl*

Other proprietary solutions are available, but most (if not all) businesses will find that Apache serves their needs. It also has the advantage that if the Operating System is changed at a later date, very little reconfiguration will need to be undertaken.

24.4 Database Servers

Many applications now utilise a Database server in one form or another. In the past, websites were manually coded and the content was written into the HTML files themselves. Now, however, the vast majority of sites are dynamic and so rely on a database of some form.

Although (most) databases use SQL to interact, there are differences between the implementations in each. Whilst basic functions will work, advanced features in one Database system may not work in another.

24.4.1 MS SQL Server

Microsoft^{TM} SQL Server 2008 uses a simplified licensing scheme, unfortunately at time of writing prices have not yet been announced, so we will use the 2008 R2 edition as our comparator[3]. There are a wide number of editions[4] available. However, for the purposes of comparison we will utilise the Standard edition in this section as this is the option most small to medium businesses will select.

There are a number of Licensing options available, however in keeping with previous sections in this part we will opt for the Server/CAL licensing model.

The Server license costs $898 whilst each Client Access License (CAL) costs $164. If your server will be running as an application host (whether a webserver or otherwise) you will probably only need a single CAL. Some organisations may need more, but this is entirely dependant on the applications in use.

So the total cost of licensing MS SQL Server 2008 R2 would be $1062.

So assuming our server will be running Microsoft^{TM} Windows® Server 2008 R2, MS Exchange (with 5 CALs) and SQL Server 2008 our running CapEx total is $3125.

24.4.2 Oracle

Oracle^{TM} Databases are widely used by large enterprises and will run on a variety of platforms. Smaller businesses may be able to utilise the Express edition, but in the interests of comparing apples to apples, the Standard license has been used here.

Oracle's standard edition can be purchased for $372 under a "Named User Plus" license. The standard edition requires a minimum of 5 users, however, which brings the total CapEx to $1860.

We can also add support for the first year for an additional $80 (per user).

The standard edition is, however, upwards compatible so it can be upgraded to one of Oracle's higher products if business needs demand it at a later date.

Oracle also offer a yearly license option which is currently $74 per user, per year.

[3] Some partners have, however, stated that although the licensing structure has been simplified, costs have risen

[4] The 2012 version promises just three thankfully!

24.4.3 MySQL

MySQL is a free and open source Database system which runs on a variety of platforms. MySQL was originally developed by Sun, however with Sun's demise the system was purchased by Oracle. Although still a very capable system, many users have begun migrating to alternatives such as PostgreSQL as a result of uncertainty about the systems future.

Despite this, many, many businesses continue to successfully use MySQL.

24.5 Example Licensing Costs

We've now examined a number of options available, including several potential applications which may be required. However, for ease of reference a few complete examples are displayed below.

Operating System	*Mail Server*	*Database Server*	*Licensing Cost*
Microsoft™ Windows® Server 2008 R2	*Exchange*	*MS SQL 2008*	*$3125*
Microsoft™ Windows® Server 2008 R2	*Exchange*	*Oracle*	*$3923*
Microsoft™ Windows® Server 2008 R2	*Exchange*	*MySQL*	*$2063*
Microsoft™ Windows® Server 2008 R2	*None*	*Oracle*	*$2889*
Microsoft™ Windows® Server 2008 R2	*None*	*MySQL*	*$1029*
Microsoft™ Windows® Server 2008 R2	*None*	*None*	*$1029*
Red Hat® Enterprise® Linux	*Postfix*	*Oracle*	*$1860 + $349 per year*
Red Hat® Enterprise® Linux	*Postfix*	*MySQL*	*$349 per year*
CentOS	*Postfix*	*Oracle*	*$1860*
CentOS	*Postfix*	*MySQL*	*$0*

Note: Web server software has not been included as both Apache and IIS are free. Where numerous free alternatives are available, only one has been included for the sake of brevity.

All prices represent licensing for 5 users where relevant.

As we can clearly see in the table above, a Linux based Mail and Database server can be obtained without needing to spend a penny in licensing costs. In comparison, using a Microsoft™ Windows® Server 2008 R2 based server will cost a minimum of $1029 in licensing. It should also be noted that CALs do usually require renewing regularly, so an ongoing licensing cost is also present where CALs are required.

Chapter 25

Retraining Staff

We briefly touched upon the issue of user training earlier, however in this chapter we will take a more thorough look at this issue.

One of the arguments previously used to favour Microsoft^{TM} solutions over Linux based systems was the shear cost of retraining staff to utilise newer systems. However, many are finding that this argument no longer holds weight due to the myriad of interface changes being made in Microsoft^{TM} Windows® at each release (which are in themselves necessitating retraining).

What kind and level of retraining would be necessary is entirely dependant on who will be using the system, and how. If you are planning on deploying Linux to the desktop, then you may need to give users basic retraining so that they can familiarise themselves with the newer interface. If you are simply planning on using Linux on servers then it may only be your IT Department which requires retraining.

Those used to the Windows® world may have heard of the Microsoft Certified Systems Engineer (MCSE) qualification. The 2003 MCSE consisted of 7 exams at a cost of $190 each, and candidates learn how to use and configure Microsoft^{TM}Windows®.

Within the Linux world, there are equivalents. However, due to the wider number of vendors you should give some consideration to which you select. For most businesses, the wisest course would be the Linux Professional Institute Certification (LPIC) as this is not distribution specific.

Assuming you wish to enrol staff on the LPIC-1, there are two exams to complete at a cost of $175 each.

If, however, you know that you will be primarily using Red Hat® based systems, you could enrol staff on the Red Hat Certified System Administrator (RHSCA) exam.

The cost for each RHCSA candidate is $400.

There are training providers available who will prepare your staff for the courses. However, it may not be strictly necessary to train staff to this extent, depending on the needs of the business and the demands upon the server. Where possible, however, training your staff is generally considered a wise move: you'll be sure that they are competent to a reasonable baseline, and they'll obtain a professional qualification.

Chapter 26

Total Cost of Ownership

The initial uncertainty around Total Cost of Ownership (TCO) is something that Microsoft^{TM} has historically played upon in their fight against Linux. However, more and more businesses are beginning to recognise that the numbers suggested by Microsoft^{TM} simply do not add up[1].

So far in this part, we have already considered a number of the considerations that contribute to the TCO of a Linux system. In this chapter we'll be looking further into this and examining a number of case studies involving real-life organisations who have made the switch to Linux.

In the final section of this chapter, we'll also view a list of well-known companies who use Linux systems.

26.1 Case Studies

26.1.1 Munich

The local government of Munich, Germany switched from Microsoft^{TM} based solutions in 2011. In March 2012, the mayor of Munich announced that as a result, the city had already saved €4m!

Munich's IT department avoided upgrading Microsoft^{TM} Windows® and Microsoft^{TM} Office® at a cost of €15m (with an additional €2.8m due within 5 years as CALs expire). The city states that 15,000 Microsoft^{TM} Office® licenses and 7,500 Microsoft^{TM} Windows® licenses would have been required as well as 7,500 new machines as their older systems would not have been able to run the latest versions of the software.

[1] In technical circles, it's recognised that they are using a tactic called Fear Uncertainty and Doubt (FUD)

Instead the city opted to continue using the older hardware by instead installing the less resource-hungry Linux Operating System.

The mayor also announced that following the change, support calls to their internal IT help-desks dropped from 70 per month to 46[2]. This becomes all-the-more deserving of recognition when considering that in the same period, the number of employees outside the IT departments increased by over 1,500.

The figures quoted as savings include the cost of support, retraining and porting costs.

Although not implemented yet, the French Government also have plans to move to non Microsoft^{TM} solutions.

26.1.2 Ernie Ball Guitar Strings

The CEO of Ernie Ball[3] has been very vocal about his decision to switch the business to Linux based systems. His initial reasons for switching were as a result of the company being scapegoated by the Business Software Alliance (BSA) an industry funded copyright enforcement body.

26.1.2.1 History

The business was raided by the BSA in 2000 (as the result of a report from a disgruntled employee), who found a number of unlicensed copies of programs. Ball immediately settled for $65,000 plus $35,000 in legal fees, but the BSA then subsequently used Ernie Ball to make an example. The companies name and reputation were tarnished on the evening news, and was even being used in the BSA's regional advertisements.

The 'unlicensed' software had originally been properly licensed. However, the company had a habit of passing machines onto other departments when necessary (so when engineering need a new PC, a clerical department may get their old system). However, the failure to uninstall old applications means that two copies of that application were installed, which is a license violation. Although the BSA could tell the second copy was not being used, it was still a license violation and no leeway was given.

The CEO of the business, understandably, felt that they had been treated very poorly ("*I couldn't treat a customer the way Microsoft dealt with me...I went from being a pro-Microsoft guy to instantly being an anti-Microsoft guy.*") and began a meeting with his IT department.

[2] An alleged 'disadvantage' of Linux is that support costs rise
[3] The worlds leading maker of premium guitar strings

26.1.2.2 Making the change

Using the words *"I don't care if we have to buy 10,000 abacuses, We won't do business with someone who treats us poorly."* Ball instructed his IT department to look into alternative solutions. Apple solutions were specifically forbidden due to Microsoft's 1997 investment of $150m in the company.

His IT department finally settled on the following combination of software

- ▷ Red Hat® Enterprise® Linux
- ▷ OpenOffice Productivity Suite
- ▷ Mozilla Web-browser
- ▷ A number of proprietary applications

Ball reports that the transition was a 'breeze' and ever since has been a very vocal Open Source evangelist.

26.1.2.3 Usage

The company is using Linux based systems for the following tasks

- ▷ Email Servers
- ▷ Email Clients
- ▷ Spreadsheets
- ▷ Word Processing
- ▷ Other job specific tasks

Ball reports having been told that it costs $1,250 per user to change over to open source. He reports that his experience shows the costs are in fact far, far lower.

26.1.2.4 Savings

By Ball's reckoning, the company saved $80,000 in licensing fees just by switching to using Open Source software where available. He's also been able to avoid upgrading hardware as a result of Linux being less resource hungry, machines that couldn't run Microsoft^{TM} Windows® 2000 were still running the new Linux system in 2008!

Ball also suggests that claims in the media about the TCO of Linux are largely false. He reports that he's not yet had a need to make a support call to Red Hat®, and doesn't have to foot the cost of dealing with malware.

He also reports increases in productivity as a result of more granular control over which applications users have available on their workstations (presumably through manipulation of user privileges in the way that we explored earlier in this book).

26.1.2.5 Observations

The example of Ernie Ball Guitar strings highlights not only the potential savings that can be achieved, but also that a business does not need to be technically minded in order to switch. Ball characterises the change as follows "*in three and half years, we went from being these idiots that were thinking emotionally rather than businesslike...to now we're smart and talking to tech guys*".

The situation today, is in fact, even easier. In 2000, Linux was not so well known and very few business centric applications existed for the more niche businesses. Today, the OSS ecosystem is full of potential applications and switching has perhaps never been easier.

The circumstances which brought about the change at Ernie Ball still exist today. Although the BSA are no longer running their "*Nail your boss*" campaign, they do still act upon allegations. A business should always take a BSA audit very, very seriously as license violation is not simply limited to running illicitly downloaded software. Many a business could potentially fall prey to the same mistakes as Ernie Ball, and the reality is that the only way this can be effectively mitigated is to use Open Source Software.

26.1.3 Printed Art

PrintedArt is a small business employing three full-time and three part-time employees and eight sales representatives. The company sells limited edition prints of fine art photography.

The company uses CentOS to run it's customer facing web-server and Ubuntu to run it's internal infrastructure. The companies website uses the Open Source Drupal Content Management system.

The company plans to continue using Linux based systems, but is contemplating the possibility of moving it's webserver from CentOS to Ubuntu in order to streamline the process of applying updates.

26.2 Companies Using Linux

In this section we'll simply be listing other companies which are successfully using Linux as it's not feasible to write a case study for each. By their very nature, most listed are large companies as smaller companies generally lack the time and resources to publicly announce when they are migrating to Linux!

- ▷ The New York Stock Exchange
- ▷ The London Stock Exchange
- ▷ US Department of Defence
- ▷ The Spanish Government
- ▷ The US Federal Aviation Administration
- ▷ The French Parliament
- ▷ ChinaBank
- ▷ Pakistani Schools & Colleges
- ▷ Cuba (Government and schools)
- ▷ Macedonian Ministry of Education and Science
- ▷ US Postal Service
- ▷ US Federal Courts
- ▷ The Government of Mexico City
- ▷ Czech Post
- ▷ Novell
- ▷ Google
- ▷ IBM
- ▷ Panasonic
- ▷ Virgin America
- ▷ Cisco
- ▷ ConocoPhillips
- ▷ Amazon
- ▷ Peugeot

- ▷ Wikipedia
- ▷ Tommy Hilfiger
- ▷ Toyota Motor Sales
- ▷ CERN
- ▷ NASA
- ▷ Continental Airlines
- ▷ Autozone

Each of these companies utilises Linux to some degree, whether to run essential systems or to run all systems. In reality, almost every IT utilising company will be using Linux to some extent. Linux is often used in network routers and switches, as well as on a number of other embedded devices. These, obviously, have been omitted as being largely irrelevant when considering whether to utilise Linux servers/desktops.

Index

Index

About the Author

I am an IT Manager & Linux Specialist working at Virya Technologies. This is my first attempt a book writing, but time permitting there will be more!

I've been managing Linux systems for over 10 years, and in my current role manage numerous Linux and Non-Linux systems on a daily basis. I made a full switch from Microsoft based products more than 9 years ago, and haven't run Windows as my main operating system at home since then.

I still have to manage and maintain Windows Servers in my current role, but as more and more businesses begin to consider Linux based solutions I'm expecting that to slowly reduce.

Copyright Notices

Below are the relevant copyright notices for any assets used within this publication.