# Linux Administrators Security Guide

LASG - 0.1.0

By Kurt Seifried (seifried@seifried.org) copyright 1999, All rights reserved.
Available at: https://www.seifried.org/lasg/

This document is free for most non commercial uses, the license follows the table of contents, please read it if you have any concerns. If you have any questions email seifried@seifried.org. If you want to receive announcements of new versions of the LASG please send a blank email with the subject line “subscribe” (no quotes) to lasg-announce-request@seifried.org.

## Table of contents

License

Preface

Forward by the author

Contributing

What this guide is and isn't

How to determine what to secure and how to secure it

Safe installation of Linux
- Choosing your install media
- It ain't over 'til...

General concepts, server verses workstations, etc

Physical / Boot security
- Physical access
- The computer BIOS
- LILO

The Linux kernel
- Upgrading and compiling the kernel
- Kernel versions

Administrative tools
- Access
  - Telnet
  - SSH
  - LSH
  - REXEC
  - NSH
  - Slush
  - SSL Telnet
  - Fsh
  - secsh
- Local
  - YaST
  - sudo
  - Super
- Remote
  - Webmin
  - Linuxconf
  - COAS

System Files

- /etc/passwd
- /etc/shadow
- /etc/groups
- /etc/gshadow
- /etc/login.defs
- /etc/shells
- /etc/securetty

Log files and other forms of monitoring

- sysklogd / klogd
    - secure-syslog
    - next generation syslog
- Log monitoring
    - logcheck
    - colorlogs
    - WOTS
    - swatch
- Kernel logging
    - auditd
- Shell logging
    - bash

Shadow passwords

- Cracking passwords
    - Jack the ripper
    - Crack
    - Saltine cracker
    - VCU

PAM

Software Management

- RPM
- dpkg
- tarballs / tgz
- Checking file integrity
    - RPM
    - dpkg
    - PGP
    - MD5
- Automatic updates
    - RPM
        - AutoRPM
        - rhlupdate
        - RpmWatch
    - dpkg
        - apt

- tarballs / tgz
- Tracking changes
    - installwatch
    - instmon
- Converting formats
    - alien

File / Filesystem security
- Secure file deletion
    - wipe (thomassr@erols.com)
    - wipe (durakb@crit2.univ-montp2.fr)

TCP-IP and network security
- IPSec
- IPv6
- TCP-IP attack programs
    - HUNT Project

PPP security

Basic network service security
- What is running and who is it talking to?
- PS Output
- Netstat Output
- lsof

Basic network services config files
- inetd.conf
- TCP_WRAPPERS

Network services
- Telnetd
- SSHD
    - Fresh Free FiSSH
    - Tera Term
    - putty
    - mindterm
    - LSH
- RSH, REXEC, RCP
- Webmin
- FTP
    - WuFTPD
- Apache
- SQUID
- SMTP
    - Sendmail
    - Qmail
    - Postfix
    - Zmailer
    - DMail

  - POPD
    - WU IMAPD (stock popd)
    - Cyrus
    - IDS POP
  - IMAPD
    - WU IMAPD (stock imapd)
    - Cyrus
  - WWW based mail readers
    - Non Commercial
      - IMP
      - AtDot
    - Commercial
      - DmailWeb
      - WebImap
  - DNS
    - Bind
    - Dents
  - NNTP
    - INN
    - DNews
  - DHCPD
  - NFSD
  - tftp
    - utftpd
  - bootp
  - cu-snmp
  - Finger
  - Identd
  - ntpd
  - CVS
  - rsync
  - lpd
    - LPRng
    - pdq
  - X Window system
  - SAMBA
    - SWAT

- File sharing methods
  - SAMBA
  - NFS
  - Coda
  - Drall
  - AFS

- Network based authentication
  - NIS / NIS+
  - SRP
  - Kerberos

- Encrypting services / data
  - Encrypting network services
    - SSL
    - HTTP - SSL
    - Telnet - SSL
    - FTP - SSL
    - Virtual private network solutions
      - IPSec
      - PPTP
      - CIPE
      - ECLiPt
  - Encrypting data
    - PGP
    - GnuPG
    - CFS
  - Sources of random data

- Firewalling
  - IPFWADM
  - IPCHAINS
  - Rule Creation
    - ipfwadm2ipchains
    - mason
    - firewall.sh
    - Mklinuxfw

- Scanning / intrusion testing tools
  - Host scanners
    - Cops
    - SBScan
  - Network scanners
    - Strobe
    - nmap
    - MNS
    - Bronc Buster vs. Michael Jackson
    - Leet scanner
    - Soup scanner
    - Portscanner
  - Intrusion scanners
    - Nessus
    - Saint
    - Cheops
    - Ftpcheck / Relaycheck
    - SARA
  - Firewall scanners
    - Firewalk
  - Exploits

- Scanning and intrusion detection tools
  - Logging tools

        - Logcheck
        - Port Sentry
    - Host based attack detection
        - Firewalling
        - TCP_WRAPPERS
        - Klaxon
        - Host Sentry
        - Pikt
    - Network based attack detection
        - NFR

- Host monitoring tools
    - check.pl
    - bgcheck
    - Sxid
    - Viperdb
    - Pikt
    - DTK

- Packet sniffers
    - tcpdump
    - sniffit
    - Ethereal
    - Other sniffers

- Virii, Trojan Horses, Worms, and Social Engineering
    - Disinfection of virii / worms / trojans
    - Virus scanners
        - AMaViS

- Password storage
    - Gpasman

- Conducting baselines / system integrity
    - Tripwire
    - L5
    - Gog&Magog
    - Confcollect
    - Backups

- Conducting audits

- Backups
    - Tar and Gzip
    - Noncommercial Backup programs for Linux
        - Amanda
        - afbackup
    - Commercial Backup Programs for Linux
        - BRU
        - Quickstart

- CTAR
- CTAR:NET
- Backup Professional
- PC ParaChute
- Arkeia
- Legato Networker

Pro's and Con's of Backup Media

Dealing with attacks

Denial of service attacks

Examples of attacks

Distribution specific tools

- SuSE

Distribution specific errata and security lists

- RedHat
- Debian
- Slackware
- Caldera
- SuSE

Internet connection checklist

Appendix A: Books and magazines

Appendix B: URL listing for programs

Appendix C: Other Linux security documentation

Appendix D: Online security documentation

Appendix E: General security sites

Appendix F: General Linux sites

Version History

## License

Terms and Conditions for Copying, Distributing, and Modifying

Items other than copying, distributing, and modifying the Content with which this license was distributed (such as using, etc.) are outside the scope of this license.

The 'guide' is defined as the documentation and knowledge contained in this file.

1. You may copy and distribute exact replicas of the guide as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the guide a copy of this License along with the guide. You may at your option charge a fee for the media and/or handling involved in creating a unique copy of the guide for use offline, you may at your option offer instructional support for the guide in exchange for a fee, or you may at your option offer warranty in exchange for a fee. You may not charge a fee for the guide itself. You may not charge a fee for the sole service of providing access to and/or use of the guide via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the guide. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the guide, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the guide.

NO WARRANTY

3. BECAUSE THE GUIDE IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE GUIDE, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE GUIDE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE GUIDE IS WITH YOU. SHOULD THE GUIDE PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

4. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE GUIDE AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE GUIDE, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## Preface

Since this is an electronic document, changes will be made on a regular basis, and feedback is greatly appreciated. The author is available at:

Kurt Seifried
**seifried@seifried.org**
(780) 453-3174

My Verisign Class 2 digital ID public key

```
-----BEGIN CERTIFICATE-----
MIIDtzCCAyCgAwIBAgIQO8AwExKJ74akljwwoX4BrDANBgkqhkiG9w0BAQQFADCB
uDEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xHzAdBgNVBAsTFlZlcmlTaWduIFRy
dXN0IE5ldHdvcmsxRjBEBgNVBAsTPXd3dy52ZXJpc2lnbi5jb20vcmVwb3NpdG9y
eS9SUEEgSW5jb3JwLiBCeSBSZWYuLExJQUIuTFREKGMpOTgxNDAyBgNVBAMTK1Zl
cmlTaWduIENsYXNzIDIgQ0EgLSBJbmRpdmlkdWFsIFN1YnNjcmliZXIwHhcNOTgx
MDIxMDAwMDAwWhcNOTkxMDIxMjM1OTU5WjCB6TEXMBUGA1UEChMOVmVyaVNpZ24s
IEluYy4xHzAdBgNVBAsTFlZlcmlTaWduIFRydXN0IE5ldHdvcmsxRjBEBgNVBAsT
PXd3dy52ZXJpc2lnbi5jb20vcmVwb3NpdG9yeS9SUEEgSW5jb3JwLiBieSBSZWYu
LExJQUIuTFREKGMpOTgxJzAlBgNVBAsTHkRpZ2l0YWwgSUQgQ2xhc3MgMiAtIE1p
Y3Jvc29mdDEWMBQGA1UEAxQNS3VydCBTZWlmcmllZDEkMCIGCSqGSIb3DQEJARYV
c2VpZnJpZWRAc2VpZnJpZWQub3JnMFswDQYJKoZIhvcNAQEBBQADSgAwRwJAZsvO
hR/FIDH8V2MfrIU6edLc98xk0LYA7KZ2xx81hPPHYNvbJe0ii2fwNoye0DThJal7
bfqRI2OjRcGRQt5wlwIDAQABo4HTMIHQMAkGA1UdEwQCMAAwga8GA1UdIASBpzCA
MIAGC2CGSAGG+EUBBwEBMIAwKAYIKwYBBQUHAgEWHGh0dHBzOi8vd3d3LnZlcmlz
aWduLmNvbS9DUFMwYgYIKwYBBQUHAgIwVjAVFg5WZXJpU2lnbiwgSW5jLjADAgEB
Gj1WZXJpU2lnbidzIENQUyBpbmNvcnAuIGJ5IHJlZmVyZW5jZSBsaWFiLiBsdGQu
IChjKTk3IFZlcmlTaWduAAAAAAAAMBEGCWCGSAGG+EIBAQQEAwIHgDANBgkqhkiG
9w0BAQQFAAOBgQAwfnV6AKAetmcIs8lTkgp8/KGbJCbL94adYgfhGJ99M080yhCk
yNuZJ/o6L1VlQCxjntcwS+VMtMziJNELDCR+FzAKxDmHgal4XCinZMHp8YdqWsfC
wdXnRMPqEDW6+6yDQ/pi84oIbP1ujDdajN141YLuMz/c7JKsuYCKkk1TZQ==
-----END CERTIFICATE-----
```

I sign all my email with that certificate, so if it isn’t signed, it isn’t from me. Feel free to encrypt email to me with my certificate, I’m trying to encourage world-wide secure email (doesn’t seem to be working though).

To receive updates about this book please subscribe to the announcements email list, don't expect an email everytime I release a new version of the guide (this list is for 'stable releases' of the guide). Send an email to: **lasg-announce-request@seifried.org** with the Subject line containing the word "**subscribe**" (no quotes) and you will automatically be placed on the list. To unsubscribe send an email with the word “**unsubscribe**” (no quotes) in the Subject line. Otherwise take a look at https://www.seifried.org/lasg/ once in a while to see if I announce anything.

## **Forward by the author**

I got my second (our first doesn’t count, a TRS-80 that died after a few months) computer in Christmas of 1993, blew windows away 4 months later for OS/2, got a second computer in spring of 1994, loaded Linux on it (Slackware 1.?) in July of 1994. I ran Slackware for about 2-3 years and switched to RedHat after being introduced to it, after 2-3 months of RedHat exposure I switched over to it. Since then I have also earned an MCSE and MCP+Internet (come to the dark side Luke...). Why did I write this guide? Because no-one else. Why is it freely available online? Because I want to reach the largest audience possible.

I have also received help on this guide (both direct and indirect) from the Internet community at large, many people have put up excellent security related webpages that I list, and mailing lists like Bugtraq help me keep on top of what is happening. It sounds cliched (and god forbid a journalist pick this up) but this wouldn't be possible without the open source community. I thank you all.

## Contributing

Contributions are welcome, especially URL’s for programs/resources that aren’t listed here yet. As for actual contributions of written material I cannot accept those at yet for a variety of reasons.

## **What this guide is and isn't**

This guide is not a general security document. This guide is specifically about securing the Linux operating system against general and specific threats. If you need a general overview of security please go buy "Practical Unix and Internet Security" available at www.ora.com. O'Reilly and associates, which is one of my favorite publisher of computer books (they make nice T-shirts to) and listed in the appendix are a variety of other computer books I recommend.

## How to determine what to secure and how to secure it

Are you protecting data (proprietary, confidential or otherwise), are you trying to keep certain services up (your mail server, www server, etc.), do you simply want to protect the physical hardware from damage? What are you protecting it against? Malicious damage (8 Sun Enterprise 10000's), deletion (survey data, your mom's recipe collection), changes (a hospital with medical records, a bank), exposure (confidential internal communications concerning the lawsuit, plans to sell cocaine to unwed mothers), and so on. What are the chances of a "bad" event happening, network probes (happens to me daily), physical intrusion (hasn't happened to me yet), social engineering ("Hi, this is Bob from IT, I need your password so we can reset it….").

You need to list out the resources (servers, services, data and other components) that contain data, provide services, make up your company infrastructure, and so on. The following is a short list:

- Physical server machines
- Mail server and services
- DNS server and services
- WWW server and services
- File server and services
- Internal company data such as accounting records and HR data
- Your network infrastructure (cabling, hubs, switches, routers, etc.)
- Your phone system (PBX, voicemail, etc.)

You then need to figure out what you want to protect it against:

- Physical damage (smoke, water, food, etc.)
- Deletion / modification of data (accounting records, defacement of your www site, etc.)
- Exposure of data (accounting data, etc.)
- Continuance of services (keep the email/www/file server up and running)
- Prevent others from using your services illegally/improperly (email spamming, etc.)

Finally what is the likelihood of an event occurring?

- Network scans – daily is a safe bet
- Social engineering – varies, usually the most vulnerable people tend to be the ones targeted
- Physical intrusion – depends, typically rare, but a hostile employee with a pair of wire cutters could do a lot of damage in a telecom closet
- Employees selling your data to competitors – it happens
- Competitor hiring skilled people to actively penetrate your network – no-one ever talks about this one but it also happens

Once you have come up with a list of your resources and what needs to be done you can start implementing security. Some techniques (physical security for servers, etc.) pretty much go without saying, in this industry there is a baseline of security typically implemented (passwording accounts, etc.). The vast majority of security problems are usually human generated, and most problems I have seen are due to a lack of education/communication between people, there is no technical ‘silver bullet’, even the best software needs to be installed, configured and maintained by people.

Now for the stick. A short list of possible results from a security incident:

- Loss of data
- Direct loss of revenue (www sales, file server is down, etc)
- Indirect loss of revenue (email support goes, customers vow never to buy from you again)
- Cost of staff time to respond
- Lost productivity of IT staff and workers dependant on IT infrastructure
- Legal Liability (medical records, account records of clients, etc.)
- Loss of customer confidence
- Media coverage of the event

## Safe installation of Linux

A proper installation of Linux is the first step to a stable, secure system. There are various tips and tricks to make the install go easier, as well as some issues that are best handled during the install (such as disk layout).

### Choosing your install media

This is the #1 issue that will affect speed of install and to a large degree safety. My personal favorite is ftp installs since popping a network card into a machine temporarily (assuming it doesn't have one already) is quick and painless, and going at 1+ megabyte/sec makes for quick package installs. Installing from CD-ROM is generally the easiest, as they are bootable, Linux finds the CD and off you go, no pointing to directories or worrying about case (in the case of an HD install). This is also original Linux media and you can be relatively sure it is safe (assuming it came from a reputable source), if you are paranoid however feel free to check the signatures on the files.

- FTP - quick, requires network card, and an ftp server (Windows box running something like warftpd will work as well).
- HTTP – also fast, and somewhat safer then running a public FTP server for installs
- Samba - quick, good way if you have a windows machine (share the cdrom out).
- NFS - not as quick, but since nfs is usually implemented in most existing UNIX networks (and NT now has an NFS server from MS for free) it's mostly painless. NFS is the only network install supported by RedHat’s kickstart.
- CDROM - if you have a fast cdrom drive, your best bet, pop the cd and boot disk in, hit enter a few times and you are done. Most Linux CDROM’s are now bootable.
- HardDrive - generally the most painful, windows kacks up filenames/etc, installing from an ext2 partition is usually painless though (catch 22 for new users however).

### It ain't over 'til...

So you've got a fresh install of Linux (RedHat, Debian, whatever, please, please, DO NOT install really old versions and try to upgrade them, it's a nightmare), but chances are there is a lot of extra software installed, and packages you might want to upgrade or things you had better upgrade if you don't want the system compromised in the first 15 seconds of uptime (in the case of BIND/Sendmail/etc.). Keeping a local copy of the updates directory for your distributions is a good idea (there is a list of errata for distributions at the end of this document), and making it available via nfs/ftp or burning it to CD is generally the quickest way to make it available. As well there are other items you might want to upgrade, for instance I use a chroot'ed, non-root version of Bind 8.1.2, available on the contrib server (ftp://contrib.redhat.com/), instead of the stock, non-chrooted, run as root Bind 8.1.2 that ships with RedHat Linux. You will also want to remove any software you are not using, and/or replace it with more secure versions (such as replacing rsh with ssh).

## General concepts, server verses workstations, etc

There are many issues that affect actually security setup on a computer. How secure does it need to be? Is the machine networked? Will there be interactive user accounts (telnet/ssh)? Will users be using it as a workstation or is it a server? The last one has a big impact since "workstations" and "servers" have traditionally been very different beasts, although the line is blurring with the introduction of very powerful and cheap PC's, as well as operating systems that take advantage of them. The main difference in today's world between computers is usually not the hardware, or even the OS (Linux is Linux, NT Server and NT Workstation are close family, etc.), it is in what software packages are loaded (apache, X, etc) and how users access the machine (interactively, at the console, and so forth). Some general rules that will save you a lot of grief in the long run:

1. Keep users off of the servers. That is to say: do not give them interactive login shells, unless you absolutely must.
2. Lock down the workstations, assume users will try to 'fix' things (heck, they might even be hostile, temp workers/etc).
3. Use encryption wherever possible to keep plain text passwords, credit card numbers and other sensitive information from lying around.
4. Regularly scan the network for open ports/installed software/etc that shouldn't be, compare it against previous results..

Remember: security is not a solution, it is a way of life.

Generally speaking workstations/servers are used by people that don't really care about the underlying technology, they just want to get their work done and retrieve their email in a timely fashion. There are however many users that will have the ability to modify their workstation, for better or worse (install packet sniffers, warez ftp sites, www servers, irc bots, etc). To add to this most users have physical access to their workstations, meaning you really have to lock them down if you want to do it right.

1. Use BIOS passwords to lock users out of the BIOS (they should never be in here, also remember that older BIOS's have universal passwords.)
2. Set the machine to boot from the appropriate harddrive only.
3. Password the LILO prompt.
4. Do not give the user root access, use sudo to tailor access to privileged commands as needed.
5. Use firewalling so even if they do setup services they won't be accessible to the world.
6. Regularly scan the process table, open ports, installed software, and so on for change.
7. Have a written security policy that users can understand, and enforce it.
8. Remove all sharp objects (compilers, etc) unless needed from a system.

Remember: security in depth.

Properly setup, a Linux workstation is almost user proof (nothing is 100% secure), and generally a lot more stable then a comparable Wintel machine. With the added joy of remote administration (SSH/Telnet/NSH) you can keep your users happy and productive.

Servers are a different ball of wax together, and generally more important then workstations (one workstation dies, one user is affected, if the email/www/ftp/etc server dies your boss phones up in a bad mood). Unless there is a strong need, keep the number of users with interactive shells (bash, pine, lynx based, whatever) to a bare minimum. Segment services up (have a mail server, a www server, and so on) to minimize single point of failure. Generally speaking a properly setup server will run and not need much maintenance (I have one email server at a client location that has been in use for 2 years with about 10 hours of maintenance in total). Any upgrades should be planned carefully and executed on a test. Some important points to remember with servers:

1. Restrict physical access to servers.
2. Policy of least privilege, they can break less things this way.
3. MAKE BACKUPS!
4. Regularly check the servers for changes (ports, software, etc), automated tools are great for this.
5. Software changes should be carefully planned/tested as they can have adverse affects (like kernel 2.2.x no longer uses ipfwadm, wouldn't that be embarrassing if you forgot to install ipchains).

Minimization of privileges means giving users (and administrators for that matter) the minimum amount of access required to do their job. Giving a user "root" access to their workstation would make sense if all users were Linux savvy, and trustworthy, but they generally aren't (on both counts). And even if they were it would be a bad idea as chances are they would install some software that is broken/insecure or other. If all a user access needs to do is shutdown/reboot the workstation then that is the amount of access they should be granted. You certainly wouldn't leave accounting files on a server with world readable permissions so that the accountants can view them, this concept extends across the network as a whole. Limiting access will also limit damage in the event of an account penetration (have you ever read the post-it notes people put on their monitors?).

## Physical / Boot security

### Physical Access

This area is covered in depth in the "Practical Unix and Internet Security" book, but I'll give a brief overview of the basics. Someone turns your main accounting server off, turns it back on, boots it from a specially made floppy disk and transfers payroll.db to a foreign ftp site. Unless your accounting server is locked up what is to prevent a malicious user (or the cleaning staff of your building, the delivery guy, etc.) from doing just that? I have heard horror stories of cleaning staff unplugging servers so that they could plug their cleaning equipment in. I have seen people accidentally knock the little reset switch on power bars and reboot their servers (not that I have ever done that). It just makes sense to lock your servers up in a secure room (or even a closet). It is also a very good idea to put the servers on a raised surface to prevent damage in the event of flooding (be it a hole in the roof or a super gulp slurpee).

### The Computer BIOS

The computer's BIOS is on of the most low level  components, it controls how the computer boots and a variety of other things. Older bios's are infamous for having universal passwords, make sure your bios is recent and does not contain such a backdoor. The bios can be used to lock the boot sequence of a computer to C: only, i.e. the first harddrive, this is a very good idea. You should also use the bios to disable the floppy drive (typically a server will not need to use it), and it can prevent users from copying data off of the machine onto floppy disks. You may also wish to disable the serial ports in users machines so that they cannot attach modems, most modern computers use PS/2 keyboard and mice, so there is very little reason for a serial port in any case (plus they eat up IRQ's). Same goes for the parallel port, allowing users to print in a fashion that bypasses your network, or giving them the chance to attach an external CDROM burner or harddrive can decrease security greatly. As you can see this is an extension of the policy of least privilege and can decrease risks considerably, as well as making network maintenance easier (less IRQ conflicts, etc.).

### LILO

Once the computer has decided to boot from C:, LILO (or whichever bootloader you use) takes over. Most bootloaders allow for some flexibility in how you boot the system, LILO especially so, but this is a two edged sword. You can pass LILO arguments at boot time, the most damaging (from a security point of view) being "`imagename single`" which boots Linux into single user mode, and by default in most distributions dumps you to a root prompt in a command shell with no prompting for passwords or other pesky security mechanisms. Several techniques exist to minimize this risk.

```
delay=X
```

this controls how long (in tenths of seconds) LILO waits for user input before booting to the default selection. One of the requirements of C2 security is that this interval be set to 0 (obviously a dual boot machines blows most security out of the water). It is a good idea to set this to 0 unless the system dual boots something else.

```
prompt
```

forces the user to enter something, LILO will not boot the system automatically. This could be useful on servers as a way of disabling reboots without a human attendant, but typically if the hacker has the ability to reboot the system they could rewrite the MBR with new boot options. If you add a timeout option however the system will continue booting after the timeout is reached.

```
restricted
```

requires a password to be used if boot time options (such as "`linux single`") are passed to the boot loader. Make sure you use this one on each image (otherwise the server will need a password to boot, which is fine if you're never planning to remotely reboot it).

```
password=XXXXX
```

requires user to input a password, used in conjunction with restricted, also make sure lilo.conf is no longer world readable, or any user will be able to read the password.

Here is an example of lilo.conf from one of my servers (the password has been of course changed).

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
default=linux
image=/boot/vmlinuz-2.2.5
        label=linux
        root=/dev/hda1
        read-only
       restricted
       password=some_password
```

This boots the system using the `/boot/vmlinuz-2.2.5` kernel, stored on the MBR of the first IDE harddrive of the system, the prompt keyword would normally stop unattended rebooting, however it is set in the image, so it can boot "`linux`" no problem, but it would ask for a password if you entered "`linux single`", so if you want to go into "`linux single`" you have 10 seconds to type it in, at which point you would be prompted for the password ("`some_password`"). Combine this with a BIOS set to only boot from C: and password protected and you have a pretty secure system.

## The Linux kernel

Linux (GNU/Linux according to Stallman if you’re referring to a complete Linux distribution) is actually just the kernel of the operating system. The kernel is the core of the system, it handles access to all the harddrive, security mechanisms, networking and pretty much everything. It had better be secure or you are screwed.

In addition to this we have problems like the Pentium F00F bug and inherent problems with the TCP-IP protocol, the Linux kernel has it’s work cut out for it. Kernel versions are labeled as X.Y.Z, Z are minor revision numbers, Y define if the kernel is a test (odd number) or production (even number), and X defines the major revision (we have had 0, 1 and 2 so far). I would highly recommend running kernel 2.2.x, as of May 1999 this is 2.2.9. The 2.2.x series of kernel has major improvements over the 2.0.x series. Using the 2.2.x kernels also allows you access to newer features such as ipchains (instead of ipfwadm) and other advanced security features.

### Upgrading and Compiling the Kernel

Upgrading the kernel consists of getting a new kernel and modules, editing /etc/lilo.conf, rerunning lilo to write a new MBR. The kernel will typically be placed into /boot, and the modules in /lib/modules/kernel.version.number/.

Getting a new kernel and modules can be accomplished 2 ways, by downloading the appropriate kernel package and installing it, or by downloading the source code from ftp://ftp.kernel.org/ (please use a mirror site), and compiling it.

Compiling a kernel is straightforward:

```
cd /usr/src
```

there should be a symlink called “linux” pointing to the directory containing the current kernel, remove it if there is, if there isn’t one no problem. You might want to ‘mv’ the linux directory to /usr/src/linux-kernel.version.number and create a link pointing /usr/src/linux at it.

Unpack the source code using tar and gzip as appropriate so that you now have a /usr/src/linux with about 50 megabytes of source code in it. The next step is to create the linux kernel configuration (/usr/src/linux.config), this can be achieved using “make config”, “make menuconfig” or “make xconfig”, my preferred method is “make menuconfig” (for this you will need ncurses and ncurses devel libraries). This is arguably the hardest step, there are hundreds options, which can be categorized into two main areas: hardware support, and service support. For hardware support make a list of hardware that this kernel will be running on (i.e. P166, Adaptec 2940 SCSI Controller, NE2000 ethernet card, etc.) and turn on the appropriate options. As for service support you will need to figure out which filesystems (fat, ext2, minix ,etc.) you plan to use, the same for networking (firewalling, etc.).

Once you have configured the kernel you need to compile it, the following commands makes dependencies ensuring that libraries and so forth get built in the right order, then cleans out any information from previous compiles, then builds a kernel, the modules and installs the modules.

`make dep` (makes dependencies)
`make clean` (cleans out previous cruft)
`make bzImage` (make zImage pukes if the kernel is to big, and 2.2.x kernels tend to be pretty big)
`make modules` (creates all the modules you specified)
`make modules_install` (installs the modules to /lib/modules/kernel.version.number/)

you then need to copy `/usr/src/linux/arch/i386/boot/bzImage`(zImage) `to /boot/vmlinuz-kernel.version.number.` Then edit /etc/lilo.conf, adding a new entry for the new kernel and setting it as the default image is the safest way (using the `default=X` command, otherwise it will boot the first kernel listed), if it fails you can reboot and go back to the previous working kernel. Run lilo, and reboot.

### Kernel Versions

Currently we are in a stable kernel release series, 2.2.x. I would highly recommend running the latest stable kernel (currently 2.2.9 as of May 1999) as there are several nasty security problems (network attacks and denial of service attacks) that affect all kernels up to 2.0.35, 2.0.36 is patched, and the later 2.1.x test kernels to 2.2.3. Upgrading from the 2.0.x series of stable kernels to the 2.2.x series is relatively painless if you are careful and follow instructions (there are some minor issues but for most users it will go smoothly). Several software packages must be updated, libraries, ppp, modutils and others (they are covered in the kernel docs / rpm dependencies / etc.). Additionally keep the old working kernel, add an entry in lilo.conf for it as "linuxold" or something similar and you will be able to easily recover in the event 2.2.x doesn't work out as expected. Don't expect the 2.2.x series to be bug free, 2.2.9 will be found to contain flaws and will be obsoleted, like every piece of software in the world.

## Administrative tools

### Access

Telnet

Telnet is by far the oldest and well known remote access tool, virtually ever Unix ships with it, and even systems such as NT support it. Telnet is really only useful if you can administer the system from a command prompt (something NT isn’t so great at), which makes it perfect for Unix. Telnet is incredibly insecure, passwords and usernames as well as the session data flies around as plain text and is a favourite target for sniffers. Telnet comes with all Linux distributions. You should never ever use stock telnet to remotely administer a system.

SSL Telnet

SSL Telnet is telnet with the addition of SSL encryption which makes it much safer and far more secure. Using X.509 certificates (also referred to as personal certificates) you can easily administer remote systems. Unlike systems such as SSH, SSL Telnet is completely GNU and free for all use. You can get SSL Telnet server and client from: ftp://ftp.replay.com/.

SSH

SSH was originally free but is now under a commercial license, it does however have many features that make it worthwhile. It supports several forms of authentication (password, rhosts based, RSA keys), allows you to redirect ports, and easily configure which users are allowed to login using it. SSH is available from: ftp://ftp.replay.com/. If you are going to use it commercially, or want the latest version you should head over to: http://www.ssh.fi/.

LSH

LSH is a free implementation of the SSH protocol, LSH is GNU licensed and is starting to look like the alternative (commercially speaking) to SSH (which is not free anymore). You can download it from: http://www.net.lut.ac.uk/psst/, please note it is under development.

REXEC

REXEC is one of the older remote UNIX utilities, it allows you to execute commands on a remote system, however it is seriously flawed in that it has no real security model. Security is achieved via the use of “rhosts” files, which specify which hosts/etc may run commands, this however is prone to spoofing and other forms of exploitation. You should never ever use stock REXEC to remotely administer a system.

Slush

Slush is based on OpenSSL and supports X.509 certificates currently, which for a large organization is a much better (and saner) bet then trying to remember several dozen passwords on various servers. Slush is GPL, but not finished yet (it implements most of the required functionality to be useful, but has limits). On the other hand it is based completely in open source software making the possibilities of backdoors/etc remote. Ultimately it could replace SSH with something much nicer. You can get it from: http://violet.ibs.com.au/slush/.

### NSH

NSH is a commercial product with all the bells and whistles (and I do mean all). It's got built in support for encryption, so it's relatively safe to use (I cannot really verify this as it isn't open source). Ease of use is high, you cd //computername and that 'logs' you into that computer, you can then easily copy/modify/etc. files, run ps and get the process listing for that computer, etc. NSH also has a Perl module available, making scripting of commands pretty simple, and is ideal for administering many like systems (such as workstations). In addition to this NSH is available on multiple platforms (Linux, BSD, Irix, etc.). NSH is available from: http://www.networkshell.com/, and 30 day evaluation versions are easily downloaded.

### Fsh

Fsh is stands for "Fast remote command execution" and is similar in concept to rsh/rcp. It avoids the expense of constantly creating encrypted sessions by bring up an encrypted tunnel using ssh or lsh, and running all the commands over it. You can get it from: http://www.lysator.liu.se/fsh/.

### secsh

secsh (Secure Shell) provides another layer of login security, once you have logged in via ssh or SSL telnet you are prompted for another password, if you get it wrong secsh kills off the login attempt. You can get secsh at: http://www.leenux.com/scripts/.

## Local

### YaST

YaST (Yet Another Setup Tool) is a rather nice command line graphical interface (very similar to scoadmin) that provides an easy interface to most administrative tasks. It does not however have any provisions for giving users limited access, so it is really only useful for cutting down on errors, and allowing new users to administer their systems. Another problem is unlike Linuxconf it is not network aware, meaning you must log into each system you want to manipulate.

### sudo

Sudo gives a user setuid access to a program(s), and you can specify which host(s) they are allowed to login from (or not) and have sudo access (thus if someone breaks into an account, but you have it locked down damage is minimized). You can specify what user a command will run as, giving you a relatively fine degree of control. If granting users access be sure to specify the hosts they are allowed to log in from and execute sudo, as well give the full pathnames to binaries, it can save you significant grief in the long run (i.e. if I give a user setuid access to "adduser", there is nothing to stop them editing their path statement, and copying "bash" into /tmp). This tool is very similar to super but with slightly less fine control. Sudo is available for most distributions as a core package or a contributed package. Sudo is available at: http://www.courtesan.com/sudo/ just in case your distribution doesn't ship with it Sudo allows you to define groups of hosts, groups of commands, and groups of users, making long term administration simpler. Several `/etc/sudoers` examples:

Give the user ‘seifried’ full access

```
seifried ALL=(ALL) ALL
```

Create a group of users, a group of hosts, and allow then to shutdown the server as root

```
Host_Alias WORKSTATIONS=localhost, station1, station2
User_Alias SHUTDOWNUSERS=bob, mary, jane
Cmnd_Alias REBOOT=halt, reboot, sync
Runas_Alias REBOOTUSER=admin
SHUTDOWNUSERS WORKSTATIONS=(REBOOTUSER) REBOOT
```

Super

Super is one of the very few tools that can actually be used to give certain users (and groups) varied levels of access to system administration. In addition to this you can specify times and allow access to scripts, giving setuid access to even ordinary commands could have unexpected consequences (any editor, any file manipulation tools like chown, chmod, even tools like lp could compromise parts of the system). Debian ships with super, and there are rpm's available in the contrib directory (buildhost is listed as "`localhost`", you might want to find the source and compile it yourself). This is a very powerful tool (it puts sudo to shame), but requires a significant amount of effort to implement properly, I think it is worth the effort though. The head end distribution site for super is at: ftp://ftp.ucolick.org/pub/users/will/.

## Remote

Webmin

Webmin is a (currently) a non commercial web based administrative tool. It’s a set of perl scripts with a self contained www server that you access using a www browser, it has modules for most system administration functions, although some are a bit temperamental. One of my favourite features is the fact is that it holds it’s own username and passwords for access to webmin, and you can customize what each user gets access to (i.e. user1 can administer users, user2 can reboot the server, and user3 can fiddle with the apache settings). Webmin is available at: http://www.webmin.com/.

Linuxconf

Linuxconf is a general purpose Linux administration tool that is usable from the command line, from within X, or via it's built in www server. It is my preferred tool for automated system administration (I primarily use it for doing strange network configurations), as it is relatively light from the command line (it is actually split up into several modules). From within X it provides an overall view of everything that can be configured (PPP, users, disks, etc.). To use it via a www browser you must first run Linuxconf on the machine and add the host(s) or network(s) you want to allow to connect (Conf > Misc > Linuxconf network access), save changes and quit, then when you connect to the machine (by default Linuxconf runs on port 98) you must enter a username and password, it only accepts root as the account, and Linuxconf doesn't support any encryption, so I would have to recommend very strongly against using this feature across public networks. Linuxconf ships with RedHat Linux and is available at: http://www.solucorp.qc.ca/linuxconf/. Linuxconf also doesn't seem to ship with any man pages/etc, the help is contained internally which is slightly irritating.

## COAS

The COAS project (Caldera Open Administration System) is a very ambitious project to provide an open framework for administering systems, from a command line (with semi graphical interface), from within X (using the qt widget set) to the web. It abstracts the actual configuration data by providing a middle layer, thus making it suitable for use on disparate Linux platforms. Version 1.0 was just released, so it looks like Caldera is finally pushing ahead with it. The COAS site is at: http://www.coas.org/.

## System Files

### /etc/passwd

The password file is arguably the most critical system file in Linux (and most other unices). It contains the mappings of username, user ID and the primary group ID that person belongs to. It may also contain the actual password however it is more likely (and much more secure) to use shadow passwords to keep the passwords in `/etc/shadow`. This file MUST be world readable, otherwise commands even as simple as ls will fail to work properly. The GECOS field can contain such data as the real name, phone number and the like for the user, the home directory is the default directory the user gets placed in if they log in interactively, and the login shell must be an interactive shell (such as bash, or a menu program) and listed in `/etc/shells` for the user to log in. The format is:

```
username:password:UID:GID:GECOS_field:home_directory:login_shell
```

### /etc/shadow

The shadow file holes the username and password pairs, as well as account information such as expiry date, and any other special fields. This file should be protected at all costs.

### /etc/groups

The groups file contains all the group membership information, and optional items such as group password (typically stored in gshadow on current systems), this file to must be world readable for the system to behave correctly. The format is:

```
groupname:password:GID:member,member,member
```

A group may contain no members (i.e. it is unused), a single member or multiple members, and the password is optional.

### /etc/gshadow

Similar to the password shadow file, this file contains the groups, password and members.

### /etc/login.defs

This file (`/etc/logins.def`) allows you to define some useful default values for various programs such as useradd and password expiry.  It tends to vary slightly across distributions and even versions, but typically is well commented and tends to contain sane default values.

### /etc/shells

The shells file contains a list of valid shells, if a user’s default shell is not listed here they may not log in interactively. See the section on Telnetd for more information.

### /etc/securetty

This file contains a list of tty’s that root can log in from. Console tty’s are usually `/dev/tty1` through `/dev/tty6`. Serial ports (if you want to log in as root over a modem say) are `/dev/ttyS0` and up typically. If you want to allow root to login via the network (a very bad idea, use sudo) then add `/dev/ttyp1` and up (if 30 users login and root tries to login root will be coming from `/dev/ttyp31`). Generally you should only allow root to login from `/dev/tty1`, and it  is advisable to disable the root account altogether.

## Log files and other forms of monitoring

One integral part of any UNIX system are the logging facilities. The majority of logging in Linux is provided by two main programs, sysklogd and klogd, the first providing logging services to programs and applications, the second providing logging capability to the Linux kernel. Klogd actually sends most messages to the syslogd facility but will on occasion pop up messages at the console (i.e. kernel panics). Sysklogd actually handles the task of processing most messages and sending them to the appropriate file or device, this is configured from within `/etc/syslog.conf`. By default most logging to files takes place in `/var/log/`, and generally speaking programs that handle their own logging (such as apache) log `to /var/log/progname/`, this centralizes the log files and makes it easier to place them on a separate partition (some attacks can fill your logs quite quickly, and a full / partition is no fun). Additionally there are programs that handle their own interval logging, one of the more interesting being the `bash` command shell. By default bash keeps a history file of commands executed in `~username/.bash_history`, this file can make for extremely interesting reading, as oftentimes many admins will accidentally type their passwords in at the command line. Apache handles all of it's logging internally, configurable from `httpd.conf` and extremely flexible with the release of Apache 1.3.6 (it supports conditional logging). Sendmail handles it's logging requirements via syslogd but also has the option (via the command line -X switch) of logging all SMTP transactions straight to a file. This is highly inadvisable as the file will grow enormous in a short span of time, but is useful for debugging. See the sections in network security on apache and sendmail for more information.

### sysklogd / klogd

In a nutshell klogd handles kernel messages, depending on your setup this can range from almost none to a great deal if for example you turn on process accounting. It then passes most messages to syslogd for actual handling, i.e. placement in a logfile. the man pages for sysklogd, klogd and syslog.conf are pretty good with clear examples. One exceedingly powerful and often overlooked ability of syslog is to log messages to a remote host running syslog. Since you can define multiple locations for syslog messages (i.e. send all kern messages to the `/var/log/messages` file, and to console, and to a remote host or multiple remote hosts) this allows you to centralize logging to a single host and easily check log files for security violations and other strangeness. There are several problems with syslogd and klogd however, the primary ones being the ease of which once an attacker has gained root access to deleting/modifying log files, there is no authentication built into the standard logging facilities.

The standard log files that are usually defined in syslog.conf are:

```
/var/log/messages
/var/log/secure
/var/log/maillog
/var/log/spooler
```

The first one (messages) gets the majority of information typically, user login's, TCP_WRAPPERS dumps information here, IP firewall packet logging typically dumps information here and so on. The second typically records entries for events like users changing their UID/GID (via su, sudo, etc.), failed attempts when passwords are required and so on. The maillog file typically holds entries for every pop/imap connection (user login and logout), and the header of each piece of email that goes in or out of the system (from whom, to where, msgid, status, and so on). The spooler file is not often used anymore as the number of people running usenet or uucp has plummeted, uucp has been basically replaced with ftp and email, and most usenet servers are typically extremely powerful machines to handle a full, or even partial newsfeed, meaning there aren't many of them (typically one per ISP or more depending on size). Most home users and small/medium sized business will not (and should not in my opinion) run a usenet server, the amount of bandwidth and machine power required is phenomenal, let alone the security risks.

You can also define additional log files, for example you could add:

```
kern.*                                                   /var/log/kernel-log
```

And/or you can log to a separate log host:

```
*.emerg                        @syslog-host
mail.*                         @mail-log-host
```

Which would result in all kernel messages being logged to /var/log/kernel-log, this is useful on headless servers since by default kernel messages go to /dev/console (i.e. someone logged in at the machines). In the second case all emergency messages would be logged to the host “syslog-host”, and all the mail log files would be sent to the “mail-log-host” server, allowing you to easily maintain centralized log files of various services.

secure-syslog

The major problem with syslog however is that tampering with log files is trivial. There is however a secure versions of syslogd, available at http://www.core-sdi.com/ssyslog/ (these guys generally make good tools and have a good reputation, in any case it is open source software for those of you truly paranoid). This allows you to cyrptographically sign logs and other ensure they haven’t been tampered with, ultimately however an attacker can still delete the log files so it is a good idea to send them to another host, especially in the case of a firewall to prevent the hard drive being filled up.

next generation syslog

Another alternative is “syslog-ng” (Next Generation Syslog), which seems much more customizable then either syslog or secure syslog, it supports digital signatures to prevent log tampering, and can filter based on content of the message, not just the facility it comes from or priority (something that is very useful for cutting down on volume). Syslog-ng is available at: http://www.balabit.hu/products/syslog-ng.html.

**Log monitoring**

logcheck

logcheck will go through the messages file (and others) on a regular basis (invoked via crontab usually) and email out a report of any suspicious activity. It is easily configurable with several ‘classes’ of items, active penetration attempts which is screams about immediately, bad activity, and activity to be ignored (for example DNS server statistics or SSH rekeying). Logcheck is available from: http://www.psionic.com/abacus/logcheck/.

colorlogs

colorlogs will color code log lines allowing you to easily spot bad activity. It is of somewhat questionable value however as I know very few people that stare at log files on an on-going basis. You can get it at: http://www.resentment.org/projects/colorlogs/.

WOTS

WOTS collects log files from multiple sources and will generate reports or take action based on what you tell it to do. WOTS looks for regular expressions you define and then executes the commands you list (mail a report, sound an alert, etc.). WOTS requires you have perl installed and is available from: http://www.vcpc.univie.ac.at/~tc/tools/.

swatch

swatch is very similar to WOTS, and the log files configuration is very similar. You can download swatch from: ftp://ftp.stanford.edu/general/security-tools/swatch/

## Kernel logging

auditd

auditd allows you to use the kernel logging facilities (a very powerful tool). You can log mail messages, system events and the normal items that syslog would cover, but in addition to this you can cover events such as specific users opening files, the execution of programs, of setuid programs, and so on. If you need a solid audit trail then this is the tool for you, you can get it at: ftp://ftp.hert.org/pub/linux/auditd/.

## Shell logging

bash

I will also cover bash since it is the default shell in most Linux installations, and thus it's logging facilities are generally used. bash has a large number of variables you can configure at or during run time that modify how it behaves, everything from the command prompt style to how many lines to keep in the log file.

`HISTFILE`
name of the history file, by default it is ~username/.bash_history

`HISTFILESIZE`
maximum number of commands to keep in the file, it rotates them as needed.

`HISTSIZE`
the number of commands to remember (i.e. when you use the up arrow key).

The variables are typically set in `/etc/profile`, which configures bash globally for all users, the values can however be over-ridden by users with the `~username/.bash_profile` file, and/or by manually using the export command to set variables such as export `EDITOR=emacs`. This is one of the reasons user directories should not be world readable, as the bash_history file can contain a lot of valuable information to a hostile party. You can also set the file itself non world readable, set your .bash_profile not to log, set the file non writeable (thus denying bash the ability to write and log to it) or link it to /dev/null (this is almost always a sure sign of suspicious user activity, or a paranoid user). For the root account I would highly recommend setting the HISTFILESIZE and HISTSIZE to a low value such as 10. Unfortunately you cannot really lock down normal user’s history files, you can set them so the user cannot delete them etc, but unless you deny the user the export command, etc. they will be able to get around having all their commands logged if they are competent. Ultimately, letting users have interactive shell accounts on the server is a bad idea and should be as heavily restricted as possible.

## Shadow passwords

In all UNIX like operating systems there are several constants, and one of them is the file `/etc/passwd` and how it works. For user authentication to work properly you need (minimally) some sort of file(s) with UID to username mappings, GID to groupname mappings, passwords for the users, and other misc. info. The problem with this is that everyone needs access to the passwd file, everytime you do an ls it gets checked, so how do you store all those passwords safely, yet keep them world readable? For many years the solution has been quite simple and effective, simply hash the passwords, and store the hash, when a user needs to authenticate take the password they enter it, hash it, if it matches it was obviously the same password. The problem with this is that computing power has grown enormously, and I can now take a copy of your passwd file, and try to brute force it open in a reasonable amount of time. So to solve this several solutions exist:

- Use a 'better' hashing algorithm like MD5. Problem: can break a lot of things if they're expecting something else.
- Store the passwords elsewhere. Problem: the system/users still need access to them, and it might cause some programs to fail if they are not setup for this.

Several OS's take the first solution, Linux has implemented the second for quite a while now, it is called shadow passwords. In the passwd file your passwd is simply replaced by an 'x', which tells the system to check your passwd against the shadow file. Anyone can still read the passwd file, but only root has read access to the shadow file (the same is done for the group file and it's passwords). Seems simple enough but until recently implementing shadow passwords was a royal pain. You had to recompile all your programs that checked passwords (login, ftpd, etc, etc) and this obviously takes quite a bit of effort. This is where RedHat shines through, in it's reliance on PAM.

To implement shadow passwords you must do two things. The first is relatively simple, changing the password file, but the second can be a pain. You have to make sure all your programs have shadow password support, which can be quite painful in some cases (this is a very strong reason why more distributions should ship with PAM).

Because of RedHat's reliance on PAM for authentication, to implement a new authentication scheme all you need to do it add a PAM module that understand it, and edit the config file for whichever program (say login) allowing it to use that module to do authentication. No recompiling, and a minimal amount of fuss and muss, right? In RedHat 6.0 you are given the option during installation to choose shadow passwords, or you can implement them later via the `pwconv` and `grpconv` utilities that ship with the shadow-utils package. Most other distributions also have shadow password support, and implementation difficulty varies somewhat. Now for an attacker to look at the hashed passwords they must go to quite a bit more effort then simply copying the /etc/passwd file. Also make sure to occasionally run pwconv and in order to ensure all passwords are in fact shadowed. Sometimes passwords will get left in /etc/passwd, and not be sent to /etc/shadow as they should be by some utilities that edit the password file.

**Cracking passwords**

In Linux the passwords are stored in a hashed format, however this does not make them irretrievable, chances are you cannot reverse engineer the password from the resulting hash, however you can hash a list of words and compare them. If the results match then you have found the password, this is why good passwords are critical, and dictionary words are a terrible idea. Even with a shadow passwords file the passwords are still accessible by the root user, and if you have improperly written scripts or programs that run as root (say a www based CGI script) the password file may be retrieved by attackers. The majority of current password cracking software also allows running on multiple hosts in parallel to speed things up.

Jack the ripper

An efficient password cracker available from: http://www.false.com/security/john/.

Crack

The original widespread password cracker (as far as I know), you can get it at: http://www.users.dircon.co.uk/~crypto/.

Saltine cracker

Another password cracker with network capabilities, you can download it from: http://www.thegrid.net/gravitino/products.html.

VCU

VCU (Velocity Cracking Utilities) is a windows based programs to aid in cracking passwords, “VCU attempts to make the cracking of passwords a simple task for computer users of any experience level.”. You can download it from: http://wilter.com/wf/vcu/.

I hope this is sufficient motivation to use shadow passwords and a stronger hash like MD5 (which RedHat 6.0 supports, I don’t know of other distributions supporting it).

## PAM

"Pluggable Authentication Modules for Linux is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users." Straight from the PAM documentation, I don't think I could have said it any better. But what does this actually mean? Take a 'normal' program, say login, when a user connects to a tty (via modem or telnet) a getty program answers the call (as it were) and usually starts up the 'login' program, login then requests a username, followed by a password, which it checks against the /etc/passwd file. So what happens if you have a spiffy new digital card authentication system? Well you have to recompile login (and any other apps that will use it) so they support the new system. As you can imagine this is quite laborious and prone to errors. PAM introduces a layer of middleware (nice buzzword huh?) between the application and the actual authentication mechanism. Once a program is PAM'ified, any authentication methods PAM supports will be usable by the program. In addition to this PAM can handle account, and session data which is something 'normal' authentication mechanisms don't do well. Using PAM for example you can easily disallow login access by normal users between 6pm and 6am, thus preventing security risks. By default RedHat systems are PAM aware, I'm not sure of any other distributions that are, thus in RedHat all I have to do to say implement shadow passwords is convert the password and group files, and possibly add one or two lines to some config files (if they weren't already added). Essentially PAM gives you a great deal of flexibility when handling user authentication, and will support other features in future such as transparent chroot'ing of users (be they telneting in, or ftping). This kind of flexibility will be required if Linux is to be an enterprise class operating system. Distributions that do not ship as "pam aware" can be made so but it requires a lot of effort (you must recompile all your programs with PAM support, install PAM, etc), it is probably easier to switch straight to a PAM'ified distribution if this will be a requirement. PAM usually comes with complete documentation, and if you are looking for a good overview you should visit: http://www.sun.com/software/solaris/pam/.

## Software Management

### RPM

RPM is a software management tool originally created by RedHat, and later GNU'ed and given to the public (http://www.rpm.org/). It forms the core of administration on most systems, since one of the major tasks for any administrator is installing and keeping software up to date. Various estimates place most of the blame for security break-ins on bad passwords, and old software with known vulnerabilities. This isn't exactly surprising one would think, but with the average server probably containing 200-400 software packages, one begins to see why keeping software up to date can be a major task.

The man page for RPM is pretty bad, there is no nice way of putting it. The book "**Maximum RPM**" (ISBN: 0-672-31105-4) on the other hand is really wonderful (freely available at http://www.rpm.org/ in post script format). I would suggest this book for any RedHat administrator, and can say safely that it is required reading if you plan to build RPM packages. The basics of RPM are pretty self explanatory, packages come in an rpm format, with a simple filename convention:

package_name-package_version-rpm_build_version-architecture.rpm

nfs-server-2.2beta29-5.i386.rpm would be “nfs-server”, version “2.2beta29” of “nfs-server”, the fifth build of that rpm (i.e. it has been packaged and built 5 times, minor modifications, changes in file locations, etc.), for the Intel architecture, and it’s an rpm file.

| **Command** | **Function** |
|---|---|
| -q | Queries Packages / Database for info |
| -i | Install software |
| -U | Upgrades or Installs the software |
| -e | Extracts the software from the system (removes) |
| -v | be more Verbose |
| -h | Hash marks, a.k.a. done-o-dial |
| **Command Example** | **Function** |
| rpm -ivh package.rpm | Install 'package.rpm', be verbose, show hash marks |
| rpm -Uvh package.rpm | Upgrade 'package.rpm', be verbose, show hash marks |
| rpm -qf /some/file | Check which package owns a file |
| rpm -qpi package.rpm | Queries 'package.rpm', lists info |
| rpm -qpl package.rpm | Queries 'package.rpm', lists all files |
| rpm -qa | Queries RPM database lists all packages installed |
| rpm -e package-name | Removes 'package-name' from the system (as listed by rpm -qa) |

RedHat 5.1 ships with 528 packages, and RedHat 5.2 ships with 573, which when you think about it is a heck of a lot of software (SuSE 6.0 ships on 5 CD's, I haven’t bothered to count how many packages). Typically you will end up with 2-300 packages installed (more apps on workstations, servers tend to be leaner, but this is not always the case). So which of these should you install and which should you avoid if possible (like the r services packages). One thing I will say, the RPM's that ship with RedHat distributions are usually pretty good, and typically last 6-12 months before they are found to be broken.

There is a list of URL's and mailing lists where distribution specific errata is later on in this document.

### dpkg

The Debian package system is a similar package to RPM, however lacks some of the functionality, although overall it does an excellent job of managing software packages on a system. Combined with the dselect utility (being phased out) you can connect to remote sites, scroll through the available packages, install them, run any configuration scripts needed (like say for gpm), all from the comfort of your console. The man page for dpkg "`man dpkg`" is quite extensive.

The general format of a Debian package file (.deb) is:

packagename_packageversion-debversion.deb

ncftp2_2.4.3-2.deb

Unlike rpm files .deb files are not labeled for architecture as well (not a big deal but something to be aware of).

| **Command** | **Function** |
|---|---|
| -I | Queries Package |
| -i | Install software |
| -l | List installed software (equiv. to rpm -qa) |
| -r | Removes the software from the system |
| **Command Example** | **Function** |
| dpkg -i package.deb | Install package.deb |
| dpkg -I package.deb | Lists info about package.deb (rpm -qpi) |
| dpkg -c package.deb | Lists all files in package.deb (rpm -qpl) |
| dpkg -l | Shows all installed packages |
| rpm -r package-name | Removes 'package-name' from the system (as listed by dpkg -l) |

Debian has 1500+ packages available with the system. You will learn to love dpkg (functionally it has everything necessary, I just miss a few of the bells and whistles that rpm has, on the other hand dselect has some features I wish rpm had).

There is a list of URL's and mailing lists where distribution specific errata is later on in this document.

### tarballs / tgz

Most modern Linux distributions use a package management system to install, keep track of and remove software on the system. There are however many exceptions, Slackware does not use a true package management system per se, but instead has precompiled tarballs (a compressed tar file containing files) that you simply unpack from the root directory to install, some of which have install script to handle any post install tasks such as adding a user. These packages can also be removed, but functions such as querying, comparing installed files against packages files (trying to find tampering, etc.) is pretty much not there. Or perhaps you want to try the latest copy of X, and no-one has yet gotten around to making a nice .rpm or .deb file, so you must grab the source code (also usually in a compressed tarball), unpack it and install it. This present no more real danger then a package as most tarballs have MD5 and/or PGP signatures associated with them you can download and check. The real security concern with these is the difficulty in sometimes tracking down whether or not you have a certain piece of software installed, determining the version, and then removing or upgrading it. I would advise against using tarballs if at all possible, if you must use them it is a good idea to make a list of files on the system before you install it, and one afterwards, and then compare them using '`diff`' to find out what file it placed where. Simply run '`find /* > /filelist.txt`' before and '`find /* > /filelist2.txt`' after you install the tarball, and use '`diff -q /filelist.txt /filelist2.txt > /difflist.txt`' to get a list of what changed. Alternatively a '`tar -tf blah.tar`' will list the contents of the file, but like most tarballs you'll be running an executable install script/compiling and installing the software, so a simple file listing will not give you an actual picture of what was installed/etc. Another method for keeping track of what you have installed via tar is to use a program such as 'stow', stow installs the package to a separate directory (/opt/stow/) for example and then creates links from the system to that directory as appropriate. Stow requires that you have Perl installed and is available from: http://www.gnu.ai.mit.edu/software/stow/stow.html.

| **Command** | **Function** |
|---|---|
| -t | List files |
| -x | Extract files |
| **Command Example** | **Function** |
| tar -xf filename.tar | untars filename.tar |
| tar -xt filename.tar | lists files in filename.tar |

### Checking file integrity

Something I though I would cover semi-separately, checking the integrity of software that is retrieved from remote sites. Usually people don't worry, but recently ftp.win.tue.nl was broken into, and the tcp_wrappers package (among others) was trojaned. 59 downloads occurred before the site removed the offending packages and initiated damage control procedures. You should always check the integrity of files you download from remote sites, some day a major site will be broken into and a lot of people will suffer a lot of grief.

RPM

RPM packages can (and typically are) PGP signed by the author. This signature can be checked to ensure the package has not been tampered with or is a trojaned version. This is described in great deal in chapter 7 of "Maximum RPM" (online at http://www.rpm.org/), but consists of adding the developers keys to your public PGP keyring, and then using the –K option which will grab the appropriate key from the keyring and verify the signature. This way to trojan a package and sign it correctly they would have to steal the developers private PGP key and the password to unlock it, which should be near impossible.

dpkg

dpkg supports MD5, so you must somehow get the MD5 signatures through a trusted channel (like PGP signed email). MD5 ships with most distributions.

PGP

Many tarballs are distributed with PGP signatures in separate ASCII files, to verify them add the developers key to your keyring and then use PGP with the –o option. This way to trojan a package and sign it correctly they would have to steal the developers private PGP key and the password to unlock it, which should be near impossible. PGP for Linux is available from: ftp://ftp.replay.com/.

MD5

Another way of signing a package is to create an MD5 checksum, the reason MD5 would be used at all (since anyone could create a valid MD5 signature of a package) is that MD5 is pretty much universal and not controlled by export laws. The weakness is you must somehow distribute the MD5 signatures securely, this is usually done via email when a package is announced (vendors such as Sun do this a lot for patches).

**Automatic updates**

RPM

There are a variety of tools available for automatic installation of rpm files.

ftp://ftp.kaybee.org/pub/linux/
AutoRPM is probably the best tool for keeping rpm's up to date, simply put you point it at an ftp directory, and it downloads and installs any packages that are newer then the ones you have. Please keep in mind however if someone poisons your dns cache you will be easily compromised, so make sure you use the ftp site's IP address and not it's name. Also you should consider pointing it at an internal ftp site with packages you have tested, and have tighter control over. AutoRPM requires that you install the libnet package Net::FTP for perl.

ftp://missinglink.darkorb.net/pub/rhlupdate/
rhlupdate will also connect to an ftp site and grab any needed updates, the same caveats apply as above, and again it requires that you install the libnet package Net::FTP for perl.

http://www.iaehv.nl/users/grimaldo/info/scripts/
RpmWatch is a simple perl script that will install updates for you, note it will not suck down the packages you need so you must mirror them locally, or make them accessible locally via something like NFS or CODA .

dpkg

dpkg has a very nice automated installer called 'apt', in addition to installing software it will also retrieve and install software required to fulfill dependencies, you can download it from: http://www.debian.org/Packages/stable/admin/apt.html.

tarballs / tgz

No tools found, please tell me if you know of any (although beyond mirroring, automatically unpacking and running "`./configure ; make ; make install`", nothing really comes to mind).

**Tracking changes**

installwatch

installwatch monitor what a program does, and logs any changes it makes to the system to syslog. It’s similar to the “time” program in that it runs the program in a wrapped form so that it can monitor what happens, you run the program as “`installwatch /usr/src/something/make`” for example (optionally you can use the “`–o filename`” to log to a specific file). installwatch is available from: http://datanord.datanord.it/~pdemauro/installwatch/.

instmon

instmon is run before and after you install a tarball / tgz package (or any package for that matter). It generates a list of files changed that you can later use to undo any changes. It is available from: http://hal.csd.auth.gr/~vvas/instmon/.

## Converting Formats

Another way to deal with packages/etc. is to convert them. There are several utilities to convert rpm files to tarballs, rpm’s to deb’s, and so on.

alien

alien is probably the best utility around for converting files, it handles rpm’s, deb’s and tarballs very well. You can download it from: http://kitenet.net/programs/alien/.

## File / Filesystem security

A solid house needs a solid foundation, otherwise it will collapse. In Linux's case this is the ext2 (EXTended, version 2) filesystem. Pretty much your everyday standard UNIX-like filesystem. It supports file permissions (read, write, execute, sticky bit, suid, guid and so on), file ownership (user, group, other), and other standard things. Some of it's drawbacks are: no journaling, and especially no Access Control Lists, which are rumored to be in the upcoming ext3. On the plus side Linux has excellent software RAID, supporting Level 0, 1 and 5 very well (RAID isn't security related, but it certainly is safety/stability related).

The basic utilities to interact with files are: ls, chown, chmod, and find. Others include ln (for creating links), stat (tells you about a file) and many more. As for creating and maintaining the filesystems themselves we have fdisk (good old fdisk), mkfs (MaKe FileSystem, format, supports ext2, dos, minix, etc.), and fsck (FileSystem ChecK, scandisk that works).
So, what is it we are trying to prevent hostile people (usually users, and or network daemons fed bad info) from doing? A Linux system can be easily compromised if access to certain files is gained, for example the ability to read a non shadowed password file results in the ability to run the encrypted passwords against crack, easily finding weak password, this is a common goal of attackers coming in over the network (poorly written CGI scripts seem to be a favorite). Alternatively if an attacker can write to the password file he/she can at the least seriously disrupt the system, or (arguably worse) get whatever level of access they want. These conditions are commonly caused by "tmp races", where a setuid program (one running with root privileges) writes temporary files, typically in /tmp, however far to many do not check for the existence of a file, thus allowing an attacker to make a hard link in /tmp pointing to the password file, and when the setuid program is run, kaboom, /etc/passwd is wiped out or possibly appended to. There are many more attacks similar to this, so how can we prevent them?

Simple, set the file system up correctly when you install. The two common directories that users have write access to are /tmp and /home, splitting these off onto separate partitions also prevents users from filling up any critical filesystem (a full / is very bad indeed). A full /home could result in users not being able to log in at all (why root is in /root). Putting /tmp and /home on separate partitions is pretty much mandatory if users have shell access to the server, putting /etc, /var, and /usr on separate partitions is also a very good idea.

The primary tools for getting information about files/filesystems are all relatively simple and easy to use. df (shows disk usage) will also show inode usage (df -i, inodes contain information about files, you can run out of these before you run out of disk space, resulting in error messages of "disk full" when 'df' claims otherwise. This is similar to file allocation entries in Windows, with vfat it actually stores names in 8.3 format, using multiple entries for long filenames, with a max of 512 8.3 entries per directory, to many long filenames and the directory is 'full'. du will tell you the size of directories, very useful for finding out where all that disk space has disappeared to, usage is 'du' (lists everything in ./) or 'du dirname', optionally -s for a summary which is useful for dirs like /usr/src/linux. To gain information about specific files the primary tool is ls (similar to DOS's 'dir' command), 'ls' shows just file/dir names, 'ls -l' shows information such as file perms, size and so on, and 'ls -la' shows directories and files beginning in .'s, typical for config files/etc. ls has a few dozen options for sorting based on size, date, in reverse order and so forth, 'man ls' for all the details. For details on a particular file (creation date, last access, inode, etc) there is 'stat', it simply tells all the vital statistics on a given file(s), very useful to see if a file is in use/etc.

To manipulate files and folders we have the typical utilities like cp, mv, rm (CoPy, MoVe and ReMove), as well as tools for manipulating security information. chown is responsible for CHanging OWNership of files, the user and group a given file belongs to (the group other is always other, similar to Novell or NT's 'everyone' group). chmod (CHange MODe) changes a files attributes, the basic ones being read, write and execute, as well there is setuid, setguid (set user and group id the program is run as to the ones that own it, often times root), sticky bit and so forth. With proper use of assigning users to groups, chmod and chown you can emulate ACL's to a degree, but it is far less flexible then Sun/AIX/NT's file permissions (although this is rumored for ext3). Please be especially careful with setuid/setguid as any problems in that program/script can be magnified greatly.

I thought I would also mention find, it find's files, and can also filter based on permissions/ownership. A couple of quick examples for hunting down those setuid/guid programs:

find all setuid programs:

```
find / -perm +4000
```

find all setgid programs:

```
find / -perm +2000
```

The biggest part of file security however is user permissions. In Linux a file is 'owned' by 3 separate entities, a User, a Group, and Other (everyone else). You can set who is the user owner, and who is the group owner by:

```
chown user:group object
```

where object is a file, directory, etc. If you want to deny execute access to all of the 3 owners simply:

```
chmod x="" object
```

where x is a|g|u|o (All/User/Group/Other), force the permissions to be equal to "" (null, nothing, no access at all) and object is a file, directory, etc. This is by far the quickest and most effective way to rip out permissions and totally deny access to users/etc (="" forces it to clear it). Remember that root can **ALWAYS** change file perms and view/edit/run the file, Linux does not yet provide safety to users from root (which many would argue is a good thing). Also whoever owns the directory the object is in (be they a user/group/other with appropriate perms on the parent directory) can also potentially edit permissions (and since root owns / it can make changes that can traverse down the filesystem to any location).

### Secure file deletion

One thing many of us forget is that when you delete a file, it isn’t actually gone. Even if you overwrite it, reformat the drive, or otherwise attempt to destroy it, chances are it can be recovered, and typically data recovery services only cost a few thousand dollars, so it might well be worth an attackers time and money to have it done. The trick is to scramble the data by repeatedly flipping the magnetic bits (a.k.a. the 1’s and 0’s) so that when finished no traces of the original data remain (i.e. magnetic bits still charged the same way they originally were). Two programs (both called wipe) have been written to do just this.

wipe (durakb@crit2.univ-montp2.fr)

wipe securely deletes data by overwriting the file multiple times with various bit patterns, i.e. all 0’s, then all 1’s. then alternating 1’s and 0’s and so forth. You can use wipe on files or on devices, if used on files remember that filename’s, creation dates, permissions and so forth will not be deleted, so make sure you wipe the device if you absolutely must remove all traces of something. You can get wipe from:  http://gsu.linux.org.tr/wipe/.

wipe (thomassr@erols.com)

This one also securely deletes data by overwriting the files multiple times, this one does not however support for wiping devices. You can get it at: http://users.erols.com/thomassr/zero/download/wipe/

## TCP-IP and network security

TCP-IP was created in a time and place where security wasn't a very strong concern. Initially the 'Internet' (then called Arpanet) consisted of very few hosts, all were academic sites, big corporations or government in nature. Everyone knew everyone else, and getting on the Internet was a pretty big deal. the TCP-IP suite of protocol is remarkably robust (it hasn't failed yet), but unfortunately it has no real provisions for security, i.e. authentication, verification, encryption and so on. Spoofing packets, intercepting packets, reading data payloads, and so is remarkably easy in today's Internet. The most common attacks are denial of service attacks since they are the easiest to execute and the hardest to defeat, followed by packet sniffing, port scanning, and other related activities.

Hostnames don't always point at the right IP addresses and IP addresses don't always reverse lookup to the right hostname. Do not use hostname based authentication if possible, as DNS cache poisoning is relatively easy, relying on IP addresses for authentication reduces the problem to one of spoofing, but is also inherently flawed. There are no mechanisms in wide spread use to verify who sent data and who is receiving it (CIPE/IPv6 and VPN technology is starting to gain momentum however).

You can start by denying inbound data that claims to originate from your network(s), as this data is obviously spoofed, and to prevent your users from spoofing attacks you should block all outbound data that is not from your IP addresses. This is relatively simple and easy to manage but the vast majority of networks do not do it (I spent about a year pestering my ISP before they started). If everyone on the Internet had egress filters (that is restricted outbound traffic to that which is from their internal IP addresses) spoofing attacks would be impossible, and thus tracing attackers back to source would be far easier. You should also block the reserved networks (127.*, 10.*, etc.), I have noticed many attacks from the Internet with packets labeled as from those IP's, if you use network address translation (like IPMASQ) and do not have it properly firewalled you could be easily attacked, or used to relay an attack to a third party.

If you must communicate securely with people consider using VPN technology, the 'best' is probably IPSec, it is an open standard supported by all major vendors, and most major vendors have actual working implementations. Please see Appendix B or the Encrypting Services and Data section for more details.

### IPSec

IPSec is encryption at the packet level, it not only provides encryption of data but can provide authentication of the parties involved if you use a signed certificates (similar to signed PGP keys). Several certificate vendors (Verisign, Thawte to name two) have plans to offer IPSec certificate signing, so that you would know for a fact who you are dealing with, and also give you the ability to implement a company wide certificate architecture. IPSec is covered fully later in this document.

### IPv6

IPv6 provides no security per se, but it does have built in hooks for future security enhancements, IPSec support and so on. If used on a network it would of course make life more difficult for an attacker as Ipv6 use is not yet widespread. If you want to learn more visit: http://www.bieringer.de/linux/IPv6/. Linux currently supports Ipv6 pretty much completely (one of the few OS's that does).

### TCP-IP attack programs

A variety of programs exist to cause TCP-IP disruption (most are Denial of Service attacks) however there are a few general purpose ones that can be useful to administrators.

HUNT Project

Th HUNT Project is a set of tools for manipulating TCP-IP connections (typically on an Ethernet LAN), that is it can reset connections, spy on them and do otherwise "naughty" things. It also includes a variety of ARP based attacks and other mischievous sources of fun, You can get HUNT at: http://www.cri.cz/kra/index.html.

## PPP security

PPP provides TCP-IP (as well as IPX/SPX, and NetBEUI) connections over serial lines (which can of course be attached to modems). It is the primary method most people use to connect to the Internet (virtually all dial-up accounts are PPP). A PPP connection essentially consists of two computing devices (computer, a Palm Pilot, a terminal server, etc) connected over a serial link (usually via modems), both ends invoke PPP, authentication is handled (one of several ways), and the link is brought up. PPP has no real support for encryption, so if you require a secure link you must invest in some from of VPN software. Most systems invoke PPP in a rather kludgy way, you 'log in' to the equipment (terminal server, etc) and then as your login shell PPP is invoked, this of course means your username and password are sent in clear text over the line, and you must have an account on that piece of equipment, in this case PPP does not handle the authentication at all. A somewhat safer way of handling this is to use PAP (Password Authentication Protocol), where the authentication is handled by PPP, so you do not require a real account on the server, however the username and password is still sent in clear text, but the system at least is somewhat safer. The third (and best) method for authentication is to use CHAP (Challenge Handshake Authentication Protocol), each side exchanges a public key, and uses it to encrypt data sent for the authentication, thus your username and password are relatively safe from snooping, however actual data transfers are sent normally. One caveat with CHAP, Microsoft's implementation uses DES instead of MD5, making it slightly 'broken' if connecting with a Linux client, there are patches available however to fix this. PPP ships with almost every Linux distribution as a core part of the OS, the Linux PPP-HOWTO is available at: http://www.interweft.com.au/other/ppp-howto/ppp-howto.html.

## Basic network service security

### What is running and who is it talking to?

You can’t start securing services until you know what is running. For this task ps and netstat are invaluable, ps will tell you what is currently running (httpd, inetd, etc), netstat will tell you what the status of ports are (at this point we’re interested in ports that are open and listening, that is waiting for connections), and finally we can take a look at the various config files that control services.

### PS Output

The program ps shows us process status (information available in the /proc/ virtual filesystem). The options most commonly used are -xau, which show pretty much all the information you’d ever want to know, please note these options vary across UNIX systems, Solaris, SCO, etc all behave differently (which is incredibly annoying). The following is typical output from a machine.

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
bin        320  0.0  0.6   760   380  ?  S   Feb 12   0:00 portmap
daemon     377  0.0  0.6   784   404  ?  S   Feb 12   0:00 /usr/sbin/atd
named     2865  0.0  2.1  2120  1368  ?  S    01:14   0:01 /usr/sbin/named -u named -g named -
t /home/named
nobody     346  0.0 18.6 12728 11796  ?  S   Feb 12   3:12 squid
nobody     379  0.0  0.8  1012   544  ?  S   Feb 12   0:00 (dnsserver)
nobody     380  0.0  0.8  1012   540  ?  S   Feb 12   0:00 (dnsserver)
nobody     383  0.0  0.6   916   416  ?  S   Feb 12   0:00 (dnsserver)
nobody     385  0.0  0.8  1192   568  ?  S   Feb 12   0:00 /usr/bin/ftpget -S 1030
nobody     392  0.0  0.3   716   240  ?  S   Feb 12   0:00 (unlinkd)
nobody    1553  0.0  1.8  1932  1200  ?  S   Feb 14   0:00 httpd
nobody    1703  0.0  1.8  1932  1200  ?  S   Feb 14   0:00 httpd
root         1  0.0  0.6   776   404  ?  S   Feb 12   0:04 init [3]
root         2  0.0  0.0     0     0  ?  SW  Feb 12   0:00 (kflushd)
root         3  0.0  0.0     0     0  ?  SW  Feb 12   0:00 (kswapd)
root         4  0.0  0.0     0     0  ?  SW  Feb 12   0:00 (md_thread)
root        64  0.0  0.5   736   348  ?  S   Feb 12   0:00 kerneld
root       357  0.0  0.6   800   432  ?  S   Feb 12   0:05 syslogd
root       366  0.0  1.0  1056   684  ?  S   Feb 12   0:01 klogd
root       393  0.0  0.7   852   472  ?  S   Feb 12   0:00 crond
root       427  0.0  0.9  1272   592  ?  S   Feb 12   0:19 /usr/sbin/sshd
root       438  0.0  1.0  1184   672  ?  S   Feb 12   0:00 rpc.mountd
root       447  0.0  1.0  1180   644  ?  S   Feb 12   0:00 rpc.nfsd
root       458  0.0  1.0  1072   680  ?  S   Feb 12   0:00 /usr/sbin/dhcpd
root       489  0.0  1.7  1884  1096  ?  S   Feb 12   0:00 httpd
root       503  0.0  0.4   724   296   2 S   Feb 12   0:00 /sbin/mingetty tty2
root       505  0.0  0.3   720   228  ?  S   Feb 12   0:02 update (bdflush)
root       541  0.0  0.4   724   296   1 S   Feb 12   0:00 /sbin/mingetty tty1
root      1372  0.0  0.6   772   396  ?  S   Feb 13   0:00 inetd
root      1473  0.0  1.5  1492  1000  ?  S   Feb 13   0:00 sendmail: accepting connections on
port 25
root      2862  0.0  0.0   188    44  ?  S    01:14   0:00 /usr/sbin/holelogd.named
/home/named/dev/log
root      3090  0.0  1.9  1864  1232  ?  S    12:16   0:02 /usr/sbin/sshd
root      3103  0.0  1.1  1448   728  p1 S    12:16   0:00 su -
root      3104  0.0  1.3  1268   864  p1 S    12:16   0:00 -bash
root      3136  0.0  1.9  1836  1212  ?  S    12:21   0:04 /usr/sbin/sshd
```

The interesting ones are: portmap, named, squid (and it’s dnsserver, unlinkd and ftpget children processes), httpd, syslogd, sshd, rpc.mountd, rpc.nfsd, dhcpd, inetd, and sendmail (this server appears to be providing gateway services, email and NFS file sharing). The easiest way to learn how to read ps output is go over the ps man page and learn what the various fields are (most are self explanatory, such as %CPU, some like SIZE are a bit obscure (SIZE is the number of 4k memory ‘pages’ a program is using)). To figure out what the running programs are a safe bet is ‘man <command_name>’, which almost always gives you the manual page pertaining to that service (such as httpd). You will notice that services like telnet, ftpd, identd and several others do not show up even though they are on, this is because they are run from inetd, the ‘superserver’, to find these services look at /etc/inetd.conf, or your netstat output.

### Netstat Output

netstat tells us pretty much anything network related that you can imagine, however it is especially good at listing active connections and sockets. Using netstat we can find which ports on which interfaces are active. The following output is from a typical server using netstat –an.

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 24.108.11.200:80        205.253.183.122:3661    ESTABLISHED
tcp        0      0 0.0.0.0:1036            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 10.0.0.10:53            0.0.0.0:*               LISTEN
tcp        0      0 28.208.55.254:53        0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:635             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
udp        0      0 127.0.0.1:1031          0.0.0.0:*
udp        0      0 0.0.0.0:1029            0.0.0.0:*
udp        0      0 0.0.0.0:800             0.0.0.0:*
udp        0      0 0.0.0.0:1028            0.0.0.0:*
udp        0      0 10.0.0.10:53            0.0.0.0:*
udp        0      0 28.208.55.254:53        0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*
udp        0      0 10.1.0.1:138            0.0.0.0:*
udp        0      0 10.1.0.1:137            0.0.0.0:*
udp        0      0 10.0.0.10:138           0.0.0.0:*
udp        0      0 10.0.0.10:137           0.0.0.0:*
udp        0      0 0.0.0.0:138             0.0.0.0:*
udp        0      0 0.0.0.0:137             0.0.0.0:*
udp        0      0 0.0.0.0:2049            0.0.0.0:*
udp        0      0 0.0.0.0:635             0.0.0.0:*
udp        0      0 0.0.0.0:514             0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
raw        0      0 0.0.0.0:1               0.0.0.0:*
raw        0      0 0.0.0.0:6               0.0.0.0:*
```

Numeric output is in my opinion easier to read, once you memorize /etc/services anyways. The interesting fields for us are the first field, type of service, the fourth field which is the IP address of the interface and the port, the foreign address (if not 0.0.0.0.* means someone is actively talking to it), and the port state. The first line is a remote client talking to the web server on this machine (port 80). We then see the www server listening on 0.0.0.0:80 which means all interfaces, port 80, followed by the DNS server running on all 3 interfaces, a samba server (139), a mail server (25), an NFS server (2049) and so on. You will notice the ftp server (21) listed, even though it is run out of inetd, and not currently in use (i.e. no one is actively ftping in), it is listed in the netstat output. This makes netstat especially useful for finding out what is active on a machine, making an inventory of active and inactive network related software on the server much easier.

### lsof

lsof is a handy program similar in idea to ps, except that it prints out what files/etc are open, which can include network sockets. Unfortunately your average lsof puts out a lot of information, so you will need to use grep or redirect it through less to make sense of it.

```
squid      9726     root    4u  inet      78774               TCP localhost:2074-
>localhost:2073 (ESTABLISHED)
squid      9726     root    5u  inet      78777               TCP localhost:2076-
>localhost:2075 (ESTABLISHED)
squid      9726     root    6u  inet      78780               TCP localhost:2078-
>localhost:2077 (ESTABLISHED)
squid      9726     root    7w   CHR        1,3              6205 /dev/null
squid      9726     root   14u  inet      78789               TCP host1:3128 (LISTEN)
squid      9726     root   15u  inet      78790               UDP host1:3130
squid      9726     root   16u  inet      78791               UDP host1:3130
squid      9726     root   12u  inet     167524               TCP host1:3128->host2:3630
(ESTABLISHED)
squid      9726     root   17u  inet     167528               TCP host1:3424-
>www.playboy.com:http (SYN_SENT)
```

Shows for example that we have squid running, listening on ports 3128 and 3130, the last two lines for example shows an open connection from an internal host to squid, and the resulting action squid takes to fulfill it (going to www.playboy.com). host1 is the squid server and host2 is the client machine making the request. This is an invaluable tool for getting a precise image of what is going on network wise with your server. You can get lsof with some distributions, please not versions of lsof compiled for kernel version 2.0.x will not work with kernel 2.2.x and vice versa, there were to many changes. The head end site for lsof is at: ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/.

## Basic network services config files

There are several important config files that control what services Linux runs and how they run. Unfortunately many of them are in differing locations depending on what/how you installed Linux and the services. Typical locations are:

Inetd server config file:

```
/etc/inetd.conf
```

Initial start up files in various forms

```
/etc/rc.d/*
/etc/*
```

The best thing to do is figure out which services you want to run, and disable/remove the rest. Please see the appropriate section in package management for your system (RPM, dpkg, tarballs).

### inetd.conf

inetd.conf is responsible for starting services, typically ones that do not need to run continuously, or are session based (such as telnet, or ftpd). This is because the overhead of running a service constantly (like telnet) would be higher then the occasional start up cost (or firing in.telnetd up) when a user wants to use it. For some services (like DNS) that service many quick connections the overhead of starting the service every few seconds would be higher then constantly running it (also with services such as DNS and email time is critical, a few seconds delay starting an ftp session won't hurt much). The man page for inetd.conf covers the basics:

```
man inetd.conf
```

the service itself is called inetd and is run at boot time, so you can easily stop/start/reload it by manipulating the inetd process. Whenever you make changes to inetd.conf you must restart inetd to make the changes effective, killall -1 inetd will restart it properly. Lines in inetd.conf can be commented out with a # as usual (this is a very simple and effective way of disabling services like rexec). It is advisable to disable as many services in inetd.conf as possible, typically the only ones in use will be ftp, pop and imap, telnet and r services should be replaced with ssh, and services like systat/netstat and finger give away far to much information. Access to programs started by inetd can be easily controlled by the use of **TCP_WRAPPERS**.

### TCP_WRAPPERS

Using tcp_wrappers makes securing your servers against outside intrusion is a lot simpler and painless then you would expect. TCP_WRAPPERS is controlled from two files:

```
/etc/hosts.allow
/etc/hosts.deny
```

hosts.allow is checked first, and the rules are checked from first to last, when it finds a rule that explicitly allows you in (i.e. a rule allowing your host, domain, subnet mask, etc) it lets you connect to the service, if it fails to find any rules that pertain to you in hosts.allow, it then

goes to check hosts.deny for a rule denying you entry. Again it checks the rules in hosts.deny from first to last, and the first rule it finds that denies you access (i.e. a rule disallowing your host, domain, subnet mask, etc) means it doesn't let you in. If it fails to find a rule denying you entry it then by default lets you. If you are paranoid like me the last rule (or only rule if you are going to a default policy of non-optimistic security) should be:
in hosts.deny:

```
ALL: 0.0.0.0/0.0.0.0
```

which means all services, all locations, i.e. a default deny policy. You might also want to just default deny access to say telnet, and leave ftp wide open to the world, to do this you would have:

in hosts.allow:

```
in.telnetd: 10.0.0.0/255.255.255.0      # allow access from my internal 
network of 10.*.*.*
in.ftpd: 0.0.0.0/0.0.0.0                # allow access from anywhere in the 
world
```

in hosts.deny:

```
in.telnetd: 0.0.0.0/0.0.0.0             # deny access to telnetd from 
anywhere
```

or if you wish to be really safe:

```
ALL: 0.0.0.0/0.0.0.0           # deny access to everything from everywhere
```

This may affect services such as ssh and nfs, so be careful!
You may wish to simply list all the services you are using separately:

```
in.telnetd: 0.0.0.0/0.0.0.0
ipop3d: 0.0.0.0/0.0.0.0
```

If you leave a service on that you shouldn't have in inetd.conf, and DO NOT have a default deny policy, you could be up the creek. It is safer (and a bit more work, but in the long run less work then rebuilding the server) to have default deny rules for firewalling and tcp_wrappers, thus is you leave something on by accident, by default there will be no access to it. If you install something that users need access and you forget to put allow rules in, they will quickly complain that they can't get access and you will be able to rectify the problem quickly. Better safe then sorry. The man pages for tcp_wrappers are very good and available by:

```
man hosts.allow
```

and/or (they are the same man page):

```
man hosts.deny
```

One minor caveat with tcp_wrappers that recently popped up on Bugtraq, tcp_wrappers interprets lines in hosts.allow and hosts.deny in the following manner:

strip off all \'s (line continuations), making all the lines complete (also note the max length of a line is about 2k, better to use multiple lines in some cases).

strip out lines starting with #'s, i.e. all commented out lines. Thus:

```
# this is a test
# in.ftpd: 1.1.1.1 \
in.telnetd: 1.1.1.1
```

means the "`in.telnetd: 1.1.1.1`" line would be ignored to.

**/etc/services**

The services file is a list of port numbers, the protocol and the corresponding name. The format is:

```
service-name            port/protocol           aliases          #
optional comment
```

for example:

```
time            37/udp          timserver
rlp             39/udp          resource        # resource location
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/tcp
domain          53/udp
```

This file is used for example when you run '`netstat -a`', and of course not used when you run '`netstat -an`'

## Network services

### Telnetd

Telnet was one of the first services on what is now the Internet, it allows you to login to a remote machine interactively, issue commands and see their results. It is still the primary default tools for remote administration in most environments, and has nearly universal support (even NT has a telnet daemon and client). It is also one of the most insecure protocols, susceptible to sniffing, hijacking, etc. If you have clients using telnet to come into the server you should definitely chroot their accounts if possible, as well as restricting telnet to the hosts they use with tcp_wrappers. The best solution for securing telnet is to disable it and use SSL'ified telnet or ssh.

Problems with telnet include:

- Clear text authentication, username and password.
- Clear text of all commands.
- Password guessing attacks (minimal, will end up in the log files)

The best solution is to turn telnet off and use ssh. This is however not practical in all situations. If you must use telnet then I strongly suggest firewalling it, have rules to allow hosts/networks access to port 23, and then a general rule denying access to port 23, as well as using tcp_wrappers (which is more efficient because the system only checks each telnet connection and not every packet against the firewall rules) however using tcp_wrappers will allow people to establish the fact that you are running telnet, it allows them to connect, evaluates the connection, and then closes it if they are not listed as being allowed in.
An example of firewalling rules:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 23
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 23
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 23
```

or in ipchains:

```
ipchains -A input -p all -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 23
ipchains -A input -p all -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 23
ipchains -A input -p all -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 23
```

An example of the same using tcp_wrappers:
In /etc/hosts.allow

```
in.telnetd: 10.0.0.0/255.0.0.0, some.trusted.host
```

And in /etc/hosts.deny

```
in.telnetd: ALL
```

There are several encrypted alternatives to telnet as mentioned before, ssh, SSLeay Telnet, and other third party utils, I personally feel that the 'best' alternative if you are going to go to the bother of ripping telnet out and replacing it with something better is to use ssh.

To secure user accounts with respect to telnet there are several things you can do. Number one would be not letting root login via telnet, this is controlled by /etc/securetty and by default in most distributions root is restricted to logging on from the console (a good thing). For a user to successfully login their shell has to be valid (this is determined by the list of shells in /etc/shells), so setting up user accounts that are allowed to login is simply a matter of setting their shell to something listed in /etc/shells, and keeping users out as simple as setting their shell to /bin/false (or something else not listed in /etc/shells. Now for some practical examples of what you can accomplish by setting the user shell to things other then shells.

For an ISP that wishes to allow customers to change their password easily, but not allow them access to the system (my ISP uses Ultrasparcs and refuses to give out user accounts for some reason, I wonder why).

in /etc/shells list:

```
/usr/bin/passwd
```

and set the users shell to /usr/bin/passwd so you end
up with something like:

```
username:x:1000:1000::/home/username:/usr/bin/passwd
```

and voila, the user telnets to the server, is prompted for their username and password, and is then prompted to change his password, if he does so successfully passwd then exits and he is disconnected, if he is unsuccessful passwd exist and he gets disconnected. The following is a transcript of such a setup when a user telnets in:

```
Trying 1.2.3.4…
Connected to localhost.
Escape character is '^]'.

Red Hat Linux release 5.2 (Apollo)
Kernel 2.2.5 on an i586
login: tester
Password:
Changing password for tester
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
Connection closed by foreign host.
```

Telnet also displays a banner by default when someone connects. This banner typically contains systems information like the name, OS, release and sometimes other detailed information such as the kernel version. Historically this was useful if you had to work on multiple OS's, however in today's hostile Internet it is generally more harmful then useful. Telnetd displays the contents of the file `/etc/issue.net` (typically it is identical to `/etc/issue` which is displayed on terminals and so forth), this file is usually recreated at boot time in most Linux distributions, from the rc.local startup file. Simply edit the rc.local file, either modifying what it puts into `/etc/issue` and `/etc/issue.net`, or comment out the lines that create those files, then edit the files with some static information.

Typical Linux `rc.local` contents pertaining to `/etc/issue` and `/etc/issue.net`:

```
# This will overwrite /etc/issue at every boot.  So, make any changes you
# want to make to /etc/issue here or you will lose them when you reboot.
echo "" > /etc/issue
echo "$R" >> /etc/issue
echo "Kernel $(uname -r) on $a $(uname -m)" >> /etc/issue

cp -f /etc/issue /etc/issue.net
echo >> /etc/issue
```

simply comment out the lines or remove the uname commands. If you absolutely must have telnet enabled for user logins make sure you have a disclaimer printed:

```
This system is for authorized use only. Trespassers will be prosecuted.
```

something like the above. Legally you are in a stronger position if someone cracks into the system or otherwise abuses your telnet daemon.

## SSHD

SSH is a secure protocol and set of tools to replace some common (insecure) ones. It was designed from the beginning to offer a maximum of security, and is designed for remote access of servers in a secure manner. SSH can be used to secure any network based traffic, by setting it up as a 'pipe', i.e. binding it to a certain port at both ends, this is quite kludgy but good for such things as using X across the Internet, in addition to this the server components runs on most UNIX systems, and NT, and the client components runs on pretty much anything. Unfortunately SSH is no longer free, however there is a project to create a free implementation of the SSH protocol.

There aren't any problems with SSH per se like there are with telnet, all session traffic is encrypted except of course and key exchange is done securely (alternatively you can preload keys at either end to prevent them from being transmitted), SSH typically runs as a daemon, and can easily be locked down by using the sshd_config file. You can also run sshd out of inetd, and thus use tcp_wrappers, and  by default the ssh rpm's from ftp.replay.com have tcp_wrappers check option compiled into them. Thus using "sshd: blahblah" in hosts.allow and hosts.deny allows you to easily restrict access to ssh. Please note earlier versions of ssh do contain bugs, and sites have been hacked (typically with man in the middle attacks or problems with buffer overflows in the ssh code), but later version of ssh address these problems.

The firewalling rules for ssh are pretty much identical to telnet, and there is of course tcp_wrappers, the problem with tcp_wrappers being that an attacker connects to the port, but doesn't get a daemon, HOWEVER they know that there is something on that port, whereas with firewalling they don't even get a connection to the port. The following is an example of allowing people to ssh from internal machines, and a certain C class on the internet (say the C class your ISP uses for it's dial-up pool of modems).

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 22
ipfwadm -I -a accept -P tcp -S isp.dial.up.pool/24 -D 0.0.0.0/0 22
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 22
```

or

```
ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 22
ipchains -A input -p tcp -j ACCEPT -s isp.dial.up.pool/24 -d 0.0.0.0/0 22
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 22
```

Or via tcp_wrappers:
hosts.allow:

```
sshd: 10.0.0.0/255.0.0.0, isp.dial.up.pool/255.255.255.0
```

hosts.deny:

```
sshd: 0.0.0.0/0.0.0.0
```

In addition to this ssh has a wonderful configuration file, in /etc/sshd/sshd_config by default in the RPM's available on ftp.replay.com. You can easily restrict who is allowed to login, which hosts, and what type of authentication they are allowed to use. The default configuration file is relatively safe but following is a more secure one with explanations. Please note all this info can be obtained by a 'man sshd' which is one of the few well written man pages out there.

```
Port 22
# runs on port 22, the standard
ListenAddress 0.0.0.0
# listens to all interfaces, you might only want to bind a firewall
# internally, etc
HostKey /etc/ssh/ssh_host_key
# where the host key is
RandomSeed /etc/ssh/ssh_random_seed
# where the random seed is
ServerKeyBits 768
# how long the server key is
LoginGraceTime 300
# how long they get to punch their credentials in
KeyRegenerationInterval 3600
# how often the server key gets regenerated
PermitRootLogin no
# permit root to login? hell no
IgnoreRhosts yes
# ignore .rhosts files in users dir? hell yes
StrictModes yes
# ensures users don't do silly things
QuietMode no
# if yes it doesn't log anything. yikes. we wanna log logins/etc.
X11Forwarding no
# forward X11? shouldn't have to on a server
FascistLogging no
# maybe we don't wanna log toto much.
PrintMotd yes
# print the message of the day? always nice
KeepAlive yes
# ensures sessions will be properly disconnected
SyslogFacility DAEMON
# who's doing the logging?
RhostsAuthentication no
# allow rhosts to be used for authentication? the default is no
# but nice to say it anyways
RhostsRSAAuthentication no
# is authentication  using rhosts or /etc/hosts.equiv sufficient
# not in my mind. the default is yes so lets turn it off.
RSAAuthentication yes
# allow pure RSA authentication? this one is pretty safe
PasswordAuthentication yes
# allow users to use their normal login/passwd? why not.
PermitEmptyPasswords no
# permit accounts with empty password to log in? hell no
```

```
Other useful sshd_config directives include:
AllowGroups - explicitly allow groups (/etc/group) to login using ssh
DenyGroups - explicitly disallows groups (/etc/groups) from logging in
AllowUsers - explicitly allow users to login in using ssh
DenyUsers - explicitly blocks users from logging in
AllowHosts - allow certain hosts, the rest will be denied
DenyHosts - blocks certain hosts, the rest will be allowed
IdleTimeout time - time in minutes/hours/days/etc, forces a logout
by SIGHUP'ing the process.
```

## Fresh Free FiSSH

Most of us still have to sit in front of windows workstations, and ssh clients for windows are a pain to find. Fresh Free FiSSH is a free ssh client for Windows 95/NT 4.0, although not yet completed. I would recommend keeping your eye on it though if you are like me and have many Windows workstations, the URL is: http://www.massconfusion.com/ssh/.

### Tera Term

Tera Term is a free Telnet client for Windows, and has an add on DLL to enable ssh support. Tera Term is available from: http://hp.vector.co.jp/authors/VA002416/teraterm.html. The add on DLL for SSH support is available from: http://www.zip.com.au/~roca/ttssh.html.

### putty

putty is a Windows SSH client, pretty good, and completely free, and also small (184k currently). You can download it from: ftp://rak.isternet.sk/mnt/rhcd/misc/putty/.

### mindterm

mindterm is a free java ssh client, you can get it at: http://www.mindbright.se/mindterm/.

### LSH

LSH is a free implementation of the SSH protocol (both client and server), LSH is GNU licensed and is starting to look like the alternative (commercially speaking) to SSH (which is not free anymore). You can download it from: http://www.net.lut.ac.uk/psst/, please note it is under development.

### RSH, REXEC, RCP

R services such as rsh, rcp, rexec and so forth are **very insecure**. There is simply no other way to state it. Their basis of security is based on the hostname/IP address of the machine connecting, which can easily be spoofed, or using techniques such as DNS poisoning otherwise compromised. By default they are not all disabled, please do so immediately. Edit /etc/inetd.conf and look for rexec, rsh and so on, and comment them out, followed by a `"killall -1 inetd`" to restart inetd.

If you absolutely must run these services use tcp_wrappers to restrict access, it's not much but it will help. Also make sure you firewall them as tcp_wrappers will allow an attacker to see that they are running, which might result in a spoofed attack, something tcp_wrappers cannot defend against if done properly. Access to the various R services is controlled via rhosts files, usually each user has their own rhosts file, unfortunately this is susceptible to packet spoofing. The problem with r services is also that once there is a minor security breach that can be used to modify files, editing a users (like root's) rhost file makes it very easy to crack a system wide open.

If you need remote administration tools that are easy to use and similar to rsh/etc I would recommend nsh (Network SHell) or SSH, they both support encryption, and a much higher level of security. Alternatively using VPN software will reduce some of the risk as you can deny packet spoofers the chance to compromise your system(s) (part of IPSec is authentication of sender and source, which is almost more important then encrypting the data in some cases).

## Webmin

Webmin is one of the better remote administration tools for Linux, written primarily in Perl it is easy to use and easy to setup. You can assign different 'users' (usernames and passwords are held internally by webmin) varying levels of access, for example you could assign bob access to shutdown the server only, and give john access to create/delete and manipulate users only. In addition to this it works on most Linux platforms and a variety of other UNIX platforms. The main 'problem' with webmin is somewhat poor documentation in some areas of usage, and the fact that the username/password pair are sent in clear text over the network (this is minimized slightly by the ability to grant access to only certain hosts(s) and networks. Most importantly it makes the system more accessible to non technical people who must administer systems in such a way that you do not have to grant them actual accounts on the server. Webmin is available at: http://www.webmin.com/webmin/, and is currently free. Webmin defaults to running on port 10000 and should be firewalled:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 10000
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 10000
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 10000
```

or in ipchains:

```
ipchains -A input -p all -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 10000
ipchains -A input -p all -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 10000
ipchains -A input -p all -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 10000
```

**FTP**

WuFTPD

FTP used to be the most used protocol on the Internet by sheer data traffic until it was surpassed by HTTP a few years ago (yes, there was a WWW free Internet once upon a time). FTP does one thing, and it does it well, transferring of files between systems (well that's not entirely true, for mirroring of date rsync beats the socks off of ftp). The protocol itself is insecure, passwords, data, etc is transferred in cleartext and can easily be sniffed, however most ftp usage is 'anonymous', so this isn't a huge problem. One of the main problems typically encountered with ftp sites is improper permissions on directories that allow people to use the site to distribute their own data (typically copyrighted material, etc). Again as with telnet you should use an account for ftping that is not used for administrative work since the password will be flying around the network in clear text.
Problems with ftp in general include:

- Clear text authentication, username and password.
- Clear text of all commands.
- Password guessing attacks (minimal, will end up in the log files)
- Improper server setup and consequent abuse of gained privileges
- Several nasty Denial of Service attacks still exist in WU-FTPD

Securing FTP isn't to bad, between firewalling and tcp_wrappers you can restrict access based on IP address / hostname quite well. In addition ftp runs chrooted by default for anyone logging in as anonymous, or an account defined as guest. With some work amount of work you can set all users that are ftping in to be chrooted to say their home directory. You can also run an ftp daemon that encrypts the data (such as the SSL add on package/etc) however this means your ftp clients must speak the encryption protocol and this isn't always to practical. Also make very sure you have no publicly accessible directories on your ftp server that are both readable and writeable, otherwise people will exploit it to distribute their own stuff (typically warez or porn).
An example of firewalling rules:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 21
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 21
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 21

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 21
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 21
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 21
```

An example of the same using tcp_wrappers:
In /etc/hosts.allow

```
in.ftpd: 10.0.0.0/255.0.0.0, some.trusted.host
```

And in /etc/hosts.deny

```
in.ftpd: ALL
```

There are several encrypted alternatives to ftp as mentioned before, SSLeay FTPD, and other third party utils. Since most ftp accounts are not used as admin accounts (BAD IDEA!), and hopefully run chrooted, the security risk is minimized. Now that we have hopefully covered all the network based parts of ftp, lets go over securing the user accounts and environment. The main mechanism for user security in ftpd is the use of chroot. For example by default all people logging in as anonymous have /home/ftp/ set as their 'root'. They cannot get out of this and say look at the contents of /home/ or /etc/. The same can be applied to groups of users and/or individuals, for example you could set all users to be chroot'ed to /home/ when they ftp in, or in extreme cases of user privacy (say on a www server hosting multiple domains) set each user chroot'ed to within their own home directory. This is accomplished through the use of /etc/ftpaccess and /etc/passwd (man ftpaccess has all the info). I will give a few examples of what needs to be done to accomplish this since it can be quite confusing at first. As well ftpd checks /etc/ftpusers and if the user attempting to login is listed in that file (like root should be) it will not let the user login via ftp.

To chroot users as they login into the ftp server is rather simple, but poorly documented. The ftp server check ftpaccess for 'guestgroup's, which are simply "guestgroup some-group-on-the-system" i.e. "guestgroup badusers", and the groupname needs to be defined in /etc/group and have members added. As well you need to edit their passwd file line so that the ftp server knows where to dump them. And since they are now chrooted into that directory on the system, they do not have access to /lib, etc so you must copy certain files into their dir for things like ls to work properly (always a nice touch).

Setting up a user (billybob) so that he can ftp in, and ends up chroot'ed in his home directory (because he keeps threatening to take the sysadmin possum hunting). In addition to this billybob can telnet in and change his password, but nothing else because he keeps trying to run ircbots on the system. The system he is on uses shadowed passwords, so that's why there is an 'x' in billybob's password field.

First off billybob needs a properly setup user account:

```
billybob:x:500:500:Billy Bob:/home/billybob/./:/usr/bin/passwd
```

this means that the ftp server will chroot billybob into /home/billybob/ and chdir him into what is now / (/home/billybob to the rest of us). The ftpaccess man file covers this bit ok, and of course /usr/sbin/passwd needs to be listed in /etc/shells

Secondly for the ftp server to know that he is being chrooted he needs to be a member of a group (badusers, ftppeople, etc) that is defined in /etc/group. And then that group must be listed in /etc/ftpaccess.

Now you need to copy some libs/binaries otherwise billybob won't be able to do a whole lot once he has ftp'ed in. The files needed are available as packages (usually called “anonftp”), once this is installed the files will be copied to /home/ftp/, you will notice there is an etc/passwd, this is simply uses to map UID's to usernames, if you want billybob to see his username and not UID, add a line for him (i.e. copy his line from the real /etc/passwd to this one). The same applies to the group file.

without "billybob:*:500:500:::" in /home/billybob/etc/passwd:

```
drwxr-xr-x   2 500      500          1024 Jul 14 20:46 billybob
```

and with the line added:

```
drwxr-xr-x   2 billybob 500          1024 Jul 14 20:46 billybob
```

and with a line for billybob's group added to the group file:

```
drwxr-xr-x   2 billybob billybob     1024 Jul 14 20:46 billybob
```

Billybob can now ftp into the system, upload and download files from /home/billybob to his hearts content, change his password all on his own, and do no damage to the system, nor download the passwords file or other nasty things.

FTP is also a rather special protocol in that the clients connect to port 21 (typically) on the ftp server, and then port 20 of the ftp server connects to the client and that is the connection that the actual data is sent over. This means that port 20 has to make outgoing connections, keep this in mind when setting up a firewall either to protect ftp servers or clients using ftp. As well there is 'passive' ftp and usually used by www browsers/etc, and involves incoming connections to the ftp server on high port numbers (instead of using 20 they agree on something else). If you intend to have a public ftp server put up a machine that JUST does the ftp serving, and nothing else, preferably outside of your internal LAN (see Practical Unix and Internet Security for discussions of this 'DMZ' concept).

## Apache

What can I say about securing Apache? Not much actually. By default Apache runs as the user 'nobody', giving it very little access to the system, and by and large the Apache team has done an excellent job of avoiding buffer overflows/etc. In general most www servers simply retrieve data off of the system and send it out, most of the danger come not from Apache but from sloppy programs that are executed via Apache (CGI's, server side includes, etc).
If going with Apache I would recommend using the 1.3 series unless you have some strange reason for sticking to 1.2, the active development is now on 1.3, and includes many new features from security, usability, stability and performance viewpoints. Most servers based upon Apache (RedHat Secure Server, Stronghold, etc.) are generally just as bug free but there are occasionally problems.

If you want to be paranoid I would suggest running Apache in a chrooted environment, this however is sometimes more trouble then it is worth. Doing this will break a great many things. You must also install numerous libraries, perl, and any other utilities that your apache server will be using, as well as any configuration files you wish to have access to. Any CGI scripts and other things that interact with the system will be somewhat problematic and generally harder to troubleshoot.

The simplest way to setup apache chrooted is to simply install it and move/edit the necessary files. A good idea is to create a directory (such as `/chroot/apache/`), preferably on a separate filesystem from /, /usr, etc (symlinks, accidental filling of partitions, etc...), and then create a file structure under it for apache. The following is an example, simply replace /chroot-http/ with your choice. You must of course execute these steps as root for it to work. RPM supports this with the --root dir directive, simply install apache and the needed libs using rpm (thus gaining it's support for dependencies/etc, making your life easier).

Apache logs requests and so forth internally, so you don't have to worry about setting up holelogd or any other strangeness in order to get your log files behaving.

About the simplest way to secure apache and insure that it doesn't have unnecessary access to your filesystem is to create a /www/ or similar directory and place ALL the websites, webcontent, cgi's and so forth under it. Then you can simply set access.conf up to deny any access to /, and enable access to /www/ and it's various cgi-bin directories.

Example for access.conf:

```
<Directory />
Options None
AllowOverride None
</Directory>

<Directory /www >
Options Indexes FollowSymLinks Includes
AllowOverride None
</Directory>
```

Access to directories can also be controlled easily, Apache supports the defining and placement of files (usually referred to as htaccess files) that can control access based on username and password, IP of origin, and so forth. This is defined in srm.conf:

```
AccessFileName .htaccess
```

The format of this file is covered in the apache documentation, and is identical to directives you would place in access.conf (well almost). User authentication via username and password is also covered in depth at: http://www.apacheweek.com/features/userauth/.

You will also want to prevent people from viewing the .htaccess file(s), place this in your srm.conf:

```
<Files .htaccess>
order allow,deny
deny from all
</Files>
```

This will disallow the viewing of any file called '.htaccess'.

Apache ruins on port 80, tcp usually, and if it is for internal use only (an Intranet, or www based control mechanism for a firewall server say) you should definitely firewall it.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 80
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 80
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 80
```

or in ipchains:

```
ipchains -A input -p all -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 80
ipchains -A input -p all -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 80
ipchains -A input -p all -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 80
```

**SQUID**

SQUID is a powerful and fast Object Cache. It proxies FTP and WWW sessions, basically giving it many of the properties of an FTP and a WWW server, but it only reads and writes files within it's cache directory (or so we hope), making it relatively safe. Squid would be very hard to use to actually compromise the system it is on, in addition to it running as a non root user (typically 'nobody'), so generally it's not much to worry about. Your main worry with squid should be improper configuration, for example if squid is hooked up to your internal network (as is usually the case), and the internet (again, very common), it could actually be used to reach internal hosts (even if they are using non routed IP addresses). Hence proper configuration of squid is very important.

The simplest way to make sure this doesn't happen is to use squid's internal configuration and only bind it to the internal interface(s), not letting the outside world attempt to use it as a proxy to get at your internal LAN, in addition to this firewalling it is a good idea. Squid can also be used as an HTTP accelerator, perhaps you have an NT WWW Server on the internal network that you want to share with the world, in this case things get a bit harder to configure but it is possible to do relatively securely. Fortunately squid has very good ACL's (Access Control Lists) built into the squid.conf file, allowing you to lock down access by names, ip's, networks, time of day, actual day (perhaps you allow unlimited browsing on the weekends for people that actually come in to the office). Remember however that the more complicated an ACL is, the slower squid will be to respond to requests.

The ACL's work by defining rules, and then applying those rules, for example:

```
acl internalnet 10.0.0.0/255.0.0.0
http_access allow internalnet
http_access deny all
```

Which defines "internalnet" as being anything with a source of 10.0.0.0/255.255.255.0, allowing it access to the http caching port, and denying everything else. Remember that rules are read in the order given, just like ipfwadm, allowing you to get very complex (and make mistakes if you are not careful). Always start with the specific rules followed by more general rules, and remember to put blanket denials after specific allowals, otherwise it might make it through. It's better to accidentally deny something then to let it though, as you'll find out about denials (usually from annoyed users) faster then things that get through (when annoyed users notice accounting files from the internal www server appearing on the Internet). The squid configuration files (squid.conf) is well commented (to the point of overkill) and also has a decent man page.

Another useful example is blocking ads, so to block them you can add the following to squid.conf:

```
acl ads dstdomain ads.blah.com
http_access deny ads
```

The acl declaration is simply a pattern, be it a destination domain name, source domain name, regex and so on, the http_access directive actually specifies what to do with it (deny, allow, etc). Properly setup this is an extremely powerful tool to restrict access to the WWW, unfortunately it does have one Achilles heel, it doesn't support user based authentication and control (not that many product do in any case). Remember that like any set of rules they are read from top to bottom, so put your specific denials and allowals first, and then the more general rules. The squid.conf file should be well commented and self explanatory, the squid FAQ is at: http://squid.nlanr.net/Squid/FAQ/FAQ.html.

One important security issue most people overlook with squid is the log files it keeps. By default squid may or may not log each request it handles (depends on the config file), from http://www.nsa.gov/ to "http://members.porn.com/cgi-bin/access&member=johndoe&password=booboo". You definitely want to disable the access logs unless you want to keep a close eye on what people view on the Internet (legally this is questionable, check with your lawyers), the directive is "cache_access_log" and to disable it set it to "/dev/null", this logs ALL accesses, and ICP queries (inter-cache communications). The next big one is the "cache_store_log", which is actually semi useful for generating statistics on how effective your www cache is, it doesn't log who made the request, simply what the status of objects in the cache is, so in this case you would see the pictures on playboys site being repeatedly served, to disable it set it to "none". The "cache_log" should probably be left on, it contains basic debugging info such as when the server was started and when it was stopped, to disable it set it to "/dev/null". Another, not very well documented log files is the "cache_swap_log" file, which keeps a record of what is going on with the cache, and will also show you the URL's people are visiting (but not who/etc), setting this to "/dev/null" doesn't work (in fact squid pukes out severely) and setting it to "none" simply changes the filename from "log" to "none". The only way to stop it is to link the file to "/dev/null" (by default the root of the www cache files /log), and also to link the "log-last-clean" to "/dev/null" (although in my quick tests it doesn't appear to store anything you can't be sure otherwise). So to summarize:

in squid.conf:

```
cache_access_log /dev/null
cache_store_log none
cache_log /dev/null
```

and link:
/var/spool/squid/log to /dev/null
/var/spool/squid/log-last-clean to /dev/null

or whichever directory holds the root of your www cache (the 00 through 0F directories).

Another important issue that gets forgotten is the ICP (Internet Cache Protocol) component of squid. The only time you will use ICP is if you create arrays or chains of proxy servers. If like me you have only the one proxy server you should definitely disabled ICP. This is easily done by setting the ICP port in squid.conf from the default "3130" to "0". You should also firewall port 3128 (the default squid port that clients bind to) from the Internet:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 3128
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 3128
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 3128
```

or in ipchains:

```
ipchains -A input -p all -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 3128
ipchains -A input -p all -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 3128
ipchains -A input -p all -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 3128
```

**SMTP**

Sendmail

Sendmail is another one of those services most of us have a love/hate relationship with. We hate to admin it and we'd love to replace it. This is usually not possible, as sendmail has many features and stability, with the downside being it runs as. Newer mailer packages such as qmail, address these problems however they also suffer problems, qmail lacks certain features (anti spam/UBE) that sendmail has, as well as a no binary distribution clause (actually you can make binary distributions but the restrictions are very heavy and you must gain the authors permission, which he will probably not grant if you do not follow his restrictions to the letter).

Sendmail has earned itself a very bad reputation for security, however I find it hard to blame software when I find systems running versions of sendmail like 8.6.x, we are now up to 8.9.1. The root of the problem is that almost everyone runs sendmail (estimates say %70 of all Internet email passes through a sendmail box on it's journey), so as soon as a bug is found, finding a system to exploit isn't all that hard. The last few releases of sendmail have been quite good, no root hacks, etc, and with the new anti spam features sendmail is finally to come of age. More information on Sendmail and source code is available from: http://www.sendmail.org/.

Chroot'ing sendmail is a good option, but a lot of work, and since it runs as root, rather debatable as to the effectiveness of this (since root can break out of a chrooted jail). I find that by keeping sendmail up to date, you can pretty much account yourself safe, of course if a new exploit comes out and the patch takes 24 hours we're up poop creek, the same applies to any piece of software though. Also by using Sendmail's advanced anti spam features you can effectively block 99% of all spam coming to you, and prevent other from using your system to relay spam.

Keeping sendmail up to date is relatively simple, I would recommend minimally version 8.9.3 (the 8.9 series has more anti-spam features, 8.8.7 has most of these features as well assuming you have a properly setup sendmail.cf). Most distributions ship 8.8.x, although the newer releases are generally shipping the 8.9.x. You can also get the source from ftp.sendmail.org, but compiling sendmail is not for the faint of heart or those that do not have a chunk of time to devote to it.

Sendmail only needs to be accessible from the outside world if you are actually using it to receive mail from other machines and deliver the mail locally. If you only want to run sendmail so that local mail delivery works (i.e. a stand alone workstation, test server or other) and so mail can easily be sent to other machines simply firewall off sendmail, or better, do not run it in daemon mode. Sendmail can be run in a queue flushing node, where it simply 'wakes' up once in a while and processes local mail, either delivering it locally, or sending it off on it's way across the 'net. To set Sendmail to run in queue mode:

edit /etc/rc.d/init.d/sendmail
and change the line:

```
daemon /usr/sbin/sendmail -bd -q1h
```

to:

```
daemon /usr/sbin/sendmail -q1h
```

note if you use your system to send lots of email you may wish
to set the queue flush time lower, perhaps -q15m (15 minutes)
now outbound and system internal mail will behave just fine,
which unless you run a mail server, is perfect.

Now for all those wonderful anti-spam features in sendmail. Sendmail's configuration files consist of (this applies to Sendmail 8.9.x):

`/etc/sendmail.cf`
Primary config file, also tells where other config files are

`/etc/mail/`
You can define the location of configuration files in sendmail.cf, typically people place them in /etc/ or /etc/mail/ (which keeps it a little less cluttered).

`access`
Access list database, allows you to reject email from certain sources (IP or domain), and control relaying easily. My access file looks like this:

```
10.0.0          RELAY
spam.com                REJECT
```

which means 10.0.0.* (hosts on my internal network) are allowed to use the email server to send email to wherever they want, and that all email to or from *.spam.com is rejected. There are lists online of known spammers, typically they are 5-10,000 entries long, this can seriously impede sendmail performance (as each connection is checked against this list), on the other hand having your sendmail machine used to send spam is even worse.

`aliases`
aliases file, allows you to control delivery of mail local to the system, useful for backing up incoming users email to a separate spool. SmartList uses this file to get mail sent to lists delivered to the programs that actually process them.

`domaintable`
domain table (adding domains) that you handle, useful for virtual hosting.

`majordomo`
configuration file for majordomo, I would personally recommend SmartList over Majordomo.

`sendmail.cw`
file containing names of hosts for which we receive email, useful if you host more then one domain.

`sendmail.hf`
location of help file (telnet to port 25 and type in "HELP")

`virtusertable`
Virtual user table, maps incoming users, i.e. maps sales@example.org to john@example.org.

Sendmail 8.9.x (and previous versions) do not really support logging of all email very nicely (something required in today's world for legal reasons by many companies). This is one feature being worked on for the release of Sendmail 8.10.x. Until then there are 2 ways of logging email, the first is somewhat graceful and logs email coming IN to users on a per user basis. The second method is not graceful and involves a simple raw log of all SMTP transactions into a file, you would have to write some sort of processor (probably in perl) to make the log useful.

Mail (incoming SMTP connections to be more precise) is first filtered by the `access` file, in here we can REJECT mail from certain domains/IP's, and RELAY mail from certain hosts (i.e. your internal network of windows machines). Any local domains you actually host mail for will need to go into `sendmail.cw`. Assuming mail has met the rules and is queued for local delivery the next file that gets checked is virtusertable, this is a listing of email addresses mapped to the account name/other email address. i.e.:

```
seifried@seifried.org                   alias-seifried
listuser@seifried.org                   listuser
@seifried.org                           mangled-emails
```

The last rule is a catch all so mangled email addresses do not get bounced, and instead sent to a mailbox. Then the aliases file is checked, if an entry is found it does what it says to, otherwise it attempts to deliver the mail to a local users mailbox, my aliases file entry for seifried is:

```
alias-seifried:         seifried, "/var/backup-spool/seifried"
```

This way my email gets delivered to my normal mailbox, and to a backup mailbox (in case I delete an email I really didn't mean to), or god forbid, Outlook decides to puke someday and hose my mailboxes. This would also be useful for corporations, as you now have a backup of all incoming email on a per user basis, and can allow them (or not) to access the file containing the backed up mail.

One caveat, when using a catch all rule for a domain (i.e. `@seifried.org`) you must create an alias for EACH account, and for mailing lists. Otherwise when it looks through the list and doesn't find a specific entry (for say mailing-list@seifried.org) it will send it to the mailbox specified by the catch all rule. For this reason alone you might not wish to use a catch all rule.

The second method is very simple, you simply start sendmail with the `-X` option and specify a file to log all transactions to. This file will grow very large very quickly, I would NOT recommend using this method to log email unless you absolutely must.

Firewalling sendmail is straightforward, it runs on port 25, tcp:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 25
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 25
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 25

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 25
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 25
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 25
```

## QMAIL

Qmail (like postfix) was created as a direct response to perceived flaws in Sendmail. Qmail is GPL with a no binary distribution clause meaning you must install it from source code. Very little code in Qmail runs as root, and it is very modular compared to sendmail (which is a pretty monolithic piece of code). You can download it from: http://www.qmail.org/.

Postfix

Postfix was written by Wietse Venema, who is also responsible for TCP_WRAPPERS. Postfix is somewhat ore modern then Sendmail or Qmail in the way it attempts to deliver email, and was written to be very secure. Like most IBM products it has a very funky license, but appears to be mostly open source and free. Postfix is available at: http://www.postfix.org/.

Zmailer
Zmailer is a GPL mailer available at: http://www.zmailer.org/. It has crypto hooks and generally looks like it is well built.

DMail

DMail is a commercial mail server, and is not open source. You can download a trial version from: http://netwinsite.com/dmail_first.htm.

## POPD

### WU IMAPD (stock popd)

POP and IMAP are fundamentally related but very different, so I have split them apart. POP stands for "Post Office Protocol" and simply allows you to list messages, retrieve them, and delete them. There are many POP servers for Linux available, the stock one that ships with most distributions if perfect for the majority of users. The main problems with POP are similar to many other protocols, in that usernames and passwords are transmitted in the clear, making it a very good target for packet sniffing. POP can be SSL'ified, however not all mail clients support SSL secured POP. Most POP servers come configured to use TCP_WRAPPERS, which is an excellent method for restricting access. Please see the earlier section on TCP_WRAPPERS for more information. POP runs as root (since it must access user mailboxes) and there have been a number of nasty root hacks in various POP servers in the past. POP runs on ports 109 and 110 (109 is basically obsolete though), using the tcp protocol. The Washington University IMAPD server also comes with a pop server and is generally the 'stock' pop server that ships with most Linux distributions. You can get it from: http://www.washington.edu/imap/.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 110
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 110
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 110

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 110
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 110
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 110
```

### Cyrus

Cyrus is an imap (it also supports pop and kpop) server aimed at 'closed' environments. That is to say that the users will not have any access to the mail server other then by imap or pop protocols. This allows Cyrus to store the mail in a much more secure manner and allows for easier management of larger installations. Cyrus is GNU licensed and available from: http://andrew2.andrew.cmu.edu/cyrus/imapd/.

### IDS POP

IDS (It Doesn't Suck) POP is a lighter popd replacement aimed at smaller installations. It is GPL and available from: http://www.nodomainname.net/software/ids-pop/.

## IMAPD

WU IMAPD (stock imapd)

IMAP is POP on steroids. It allows you to easily maintain multiple accounts, have multiple people access one account, leave mail on the server, just download the headers, or bodies and no attachments, and so on. IMAP is ideal for anyone on the go or with serious email needs. The default POP and IMAP servers that most distributions ship (bundled together into a single package named imapd oddly enough) fulfill most needs.

IMAP also starts out as root, although imapd typically drops to the privilege of the user accessing it, and cannot be easily set to run as a non root user since they have to open mailboxes (and in IMAP's case create folders, files, etc. in the user's home directory), so they cannot drop privileges as soon as one would like. Nor can they easily be chrooted (IMAP needs access to /var/spool/mail, and IMAP needs access to the user's home directory). The best policy is to keep the software up to , and if at all possible firewall pop and imap from the outside world, this works well if no-one is on the road and needs to collect their email via the Internet. Washington University (WU) IMAPD is available from: http://www.washington.edu/imap/.

IMAP runs on port 143 and most IMAPD servers support TCP_WRAPPERS, making it relatively easy to lock down.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 143
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 143
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 143

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 143
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 143
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 143
```

Cyrus

Cyrus is an imap (it also supports pop and kpop) server aimed at 'closed' environments. That is to say that the users will not have any access to the mail server other then by imap or pop protocols. This allows Cyrus to store the mail in a much more secure manner and allows for easier management of larger installations. Cyrus is GNU licensed and available from: http://andrew2.andrew.cmu.edu/cyrus/imapd/.

### WWW based mail readers

One of the better solutions is to use a www based mail client, these can usually be run under a secure www server with minimal extra work, and have the added bonus of letting users check email safely from locations that would normally make checking email difficult (while on vacation in Europe for example). Unfortunately the majority of www based mail reading clients stink, and the good ones cost an arm and a leg.

#### Non Commercial

##### IMP

IMP requires the Horde module (available on the same site) and a www server capable of PHP3 support. You can download IMP and Horde from: http://www.horde.org/imp/.

##### AtDot

AtDot is GNU licensed and written in Perl. It has several modes of operation making it suitable for a variety of www based email solutions (hotmail style providers, ISP's, etc.). You can download it from: http://www.nodomainname.net/software/atdot/.

#### Commercial

##### DmailWeb

http://netwinsite.com/dmailweb/index.htm

##### WebImap

http://netwinsite.com/webimap/index.htm

**DNS**

Bind

DNS is an extremely important service for IP networks, I would not hesitate to say probably the **MOST** important network service (without no-one can find anything). It also requires connections coming in from the outside world, and due to the nature and structure of DNS the information DNS servers claim to have may not be true. The main provider of DNS server software (named, the de facto standard) is currently looking at adding a form of DNS information authentication (basically using RSA to cyrptographically sign the data, proving it is 'true').

Most distributions are finally shipping bind 8.x, however none (to my knowledge) have shipped it setup for non root, chrooted use by default. Making the switch is easy however:

`-u`
specifies which UID bind will switch to once it is bound to port 53 (I like to use a user called 'named' with no login permissions, similar to 'nobody').

`-g`
specifies which GID bind will switch to once it is bound to port 53 (I like to use a group called 'named', similar to 'nobody').

`-t`
specifies the directory that bind will chroot itself to once started. /home/named is a good bet, in this directory you should place all the libraries and config files bind will require.

An even easier way of running bind chroot'ed is to download the bind-chroot package, available as a contrib package for most distributions, and install it. Before installation you will need a user and group called named (which is what the bind server changes it UID/GID to), simply use `groupadd` and `useradd` to create the user/group. Some packages uses holelogd to log bind information to /var/log/messages (as bind would normally do), if this isn’t available you will have to install it be hand which is a chore. In addition to this the default configuration file for bind is usually setup securely (i.e. you cannot query bind for the version information).

Another aspect of bind is the information it contains about your network(s). When a person queries a DNS server for information they typically send a small request for one piece of information, i.e.: what is the IP address for www.seifried.org? And there are domain transfers, where a DNS server requests all the information for say seifried.org, and grabs it and can then make it available to other (in the case of a secondary DNS server). This is potentially very dangerous, it can be as or more dangerous then shipping a company phone directory to anyone that calls up and asks for it. Bind version 4 didn't really have much security, you could limit transfers to certain server, but not selectively enough to be truly useful. This has changed in Bind 8, documentation is available at http://www.isc.org/bind.html. To make a long story short in Bind 8 there are global settings and most of these can also be applied on a per domain basis. You can easily restrict transfers AND queries, log queries, set maximum data sizes, and so on. Remember, when restricting zone queries you must secure ALL name servers (master and the secondaries), as you can transfer zones from a secondary just as easily as a master.

Here is a relatively secure named.conf file (stolen from the bind-chroot package at ftp.tux.org):

```
options {
        // The following paths are necessary for this chroot
        directory "/var/named";
        dump-file "/var/tmp/named_dump.db";             // _PATH_DUMPFILE
        pid-file "/var/run/named.pid";                  // _PATH_PIDFILE
        statistics-file "/var/tmp/named.stats";         // _PATH_STATS
        memstatistics-file "/var/tmp/named.memstats";   // _PATH_MEMSTATS
        // End necessary chroot paths
        check-names master warn;                        /* default. */
        datasize 20M;
};

zone "localhost" {
        type master;
        file "master/localhost";
        check-names fail;
        allow-update { none; };
        allow-transfer { any; };
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "master/127.0.0";
        allow-update { none; };
        allow-transfer { any; };
};

// Deny and log queries for our version number except from localhost
zone "bind" chaos {
        type master;
        file "master/bind";
        allow-query {localhost; };
};

zone "." {
        type hint;
        file "named.zone";
};

zone "example.org" {
       type master;
       file "zones/example.org";
      allow-transfer {
                10.2.1.1;
                10.3.1.1;
      };
};
```

Bind runs on port 53, using both udp and tcp, udp is used for normal domain queries (it's lightweight and fast), tcp is used for zone transfers and large queries (say dig www.microsoft.com). Thus firewalling tcp is relatively safe and will definitely stop any zone transfers, but the occasional DNS query might not work. It is better to use named.conf to control zone transfers.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 53
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 53
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 53

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 53
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 53
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 53
```

would block zone transfers and large queries, the following would block normal queries (but not zone transfers, so if locking it down remember to use both sets of rules)

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 53
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 53
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 53

or

ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 53
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 53
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 53
```

## Dents

Dents is a GPL DNS server, currently in testing stages (release 0.0.3). Dents is being written from the ground up with support for SQL backends, integration with SNMP, uses CORBA for it’s internals. All in all it should give Bind a serious run for the money, I plan to test and evaluate it, but until then you’ll just have to try it yourself. Dents is available at: http://www.dents.org/.

**NNTP**

INN

The usenet server INN has had a long and varied history, for a long period there were no official releases and it seemed to be in a state of limbo, however it is back for good now it would seem. The server software is responsible for handling a potentially enormous load, if you take a full newsfeed the server must process several hundred articles per second, some several kilobytes in size. It must index these articles, write them to disk, and hand them out to clients that request them. INN itself is relatively secure, since it handles data with a directory and generally doesn't have access outside of that, however as with any messaging system if you use it for private/confidential material you must be careful. INN is currently maintained by ISC and is available at: http://www.isc.org/inn.html.

One of the main security threats with INN is resource starvation on the server, if someone decides to flood your server with bogus articles, or there is a sudden surge of activity you might be in trouble if capacity is lacking. INN has had several bad security holes in past, but with today's environment the programmers seem to have chased down and eliminated all of them (none have surfaced recently). It is highly recommended (for more then security reasons alone) that you place the news spool on a separate disk system, let alone partition, you might also wish to use ulimit to restrict the amount of memory available so that it cannot bring the server to it's knees.

As for access you should definitely not allow public access, any news server that is publicly accessible will be quickly hammered by people using it to read news, send spam and the like. Restrict reading of news to your clients/internal network, and if you are really worried force people to login. Client access to INN is controlled via the `nnrp.access` file, you can specify IP address(s), domain names and domains (such as *.me.com), as well as there access levels (read and post), the newsgroups they do or don't have access to and you can also specify a username and password, but since this is linked to the host/domain it gets somewhat messy.

example of nnrp.access:

```
*:: -no - : -no- :!*
# denies access from all sites, for all actions (post and read), to all
groups.
*.me.com::Read Post:::*
# hosts in me.com have full access to all groups
*.them.com::Read:::*, !me.*
# hosts in them.com have read access to everything but the me hierarchy
*.aol.com:Read Post:myname:mypassword:*
# give me access from my AOL account using a username and password
```

If you are going to run a news server I highly recommend the O'Reilly book "Managing Usenet", as Usenet is similar to Sendmail, a total beast to get running smoothly and keep happy.

News should be firewalled as most servers typically server an internal group, and access connections from one or two upstream feeds:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 119
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 119
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 119

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 119
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 119
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 119
```

DNews

http://netwinsite.com/dnews.htm

**DHCPD**

DHCPD is something all network admins should use. It allows you to serve information to clients regarding their network settings/etc, typically meaning that the only client setup needed for networking is leaving the defaults and turning the machine on. It also allows you to reconfigure client machines easily (say move from using 10.0.1.0 to 10.0.2.0, or a new set of DNS servers). In the long run (and short run even) DHCP will save you enormous amounts of work, money and stress. I run it at home with only 8 client machines and have found life to be much easier. DHCPD is maintained by the ISC and is at: http://www.isc.org/dhcp.html.

I also highly recommend you run DHCPD version 2.x (3.x is in pre alpha stages), it's got a lot of new features, and is easier to setup and work with. The absolute latest version(s) of this tend to be a bit neurotic however, be warned it is beta software. Definitely firewall DHCPD off from the Internet. DHCP traffic should only be on local segments, possibly forwarded to a DHCP server on another segment, but the only DHCP traffic you would see coming over the Internet would be an attack/DOS (they might reserve all your IP's, thus leaving your real clients high and dry). If you are forwarding DHCP traffic over the Internet, DON'T. This is a really bad idea for a variety of reasons (primarily performance / reliability, but security as well).

I recommend the DHCPD server be only a DHCP server, locked up somewhere (if you rely on DHCP for your network and the DHCP server fails your network is in serious trouble), allowed to do it's job quietly, if you need to span subnets (i.e. you have multiple ethernet segments, only one of which has a DHCP server physically connected to it) use a DHCP relay (NT has one built in, the DHCP software for Linux has the capability, etc.). There are also several known problems with NT and DHCP, NT RAS has a rather nasty habit of sucking up IP addresses like crazy (I have seen an NT server grab 64 and keep them indefinitely), because it is trying to reserve IP's for the clients that will be dialing in/etc. Either turn NT's RAS off, or put it on it's own subnet, the MAC address it sends to the DHCP server is very strange (and spells out RAS in the first few bytes) and is not easy to map out.
Chroot'ing DHCPD

DHCPD consists of 2 main executables:

- dhcpd - the DHCP
- dhcrelay - a DHCP relay (to relay requests to a central DHCP server since DHCP is based on broadcasts, which typically don't (and shouldn't) span routers/etc.

DHCPD requires 2 libraries:

- /lib/ld-linux.so.2
- /lib/libc.so.6

A config file:

- /etc/dhcpd.conf - configuration info, location of boot files, etc.

And a few other misc. files:

- /etc/dhcpd.leases - a list of active leases
- a startup file, you can modify the one it comes with or roll your own

The simplest way to setup named chrooted is to simply install dhcpd (latest one preferably) and move/edit the necessary files. A good idea is to create a directory (such as /chroot/dhcpd/), preferably on a separate filesystem from /, /usr, etc (symlinks...), and then create a file structure under it for dhcpd. The following is an example, simply replace /chroot/dhcpd/ with your choice. You must of course execute these steps as root for it to work.

```
# Install bind so we have the appropriate files
#
rpm -i dhcpd-2.0b1pl0-1.i386.rpm
#
# Create the directory structure
#
cd /chroot/dhcpd/            # or wherever
mkdir ./etc
mkdir ./usr/sbin
mkdir ./usr
mkdir ./var/dhcpd
mkdir ./var
mkdir ./lib
#
# Start populating the files
#
cp /usr/sbin/dhcpd ./usr/sbin/dhcpd
cp /etc/dhcpd.conf ./etc/dhcpd.conf
cp /etc/rc.d/init.d/dhcpd ./etc/dhcpd.init
cp /etc/rc.d/init.d/functions ./etc/functions
#
# Now to get the latest libraries, change as appropriate
#
cp /lib/ld-linux.ld-linux.so.2 ./lib/
cp /lib/libc.so.6 ./lib/
#
# And create the necessary symbolic links so that they behave
# Remember that named thinks /chroot-dns/ is /, so use relative links
```

Then modify or create your startup script.

Once this is done simply remove the original startup file and create a symlink from where it was pointing to the new one, and dhcpd will behave 'normally' (that is it will be automatically started at boot time), while in fact it is separated from your system. You may also wish to remove the 'original' DHCPD files laying about, however this is not necessary.

If you have done the above properly you should have a /chroot/dhcpd/ (or other dir if you specified something different) that contains everything required to run dhcpd.
And a ps -xau should show something like:

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
root      6872  0.0  1.7   900   532  p0 S    02:32   0:00 ./usr/sbin/dhcpd
-d -q
root      6873  0.0  0.9   736   288  p0 S    02:32   0:00 tee
./etc/dhcpd.log
```

DHCPD should definitely be firewalled from external hosts as there is no reason an external host should be querying your DHCP server for IP’s/etc, in addition to this making it available to the outside world could result in an attacker starving the DHCP server of addresses assuming you use a dynamic pool(s) of addresses, you could be out of luck for your internal network, and learning about the structure of your internal network. DHCP runs on port 67, udp because the amounts of data involved are small and a fast response is critical.

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 67
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 67
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 67

or

ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 67
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 67
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 67
```

**NFSD**

NFS stands for Network File System and is just that, it is a good way to distribute filesystems, read only and read/write, while maintaining a degree of security and control assuming your network is enclosed and secure. NFS is primarily meant for use in a high bandwidth environment (i.e. a LAN) where security risks are not high, or the information being shared is not sensitive (i.e. a small trusted LAN behind a firewall exchanging CAD/CAM diagrams, or a large university lab using nfs to mount /usr/. If you need a high level of security, i.e. encrypted data between hosts, NFS is not the best choice. I personally use it at across my internal LAN (this machine has 2 interfaces, guess which one is heavily firewalled), to share file systems containing rpm's, this website, etc. Safer alternatives include SAMBA (free) and now IBM is porting AFS to Linux (costly but AFS is a sweet hunk of code).

NFS has a few rudimentary security controls, the first one would be firewalling, using NFS across a large, slow public network like the Internet just isn't a good idea in any case, so firewall off port 2049, UDP. Since NFS runs as a set of daemons, tcp_wrappers are of no use unless NFS is compiled to support them. The config file for NFS actually has quite a few directives, the bulk of which deal with user id and group id settings (map everyone to nobody, perhaps map all the engineering clients to 'engineer', etc, etc) but no real mechanisms for authentication (your client claims to be UID 0, this is why root's id is squashed by default to nobody). NFS read only exports are pretty safe, you only have to worry about the wrong people getting a look at your info (if it is sensitive) and or creating a denial of service attack (say you have a directory world readable/etc for sharing kernel source, and some gomer starts sucking down data like crazy...). Writeable exports are a whole other ball game, and should be used with extreme caution, since the only 'authentication' is based on IP/hostname (both easily spoofable), and UID (you to can run Linux and be UID 0). Bounce a client down with a DOS attack, grab their IP, mount the writeable share and go to town. You say "but they'd have to know the IP and UID", packet sniffing is not rocket science folks, nor is 'showmount'.
So, how do we go about securing NFS? The first is to firewall it, especially if the machine is multi-homed, with an interface connected to a publicly accessible network (the Internet, the student lab, etc.). If you plan to run NFS over a publicly accessible network it better be read only, and you will be far better off with a different product then NFS. The second and most interesting part is the /etc/exports file. This controls what you allow clients to do, and how they do it.

A sample exports file.

```
#
# Allow a workstation to edit web content
/www                   10.0.0.11(rw,no_root_squash)
#
# Another share to allow a user to edit a web site
/www/www.bobo.org              10.0.0.202(rw,no_root_squash)
#
# Public ftp directory
/home/ftp                      *.example.org(ro,all_squash)
```

The structure of the exports file is pretty simple, directory you wish to export, client (always use IP’s, hostnames can easily be faked), and any options. The client can be a single IP (10.0.0.1), hostname (gomer.poncho.net), a subnet (10.0.0.0/255.255.255.0), or a wildcard (*.bigdaddy.mil). Some of the more interesting (and useful) directives for the exports file are:

secure - the nfs session must originate from a privileged port, i.e. root HAS to
be the one trying to mount the dir. This is useful if the server you are
exporting to is secured well.

`ro` - a good one, Read Only, enough said.
`noaccess` - used to cut off access, i.e. export /home/ but do a noaccess on /home/root
`root_squash` - squashes root's UID to the anonymous user UID/GID (usually 'nobody'), very useful if you are exporting dirs to servers with admins you do not 100% trust (root can almost always read any file.... HINT)
`no_root_squash` - useful if you want to go mucking about in exported dirs as root to fix things (like permissions on your www site)
`squash_uids` and `squash_gids` - squash certain UID(s) or GID(s) to the anonymous user, in RedHat a good example would be 500-10000 (by default RedHat starts adding users and groups at 500), allowing any users with lower UID's (i.e. special accounts) to access special things.
`all_squash` - a good one, all privileges are revoked basically and everyone is a guest.
`anonuid` and `anongid` - specifically set the UID / GID of the anonymous user (you might want something special like 'anonnfs').

The `man exports` page is actually quite good.

Beyond this there isn't much you can do to secure NFS apart from ripping it out and putting some other product in (like AFS, Coda, etc). NFS is good, every flavor of UNIX supports it, and is very easy to setup, work with and maintain. It's also 'old faithful', been around a long time. Just check "Practical Unix and Internet Security", they also state in bold not to use NFS if security is a real issue.

NFS should be restricted from the outside world, it runs on port 2049, udp, as well as using RPC which runs on port 111, udp/tcp, and makes use of mountd which runs on port 635, udp. Replace the 2049 with 111, and 635 udp and tcp to secure those services (again the best idea is a blanket rule to deny ports 1 to 1024, or better yet a default policy of denial).

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 2049
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 2049
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 2049

or

ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 2049
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 2049
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 2049
```

## tftp

tftp (Trivial File Transfer Protocol) is used for devices that require information from a network server, typically at boot time. It is an extremely simple form of ftp, with most of the security and advanced commands stripped off, it basically allows a device to retrieve (and upload) files from a server in a very simple manner. tftp is almost exclusively used for diskless workstations, router configuration data, and any device that boots up, and requires information it cannot store permanently. As such it presents a rather large security hole, just imagine if someone were to connect to your tftp server and grab the boot file for your main Cisco router. Additionally tftp defaults to granting complete access to the file system, traditionally this meant people would scan large blocks of hosts, and retrieve the `/etc/passwd` file among others. Fortunately modern versions of tftp can be locked down, they accept a directory name that they essentially limit access to, and TCP_WRAPPERS can be used to limit access to certain hosts only. By default tftp (at least for RedHat) defaults to giving access only to the `/tftpboot` directory (which usually doesn't exist, so create it if you need it). It is a very good idea to keep the tftp directory as separate form the system as possible. This is done by specifying the directory or directories you want tftp to have access to after the tftp command in `inetd.conf`. Th following example starts tftp normally and grants it access to the `/tftpboot` directory and the `/kickstart` directory.

```
tftp   dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd /tftpboot
/kickstart
```

Also remember tftp uses UDP, so a '`ps xau`' won't necessarily show who is logged in or what they are doing (as opposed to ftp which shows up) unless they are currently downloading a file (since most tftp applications resolve around small files it is unlikely you will catch someone in the act as it were). The best place to monitor tftp is from syslog, but even then tftp doesn't log IP addresses or anything truly useful. The following is some ps output, and some syslog output of an active tftp session.

```
nobody     744  0.0  0.6   780   412  ?  R    14:31   0:00 in.tftpd
/tftpboot

Apr 21 14:31:15 hostname tftpd[744]: tftpd: trying to get file: testfile
Apr 21 14:31:15 hostname tftpd[744]: tftpd: serving file from /tftpboot
```

TFTP can be easily restricted using TCP_WRAPPERS and firewalling, tftp runs on port 69, UDP so simply restrict access to that needed by your various diskless workstations, routers and the like, it is also a good idea to block all tftp traffic at your network borders, as there is no need for a machine to remote boot using tftp across the Internet/etc. Also tftp runs as the user nobody, but since no authentication is done and all devices accessing the tftp server are doing so as 'nobody' file level security is pretty well useless. All in all a very insecure protocol. TFTP runs on port 69, udp.

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 69
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 69
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 69

or

ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 69
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 69
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 69
```

utftpd

utftpd is a secure replace for the stock tftpd, it provides much finer access control and support for some other interesting features (such as revision control). You can also base access on the clients IP address, meaning your router configurations and diskless workstation configurations can be kept separate and discrete from each other. utftpd is GPL licensed and available at: http://www.nrw.net/uwe/utftpd.html.

### bootp

bootp is basically the precursor to dhcpd, it has less options and less configurability, but basically does the same tasks, helping devices boot up on the network and giving them the information they need. I would not recommend running bootp unless you have some truly ancient equipment that chokes on a DHCP server. If someone really wants me to write some more on this email me and I will. Otherwise I will consider bootp to be of historical interest only.

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 69
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 69
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 69
```

or

```
ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 69
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 69
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 69
```

### cu-snmp

SNMP (Simple Network Management Protocol) was designed to let heterogeneous systems and equipment talk to each other, report data and allow modifications to there settings over a TCP-IP network. For example an SNMP enabled device (such as a Cisco router) can be monitored/configured from an SNMP client, and you can easily write scripts to say alert you if denied packets/second rises above 20. Unfortunately SNMP has no security built into it. SNMPv1, originally proposed in RFC 1157 (May 1990) and section 8 (Security Considerations) reads thusly: "Security issues are not discussed in this memo.". I think that about sums it up. In 1992/1993 SNMPv2 was released, and did contain security considerations however these security considerations were dropped later on when they were shown to be completely broken. Thus we end up today with SNMPv2 and no security. Currently the only way to protect your SNMP devices consists of setting the community name to something hard to guess (but it is very easy to sniff the wire and find the name), and firewall/filter SNMP so that only the hosts that need to talk to each other can (which leaves you open to spoofing). Brute force community name attacks are easy to do and usually effective, and there are several tools specifically for monitoring SNMP transmissions and cracking open an SNMP community, it is a pretty dangerous world out there. These risks are slightly mitigated by the usefulness of SNMP, if properly supported and implemented it can make network administration significantly easier. In almost every SNMP implementation the default community name is "public" (this goes for Linux, NT, etc), you must change this, to something obscure (your company name is a bad idea). Once a person has your community name they can conduct an "snmpwalk" and take over your network. SNMP runs over UDP on ports 161 and 162, block this at all entrances to your network (the backbone, the dialup pool, etc). If a segment of network does not have SNMP enabled devices or an SNMP console you should block SNMP to and from that network. This is your only real line of defense with SNMP. Additionally the use of IPSec (or other VPN software) can greatly reduce the risk from sniffing. The RFC's for SNMPv3 however go extensively into security (especially RFC 2274, Jan 1998) so there is hope for the future. If you are purchasing new SNMP aware/enabled products make sure they support SNMPv3, as you then have a chance at real security.

There are no specific problems with cu-snmpd per se, apart from the general SNMP problems I have covered. The cu-snmp tools and utilities only support SNMPv1 and SNMPv2, so remember to be careful when using them on or across untrusted networks as your main line of security (the community name) will be out in the open for anyone to see.

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 161:162
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 161:162
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 161:162
```

or

```
ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 161:162
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0
161:162
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 161:162
```

### Finger

Finger is one of those things most admins just disable and ignore. It is a useful tool on occasion, but if you want to allow other admins to figure out which of your users is currently trying to crack their machines, use identd. Finger lets out way to much info, and is a favorite tool for initial probes and data gathering on targets. There have also been several nasty DOS attacks released, basically consisting of sending hundreds of finger requests and in certain configurations just watching the server croak. Please don't run finger. Many distributions ship with it enabled, but to quote inetd.conf from RedHat:

```
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
```

If you still have the urge that you absolutely must run it use -u to deny finger @host requests that are only ever used to gather information for future attacks. Disable finger, really. Fingerd has also been the cause of a few recent and very bad denial of service attacks, especially if you run NIS with large maps, DO NOT, repeat NOT run fingerd.  Cfingerd (configurable fingerd) is a great replacement for the stock fingerd, it was built with security in mind, runs as a non root user typically, and users can easily configure it so they aren’t fingurable. Cfingerd is available from: http://ftp.bitgate.com/cfingerd/. Finger runs on port 79, and cfingerd runs on port 2003, both use tcp.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 79
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 79
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 79

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 79
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 79
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 79
```

**Identd**

The identd service is used to map users/processes to ports in use. For example most irc servers attempt to find out who is connecting to them by doing an identd lookup, which basically consists of asking the identd what information it has about a port number, and can range from nothing (if no-one is using that particular port) to a username, groupname, process id, and other interesting information. The default setting in most distributions is that identd is on (it is polite to run it, irc servers and newer versions of sendmail check identd responses), and will only hand out the username. The primary use of identd is to allow remote systems some means of tracking down users that are connecting to their servers, irc, telnet, mail, or other, for authentication purposes (not a good idea since it is very easy to fake. The local university here in Edmonton requires identd if you want to telnet into any of the main shell servers, primarily so they can track down compromised accounts quickly. Identd is a useful tool, but generally only on machines with users you do not trust (i.e. shell account servers). It is also a two edged sword, while it gives out information useful for tracking down attackers (definitely people you want to boot off of your servers) it can also be used to gain information about users on your system, leading to their accounts being compromised. I would suggest only running identd on servers with shell accounts/etc.

Identd supports quite a few features, and can be easily set to run as a non root user. Depending on your security policies you may not want to give out very much information, or you might want to give out as much as possible. Simply tack the option on in inetd.conf, after in.identd (the defaults are -l -e -o).

```
-p port
-a address
```
Can be used to specify which port and address it binds to (in the case of a machine with IP’s aliased, or multiple interfaces), this is generally only useful if you want internal machines to connect, since external machines will probably not be able to figure out what port you changed it to.

```
-u uid
-g gid
```
Are used to set the user and group that identd will drop it's privileges to after connecting to the port, this will result in it being far less susceptible to compromising system security. As for handling the amount of information it gives out:

```
-o
```
Specifies that identd will not return the operating system type, and simply say "UNKNOWN", a very good option.

```
-n
```
Will have identd return user numbers (i.e. UID) and not the username, which still gives them enough information to tell you and allow you to track the user down easily, without giving valuable hints to would be attackers.

```
-N
```
Allows users to make a ~/.noident file, which will force identd to return "HIDDEN-USER" instead of information.

```
-F format
```

Enables you to specify far more information then is standard, everything from user name and number to the actual PID, command name, and command name and arguments that were given! This I would only recommend for internal use, as it is a lot of information attackers could find useful.

In general I would advise disabling identd, primarily due to the number of denial of service attacks it is susceptible to. You should only run it if you want to make the lives of other administrators easier, in tracking down which of your users are being bad. There are however other versions of identd available, some with security enhancements (I do not endorse these as I have yet to test them):

http://insecurity.net/ - Paul's secure identd written in perl
http://www.ojnk.nu/~odin/ - ojnk identd
http://www.tildeslash.org/nullidentd.html - null identd
http://www.ajk.tele.fi/~too/sw/ - fake identd
http://p8ur.op.het.net/midentd/ - midentd

Identd runs on port 113 using tcp, and typically you will only need if you want to IRC (many irc networks require an identd response), or be nice to systems running daemons (such as tcp_wrapped telnet, or sendmail) that do identd lookups on connections.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 113
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 113
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 113

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 113
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 113
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 113
```

### ntpd

NTP (Network Time Protocol) is rather simple in it's mission, it keeps computers clocks in synchronization. So what? Try comparing log files from 3 separate servers if their clocks are out of synch by a few minutes. NTP simply works by a client connecting to a time server, working out the delay between them (on a local LAN it might be only 1-2ms, across the internet it might be several hundred ms), and then it asks for the time and sets it's own clock. Additionally servers can be 'clustered' to keep themselves synchronized, the chances of 3 or more servers losing track of what time it is (also called 'drift') is relatively low. The time signal is typically generated by an atomic clock or GPS signal, measured by a computer, these are 'stratum 1' time servers, below them are stratum 2 time servers that typically are publicly accessible, a company might maintain it's own stratum 3 time servers if they have sufficient need, and so on. The data NTP exchanges is of course not terribly sensitive, it's a time signal, however if an attacker were able to tamper with it all sorts of nastiness could result, log files might be rendered unusable, accounts might be expired early, cron jobs that backup your server might run in prime time causing delays, etc. Thus it is a good idea to run your own time server(s), and set the maximum adjustment they will make to only a few seconds (they shouldn't drift very much in any case). If you are really paranoid, or have a great number of clients you should consider buying a GPS time unit, they come in all shapes and sizes, from a 1U rack mount job that plugs directly into your LAN to ISA and PCI cards that plug into a server and have an antenna. It is a good idea to firewall off your timeserver, as a denial of service attack would be detrimental to your network, in addition to this if possible you should use the encryption available in ntpd, based on DES it is generally sufficient to thwart most attackers. NTP is available from: http://www.eecis.udel.edu/~ntp/. There usually are no man pages with ntpd or xntpd (wonderful huh) but documentation can be found in /usr/doc/name/ typically, or at: http://www.eecis.udel.edu/~ntp/ntp_spool/html/index.htm. NTP runs on port 123 using udp and tcp, firewalling it is relatively simple:

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 123
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 123
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 123
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 123
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 123
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 123
```

or

```
ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 123
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 123
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 123
ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 123
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 123
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 123
```

## CVS

CVS allows multiple developers to work together on large source code projects and maintain a large code base in a somewhat sane manner. CVS's internal security mechanisms are rather simple (and some would say weak) on their own, and I would have to agree. CVS's authentication is typically achieved over the network using pserver, usernames are sent in clear text, and passwords are trivially hashed (no security really). To get around this you have several good options. In a Unix environment probably the simplest method is to use SSH to tunnel connections between the client machines and the server, "Tim TimeWaster" (Tim Hemel) has written an excellent page covering this that I won't bother to rehash, and it is available at: http://cuba.xs4all.nl/~tim/scvs/. A somewhat more complicated approach (but better in the long run for large installations) is to kerberize the CVS server and clients, typically large networks (especially in university environments) already have an established Kerberos infrastructure. Details on kerberizing CVS are available at: http://www.cyclic.com/cyclic-pages/security.html. Apart from that I would strongly urge firewalling CVS unless you are using it for some public purpose (such as an open source project across the Internet).

Another tool for securing CVS that just appeared is “cvsd”, a wrapper for pserver that chroot’s and/or suid’s it to a harmless user. cvsd is available at: http://cblack.mokey.com/cvsd/ in rpm format and a source tarball.

There are other less obvious concerns you should be aware of, when dealing with source code you should be very to ensure no Trojan horses or backdoors are emplaced. In an open source project this is relatively simple, review the code people submit, especially if it is a publicly accessible effort, such as the Mozilla project. Other concerns might be destruction of the source code, make sure you have back ups. CVS uses port 2401, tcp.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 2401
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 2401
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 2401
```

or

```
ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 2401
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 2401
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 2401
```

**rsync**

rsync is an extremely efficient method for mirroring files, be it source code files of a CVS tree, a web site, or even this document. rsync preserves file permissions, links, file times and more, in addition to this it supports an anonymous mode (which incidentally I use for the mirroring of this document) that makes life very easy for all concerned. The rsync program itself can act as the client (run from a command line or script) and as the server (typically run from inetd.conf). The program itself is quite secure, it does not require root privileges to run as a client nor as the server (although it can if you really want it to), and can chroot itself to the root directory of whatever is being mirrored (this however requires root privileges and can be more dangerous then it is worth). You can also map the user id and group id it will access the system as (the default is nobody for most precompiled rsync packages and is probably the best choice). In non anonymous mode rsync supports usernames and passwords, that are encrypted quite strongly using 128 bit MD4. The "`man rsyncd.conf`" page quite clearly covers setting up rsync as a server and making it relatively safe. The default configuration file is `/etc/rsyncd.conf`, and has a global section, and module sections (basically each shared out directory is a module).

rsyncd.conf example:

```
motd file = /etc/rsync.motd   # specifies a file to be displayed, legal
disclaimer, etc
max connections = 5           # maximum number of connections so you don't
get flooded
[pub-ftp]
       comment = public ftp area    # simple comment
       path = /home/ftp/pub         # path to the directory being exported
      read only = yes         # make it read only, great for exported
directories
      chroot = yes                  # chroot to /home/ftp/pub
      uid = nobody                  # explicitly set the UID
      gid = nobody                  # explicitly set the GID
[secret-stuff]
      comment = my secret stuff
      path = /home/user/secret            # path to my stuff
      list = no                     # hide this module when asked for a
list
      secrets file = /etc/rsync.users     # password file
      auth users = me, bob, santa         # list of users I trust to see my
secret stuff
      hosts allow = 1.1.1.1, 2.2.2.2      # list of hosts to allow
```

As you can see rsync is quite configurable, and generally quite secure, the exception being the actual file transfers which are not encrypted in any way. If you need security I suggest you use SSH to tunnel a connection, or some VPN solution like FreeS/WAN. Also make sure you are running rsync 2.3.x or higher as a potential root compromise was found in 2.2.x. Rsync is available at: http://rsync.samba.org/. Rsync runs on port 873, tcp.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 873
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 873
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 873
```

```
or
```

```
ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 873
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 873
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 873
```

## lpd

lpd is the UNIX facility for printing (Line Printer Daemon). It allows you to submit printjobs, run them through filters, manage the print queues, and so on. lpd can accept print jobs locally, or over the network, and access various parts of the system (printers, logging daemons, etc), hence making it a potential security hole. Historically lpd has been the source of several nasty root hacks, however it seems to have been mostly fixed now, there are still many potential denial of service attacks though due to it’s function (something simple like submitting huge printjobs and running the printer out of paper). Fortunately lpd is slowly being phased out with the advent of network aware printers, however there is still a huge of printing done via lpd. lpd access is controlled via `/etc/hosts.equiv`, and `/etc/hosts.lpd`. You should also firewall lpd from the outside world, and if you need to send printjobs across public networks, remember anyone can read them, so a VPN solution is a good idea. lpd runs on port 515 using tcp. The hosts.lpd file should contain a list of hosts (workstation1.yourdomain.org, etc), one per line that are allowed to use the lpd services on the server, you might as well use ipfwadm/ipchains.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 515
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 515
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 515
```

or

```
ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 515
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 515
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 515
```

### LPRng

An alternative to the stock lpd is “LPRng” (LPR Next Generation), it provides new enhancements and also supports a higher level of security. LPRng supports Kerberos and PGP based authentication, as well as a restrictions files, /etc/lpd.perms, which allows you to control access based on user, group, authentication, IP, and so on, allowing for extremely flexible and secure configurations. LPRng has excellent documentation and is available at: http://www.astart.com/lprng/LPRng.html.

### pdq

pdq is another LPD replacement, no real emphasis on enhanced security but it does seem to offer some improvements management and performance wise over the stock LPD. You can get pdq from: http://feynman.tam.uiuc.edu/pdq/.

## X Window System

The X Window System provides a network transparent method for sharing graphical data, or more specifically for exporting the display of a program to a remote (or the local) host. Using it you can run a powerful 3d rendering package on your SGI origin 2000 and display it on a 486. Essentially it's the granddaddy to all this 'thin client' hype that is becoming very popular nowadays. It was created by MIT, and at the time security was not much of a concern, this of course has led to more then a few nasty bugs being found, as well the level of control X is given (it handles keystrokes, mouse movements, draws the screen, etc) means if it is compromised very bad things can happen. This data, if sent over the network (i.e. the X program being run is displaying on a remote host) can easily be logged, so sensitive information (like an xterm being used to login to another remote system) is vulnerable. In addition to these problems the authentication protocol that X uses is relatively weak (although it has been improved). Running a graphical xemacs session on a server 3 timezones away however can be a very handy thing.

X is very predictable in port usage, almost all implementations and installations of X use port 6000 for the first session and increment by one for other sessions, thus making it quite easy to scan for. If you are not going to be using X to display program running on remote systems I suggest strongly you firewall port 6000. Control over who/what is allowed to connect to the X server can be accomplished several ways.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 6000:6100
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 6000:6100
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 6000:6100

or

ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 6000:6100
ipchains -A input -p tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0
6000:6100
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 6000:6100
```

xhost

xhost simply allows you to specify which machines are, or aren't allowed to connect to the X server, this is a very simplistic security mechanism and is not really suitable in any modern environment, however used in conjunction with other mechanisms it can help. The command is quite simple: '`xhost +hostname.com`' adds hostname.com, '`xhost -hostname.com`' removes hostname.com from the list, you must also specify '`xhost -`' to turn on the access control list, or else everyone is let in by default.

mkxauth

mkxauth is definitely a step up, it helps create .Xauthority files, and merge them, which are used to specify hostnames and the related magic cookies (basically a token used to gain access). These cookies can then be used to gain access to a remote X host (you essentially have a copy of the cookie on each end) and are transferred either plain text (insecure) or DES encrypted (quite secure). Using this method you can be relatively safe and secure. Xauthority files can also be used in conjunction with Kerberos, removing the necessity to copy Xauthority files around and keep them in synchronization. Hosts authenticate to each other through a central Kerberos key server(s) in an encrypted fashion, this method is most appropriate for large installations/etc. mkxauth has an excellent man page '`man mkxauth`' and more generalized details are available in the Xsecurity man page (not sure how common this name page is) '`man Xsecurity`'.

SSH

SSH can be used to create a tunnel between hosts (or more specifically between two X servers), thus encrypting the channel, providing authentication, and generally making things safer. The following web page explains it in detail: http://csociety.ecn.purdue.edu/~sigos/projects/ssh/forwarding/.

**Samba**

SAMBA is one of the best things since sliced bread, that is if you want to share files and printers between Windows and *NIX. It is also somewhat misunderstood, and suffers heavily from interaction with various (sometimes broken) Windows clients. SAMBA has a great many kludges that attempt to make it somewhat sane, but can lead to what looks like broken behavior sometimes. SAMBA simply gives access to the filesystem VIA SMB (Server Message Block), the protocol Windows uses to share files and printers. It verifies the username and password given (if required) and then gives access to the files according to the file permissions and so forth that are set. I'm only going to cover Samba 2.x, Samba 1.x is pretty old and obsolete.

Samba 2.x is controlled via `smb.conf`, typically in `/etc (man smb.conf)`. In `/etc/smb.conf` you have 4 main areas of configuration switches: `[globals]`, `[printers]`, `[homes]`, and each `[sharename]` has it's own configuration (be it a printer or drive share). There are a hundred or so switches, the smb.conf man page covers them exhaustively. Some of the important (for security) ones are:

`security = xxxx` where `xxxx` is share, server or domain, share security is per share, with a password that everyone uses to get at it, server means the samba server itself authenticates users, either via `/etc/password`, or `smbpasswd`. If you set it to domain, samba authenticates the user via an NT domain controller, thus integrating nicely into your existing NT network (if you have one).

`guest account = xxxx` where `xxxx` is the username of the account you want the guest user to map to. If a share is defined as public then all requests to it are handled as this user.

`hosts allow = xxxx` where `xxxx` is a space separated list of hosts / IP blocks allowed to connect to the server.

`hosts deny = xxxx` where `xxxx` is a space separated list of hosts / IP blocks not allowed to connect to the server.

`interfaces = xxxx` where `xxxx` is a space separated list of IP blocks that samba will bind to

I would also highly recommend installing and using SWAT (samba Web Administration Tool) as it will cut down on the mistakes/etc that you are liable to make. Samba and SWAT are available at: http://www.samba.org/ and ship with almost every distribution.

**SWAT**

SWAT is a very nice administration tool to setup your smb.conf. The main problem is that is requires you to use the root account and password to ‘log’ in, and runs as a separate process out of inetd.conf, so there is no easy way to encrypt it, and as far as I can tell no way to grant others users administrative access to SWAT. Having said that however it is a good tool for cutting down on mistakes made while editing smb.conf. You can also run SWAT with the –a switch, meaning no password will be required, and using TCP_WRAPPERS to restrict access to certain workstations (although you’d still be open to IP spoofing), essentially SWAT was not meant as a secure administrative tool, but it is useful. SWAT comes with samba (usually) and is available at: http://www.samba.org/, a demo of SWAT is online at: http://anu.samba.org/cgi-bin/swat/.

## File sharing methods

I thought I should also give a brief overview of the various file sharing methods that exist for Linux, show you the pro’s and con’s as it were.

### SAMBA

Samba is the best option (performance wise, security wise, etc.) for sharing files with Windows client machines, I would not really recommend it for sharing files between Linux machines.

### NFS

NFS is not very secure, but it is easy to use, and can be made relatively safe if the environment it is in is not to hostile. I would not recommend this as a method for sharing files/etc if security is a concern (which arguably means you shouldn’t use it for diskless workstations, but such is life).

### Coda

An advanced network filesystem, not very fun to implement. http://www.coda.cs.cmu.edu/.

### Drall

An https based systems for sharing files among machines securely. http://www.edlund.org/projects/drall/index.html.

### AFS

A high end, commercial file sharing application for large installations. The FAQ is available at: http://www.angelfire.com/hi/plutonic/afs-faq.html.

## Network Based Authentication

### NIS / NIS+

NIS and NIS+ (formally known as “yellow pages”) stands for Network Information Service. Essentially NIS and NIS+ provide a means to distribute password files, group files, and other configuration files across many machines, providing account and password synchronization (among other services). NIS+ is essentially NIS with several enhancements (mostly security related), otherwise they are very similar. To use NIS you set up a master NIS server that will contain the records and allow them to be changed (add users, etc), this server can distribute the records to slave NIS machines that contain a read only copy of the records (but they can be promoted to master and set read/write if something bad happens). Clients of the NIS network essentially request portions of the information and copy it directly into their configuration files (such as /etc/passwd), thus making them accessible locally. Using NIS you can provide several thousand workstations and servers with identical sets of usernames, user information, passwords and the like, significantly reducing administration nightmares. However this is part of the problem, in sharing this information you make it accessible to attackers, which is something NIS+ attempts to resolve, the problem is NIS+ is an utter nightmare to set up. An alternative strategy would be to use some sort of VPN support (like FreeS/WAN, doesn’t it seem to solve almost any problem?) and encrypt the data before it gets onto the network. There is an NIS / NIS+ howto at: http://metalab.unc.edu/LDP/HOWTO/NIS-HOWTO.html, and O’Reilly has an excellent book on the subject. NIS / NIS+ runs over RPC which uses port 111, both tcp and udp. This should definitely be blocked at your network border, but will not totally protect NIS / NIS+.

```
ipfwadm -I -a accept -P udp -S 10.0.0.0/8 -D 0.0.0.0/0 111
ipfwadm -I -a accept -P udp -S some.trusted.host -D 0.0.0.0/0 111
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 111
```

or

```
ipchains -A input -p udp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 111
ipchains -A input -p udp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 111
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 111
```

Since NIS and NIS+ are RPC services they tend to user higher port numbers (i.e. above 1024) in a somewhat random fashion, making firewalling of it rather painful. The best solutions is to place your NIS server(s) on machines internally that are blocked completely from talking to the Internet, inbound and outbound.

SRP

SRP is a relative newcomer, however it has several advantages over some of the older programs. SRP is free for non commercial use and does not use encryption per se to secure the data, so exporting it outside of the US isn’t as much of a problem. SRP uses one way hashes and provides authentication of both parties. The disadvantage is that SRP only encrypts the login (username and password) so any data transferred (such as the telnet session or ftp sites) are vulnerable. You can get SRP from: http://srp.stanford.edu/srp/. SRP currently has Telnet and FTP support (for windows as well) although SRP enabling other protocols is relatively straightforward.

Kerberos

Kerberos is a modern network authentication system based on the idea of handing a user a ticket once they have authenticated to the Kerberos server (similar to NT's use of tokens). Kerberos is available from: http://web.mit.edu/kerberos/www/. The Kerberos FAQ is available at: http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html. Kerberos is appropriate for large installations as it scales better and is more secure then NIS / NIS+. Kerberizing programs such as telnet, imap and pop can be achieved with some effort, Windows clients with Kerberos support are harder to find however.

# Encrypting services / data

## Encrypting network services

Virtually all network traffic is unencrypted and easily read by an attacker. If someone cracks a machine on your internet and installs a password sniffer (basically your common packet sniffer with a filter) your entire network can be compromised in a matter of hours. One ISP that shall remain nameless placed co-hosted customer machines on the same LAN, using a normal ethernet hub, meaning all machines could see each others traffic (users retrieving email via pop, telnet sessions, etc). This is one of the major reasons why encrypting data traffic is a good idea.

Various mechanisms exist and/or are being developed to encrypt network data traffic, at various levels of the network stack. Some schemes only encrypt the data sent (such as PGP encrypted email), some encrypt the session (SSL), and some encrypt the data payload of the packets (IPSec and other VPN's). Ultimately the best solution will be IPSec (my opinion), as it requires no modifications to the applications, and provides for a very high level of security between computers.  Currently there are  no widely used data encryption solutions, as Microsoft does not support many, which is a serious hindrance to any mass solution (to be fair Microsoft does have beta IPSec support, but it is not ready yet). The best scheme currently available is SSL, Secure Sockets Layer, originally proposed by Netscape. SSL encrypts the data at the session level, thus is your application supports SSL and the server supports SSL you are in luck. Most www browsers, some email/news readers, and a few ftp and telnet clients support SSL currently, for Linux most services can be SSL'ified. SSL does however require clients with SSL capabilities, something you won't be able to get most people to support, which means SSL'ified services are typically restricted to within an organization. The SSL libraries are available at http://www.openssl.org/.

### SSL

HTTP - SSL

The most common www server, Apache, has very good SSL support, which can be downloaded free of charge outside the US (US patents on RSA/etc mean you have to pay royalties within the US, so the free software is illegal) from http://www.apache-ssl.org/. There are many commercial www servers that support SSL, the majority also based off of Apache, such as RedHat Secure Server, Stronghold, and so forth.

Telnet - SSL

A drop in replacement for telnet, SSLtelnet and MZtelnet provide a much higher level of security then plain old telnet, although SSLtelnet and MZtelnet are not as flexible as SSH, they are perfectly free (i.e. GNU licensed) which SSL is not. The server and client packages are available as tarballs at: ftp://ftp.uni-mainz.de/pub/internet/security/ssl/, and as RPM packages at ftp://ftp.replay.com/pub/replay/linux/redhat/.

FTP - SSL

Also a drop in replacement for your favorite ftpd (probably WuFTPD), also available as a set of patches for WuFTPD. This is highly appropriate as most servers have many users that require ftp access. The tarball is available at: ftp://ftp.uni-mainz.de/pub/internet/security/ssl/, and as RPM packages at ftp://ftp.replay.com/pub/replay/linux/redhat/.

## Virtual private network solutions

IPSec

IPSec is a proposed (and now implemented in some operating systems) standard for IP level encryption. Linux IPSec support varies, with some distributions shipping it, but the majority do not (US export laws are one major reason). IPSec operates using public key cryptography, keys are exchanged and keyservers are also available (pluto, photuris, etc.). One major problem with IPSec is that the protocol has not been fully ratified as of yet, so implementations can vary. The main Linux IPSec effort (FreeS/WAN) is hosted at: http://www.xs4all.nl/~freeswan/ and since it is headed up by Canadians (my home and native land) it is fully exportable to anywhere in the world that allows the use of strong crypto (exceptions being France, China, Libya, etc). They just released version 1.0, and there is support for the 2.2.x series of kernels. The primary problems with FreeS/WAN currently (in my opinion) is the way tunnels and routing are handled is somewhat klunky, but they are working on it and making excellent progress.

PPTP (Point to Point Tunneling Protocol)

PPTP is a proprietary protocol created by Microsoft for VPN solutions. To date it has been shown to contain numerous and massive flaws. However if you need to integrate Linux into a PPTP environment all is not lost, http://www.moretonbay.com/vpn/pptp.html contains a Linux implementation of PPTP.

CIPE (Cyrpto IP Encapsulation)

CIPE is a free IP level encryption scheme, meant for use between routers. It is appropriate for 'bridging' networks securely together over insecure networks (like the Internet). The official cite for CIPE is at: http://sites.inka.de/~W1011/devel/cipe.html. I would however recommend FreeS/WAN as a better long term solution.

ECLiPt Secure Tunnel (currently in beta)

Another GNU licensed solution for Linux VPN's. Currently in beta (and not recommended for mass use) but I thought I should mention it anyways since it seems to be a serious effort. The official page is at: http://eclipt.uni-klu.ac.at/projects/est/. Again I would have to recommend FreeS/WAN as a better long term solution.

## Encrypting Data

Several encryption programs are also available to encrypt your data, some at the file level (PGP, GnuPG, etc.) and some at the drive level (Cryptographic File System for example). These systems are very appropriate for the storage of secure data, and to some degree for the transmission of secure data (although both ends will require the correct software, compatible versions, and an exchange of public keys will somehow have to take place) which is unfortunately an onerous task for most people. In addition to this you have no easy way of trusting someone's public key unless you receive it directly from them (such as at a key signing party), or unless it is signed by someone else you trust (but how do you get the trusted signer's key securely?). Systems for drive encryption such as CFS (Cryptographic FileSystem) are typically easy to implement, and only require the user to provide a password or key of some form to access their files.

PGP (Pretty Good Privacy)

The granddaddy of public encryption, this is by far one of the most popular programs as it is supported under Unix, Windows and Macintosh. Unfortunately it has now been commercialized, which has resulted in a loss of quality for users. I personally believe any software used to encrypt or otherwise secure data MUST be open source or how else can you be sure it is secure. PGP is now sold by Network Associates and I cannot in good faith recommend it as a security mechanism for the secure storage and transmission of files. PGP is available for download from ftp://ftp.replay.com/.

GnuPG (Gnu Privacy Guard)

The alternative to PGP, GnuPG (GPG) is a direct replacement that is fully opensource and GNU licensed (as if the name didn't give it away). This tool is available at: http://www.gnupg.org/, as source code or precompiled binaries for windows, and RPM's.

CFS (CryptoGraphic Filesystem)

CFS allows you to keep data on your harddrive in an encrypted format, and is significantly easier to use then a file encryption program (such as PGP) if you have many files and directories you want to keep away from curious people. The official distribution site is at: http://www.cryptography.org/, and RPM's are available at: ftp://ftp.replay.com/pub/replay/linux/redhat/, and Debian binaries are at: http://www.debian.org/Packages/stable/otherosfs/cfs.html.

### Sources of random data

In order for encryption to be effective, especially on a large scale such as IPSec across many hosts, good sources of random, cyrptographically secure data are needed. In Linux we have `/dev/random` and `/dev/urandom` which are good but not always great. Part of the equation is measuring 'random' events, manipulating that data and then making it available (via (u)random). These random events include: keyboard and mouse input, interrupts, drive reads, etc, however as hard drives ship with more and more cache (IBM Deskstars come with 4 megabytes cache on the drive), and more and more server have no keyboard/mouse being used the sources of data become harder to find, some sources like network activity are not entirely appropriate because the attacks may be able to measure it as well (granted this would be a very exotic attack, but enough to worry people nonetheless). There are several sources of random data that can be used (or at least they appear random), radioactive decay and radio frequency manipulations are two popular ones. Unfortunately the idea of sticking a radioactive device in a computer makes most people nervous. And using manipulated radio frequencies is prone to error, and the possibility of outside manipulation.

## Firewalling

Firewalling is the practice of filtering TCP-IP traffic, typically at the point where your network connects to another (i.e. the Internet, a customers LAN or other) network, that may be untrusted (in the case of the Internet) or perhaps even trusted (another floor of your building). Like firewalls in a large building, a network firewall can prevent and even block the spread of an attack.

Linux has had firewalling capacity for quite a while now in the form of ipfwadm, which was a very simplistic packet level filter, but quite effective and good enough for most people. With the advent of kernel 2.1+ this has been replaced with ipchains which is quite a bit more sophisticated. Both are still basic packet filters however and do not allow for advanced features such as stateful inspection or some types of proxying connections, however Linux does support IPMASQ, an advanced form of NAT (Network Address Translation). IPMASQ allows you to hook up a network of computers to the Internet, but proxy their connections at the packet level, thus all traffic appears to be coming and going to one machine (the Linux IPMASQ box), which affords a high degree of protection to the internal network. As an added bonus the clients on the internal network require NO proxy configuration, as long as the Linux IPMASQ server is configured correctly things will work quite well.

Both ipchains and ipfwadm provide the following basic capabilities:

- blocking / allowing data to pass through based on IP/port/interface source / destination
- masquerading of connections, based on IP/port/interface source / destination

In addition to which ipchains supports:

- port forwarding
- creation of chains, for more intricate rules and conditions, easier to maintain
- quality of service (QOS) routing, useful on low speed connections or otherwise saturated connections
- specification of IP/port/interface as well as inverse specification (using the !)

The Firewall-HOWTO and "`man <command>`" (ipchains or ipfwadm) page both cover in great detail the mechanics for setting up rules, but don't really cover the strategy for firewalling safely. Your first choice to make is whether to go with default deny or default allow policies, followed by which services and hosts you wish to allow and block.

When deciding policy you should choose a policy that will default to denying everything unless specifically allowed through (that is if there is a failure it will hopefully be minimized via default policies) or a policy that allows everything and blocks certain services/hosts. I typically use a policy of default denial as it can accommodate mistakes and changes more safely then a policy that defaults to allowing data through. Case in point, you have a server secured via firewalling, currently running apache, you install WU-FTPD on it for internal use (so people can upload files) at 3 am, you forget to change the firewall rules. If you have chosen a policy of default allowal anyone on the Internet can access the ftp server, and silly you, you installed an old version which allowed someone to compromise the machine. If on the other hand you go with a policy of default denial they would not have access to the ftp server, and neither would your users, but you would find out quite quickly.

I have decided to not cover specific firewalling rules in this section, for each network service I will provide examples, as to properly firewall a protocol you need to understand how it behaves. There is a huge difference between firewalling www and ftp for inbound and outbound access for example. Some general concepts/rules:

## IPFWADM

Ipfwadm is a solid packet filter for Linux, although it lacks a lot of features available in Ipchains. Ipfwadm only supports 3 targets for a packet, accept deny or reject, whereas ipchains rules can be targeted at 6 built in targets, or a user defined target. Ipfwadm is really only appropriate for a simple IP level firewall, ipmasquerading and if you plan to use FreeS/WAN (which currently does not support kernel 2.2.x). The basic options are: specify a direction (in out or both, useful with the interface flag), input rules, output rules, forwarding rules (say you have multiple interfaces, also covers the masquerading rules) and masquerade rules which control the behavior of masquerading (timeouts, etc). You can insert, append and delete rules, set default policies, and list all the rules, unlike ipchains you only have the 3 targets (ACCEPT, DENY, REJECT). Other then that it is very similar to ipchains, with some minor variations. The following is a script appropriate for a server bridging 2 networks (10.0.0.x on eth0, 10.0.0.1 and 192.168.0.x on eth1, 192.168.0.1) with a mail server running.

```
#!/bin/bash
#
# Flush all the rule sets first
#
ipfwadm -f -I
ipfwadm -f -O
ipfwadm -f -F
#
# Allow forwarding between the two networks and otherwise deny it for
security
#
ipfwadm -F -a accept -P all -S 10.0.0.0/24 -i eth0 -D 192.168.0.0/24
ipfwadm -F -a accept -P all -S 192.168.0.0/24 -i eth1 -D 10.0.0.0/24
ipfwadm -F -p deny
#
# And of course we have to allow those packets in
#
ipfwadm -I -a accept -P tcp -S 10.0.0.0/24 -i eth0 -D 192.168.0.0/24
ipfwadm -I -a accept -P tcp -S 192.168.0.0/24 -i eth1 -D 10.0.0.0/24
#
# Let them access the mail server port on the server but nothing else
#
ipfwadm -I -a accept -P tcp -S 10.0.0.0/24 -i eth0 -D 10.0.0.1 25
ipfwadm -I -a accept -P tcp -S 192.168.0.0/24 -i eth0 -D 192.168.0.1 25
ipfwadm -I -p deny
```

There is not time you should choose ipfwadm over ipchains, FreeS/WAN now supports the 2.2.x series of kernels.

## IPCHAINS

Several new things in IPCHAINS, you can create chains of rules (hence the name) and link them together, making administration of firewalls far easier. Ipchains supports more targets then ipfwadm, you can point a rule at: ACCEPT, DENY, REJECT, MASQ, REDIRECT, or RETURN or a user defined chain. As such it is very powerful, for example I could redirect all packets bound for port 80 (i.e. any www traffic) going through my gateway machine to be redirected to local port 3128, the squid proxy server. You can also use this in conjunction with quality of service routing, the example given in ipfwadm's documentation is that of prioritizing traffic going over a PPP link, you can give telnet traffic a much higher priority then say ftp, reducing latency problems caused by a saturated link. Typically I create an `/etc/rc.d/init.d/ipchains-sh`(or wherever appropriate) and call it immediately after the networking is brought up, this leaves a small time in which the server is vulnerable, but minimally so since no network daemons are running. The following script is appropriate for a gateway with 2 interfaces running, the reason I have used the DENY instead of REJECT target is so that the packet is dropped and not responded to in any way, this slows down network scans (as they wait for the packet to timeout instead of receiving a response) and gives away less information. I would also advise against logging data unless you have a significant amount of drive space available, for each packet I send (several bytes) many bytes of drive space is used up to create a log entry, making it easy to overwhelm syslog and/or your drive space on a fast connection. The ipchains homepage is at: http://www.rustcorp.com/linux/ipchains/.

```
#!/bin/bash
#
# This script sets up firewall rules appropriate for a server with 2
interfaces
# running as a gateway
# This script needs to be edited if you plan to use it.
# We assume the internal machines call all talk to the gateway, so no rules
block
# internal traffic
#
# A couple of variables
#
# ETH0 is the IP address on ETH0 (the external interface)
# ETH0NET is the network
# ETH0NETMASK is the network mask
# TRUSTEDHOST1 is a trusted host (for webmin/ssh)
# TRUSTEDHOST2 is a trusted host (for webmin/ssh)
# ETH1IP is the IP address on ETH1 (internal interface)
# ETH1NET is the network
# ETH1NETMASK is the network mask
#
ETH0IP=1.1.1.1
ETH0NET=1.1.1.0
ETH0NETMASK=24
TRUSTEDHOST1=1.5.1.1
TRUSTEDHOST2=1.5.1.2
ETH1IP=10.0.0.1
ETH1NET=10.0.0.0
ETH1NETMASK=24
#
PATH=/sbin
# FLUSH ALL RULES
```

```
ipchains -F input
ipchains -F output
ipchains -F forward
# ANTI-SPOOFING
ipchains -A input -p all -j DENY -s 10.0.0.0/8 -i eth0 -d 0.0.0.0/0
ipchains -A input -p all -j DENY -s 127.0.0.0/8 -i eth0 -d 0.0.0.0/0
ipchains -A input -p all -j DENY -s 192.168.0.0/16 -i eth0 -d 0.0.0.0/0
ipchains -A input -p all -j DENY -s 172.16.0.0/16 -i eth0 -d 0.0.0.0/0
ipchains -A input -p all -j DENY -s $ETH0IP -i eth0 -d 0.0.0.0/0
# ICMP FIRST
ipchains -A input -p icmp -j ACCEPT -s $ETH0NET/$ETH0NETMASK -i eth0 -d
0.0.0.0/0
ipchains -A input -p icmp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0
# SSH
ipchains -A input -p tcp -j ACCEPT -s $TRUSTEDHOST1 -i eth0 -d 0.0.0.0/0 22
ipchains -A input -p tcp -j ACCEPT -s $TRUSTEDHOST2 -i eth0 -d 0.0.0.0/0 22
# BLOCKING 1:1023
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 1:1023
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 1:1023
# BLOCKING OTHER THINGS
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 1109
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 1524
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 1600
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 2003
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 2049
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 2105
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 3001
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 3001
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0
3128:3130
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0
3128:3130
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 3306
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 3306
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 4444
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0
6000:6100
ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0
6000:6100
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 6667
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 7000
# WEBMIN
ipchains -A input -p tcp -j ACCEPT -s $TRUSTEDHOST1 -i eth0 -d 0.0.0.0/0
10000
ipchains -A input -p tcp -j ACCEPT -s $TRUSTEDHOST2 -i eth0 -d 0.0.0.0/0
10000
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -i eth0 -d 0.0.0.0/0 10000
# FORWARD RULES
ipchains -P forward DENY
ipchains -A forward -p all -j MASQ -s $ETH1NET/$ETH1NETMASK -d 0.0.0.0/0
```

## Rule Creation

ipfwadm2ipchains

A simple script that converts ipfwadm rules to ipchains rules, making migration a snap. The script is available at: http://users.dhp.com/~whisper/ipfwadm2ipchains/

mason

Mason is an automated firewall rule generator for ipfwadm and ipchains. You load it up and it monitors the packets flowing through the machine, then based on that creates a set of rules to allow that type of access. A good tool for first time firewall admins, available from: http://users.dhp.com/~whisper/mason/.

firewall.sh

A dialog based script that walks you through creation of firewall rules, nicely done and good for new users or admins with RSI, available from: http://devplanet.fastethernet.net/.

Mklinuxfw

Mklinuxfw is a perl tool that aims to provide a variety of interfaces (CGI, KDE, command line, etc.) to creation of firewall rules. It currently supports a CGI interface and GTK is in progress. You can download it from: http://www.madhouse.org.uk/~red/framepage.phtml?/mklinuxfw/index.html.

## Scanning / intrusion testing tools

Over the last few years the number of security tools for Windows and UNIX has risen dramatically, even more surprising is the fact that most of them are freely available on the Internet. I will only cover the free tools since most of the commercial tools are ridiculously expensive, are not open source, and in many cases have been shown to contain major security flaws (like storing passwords in clear text after installation). In any case any serious cracker/hacker will have these tools at their disposal, so why shouldn't you?

There are several main categories of tools, ones that scan hosts from within that host, ones that scan other hosts and report back variously what OS they are running (using a technique called TCP-IP fingerprinting), services that are available and so on, at the top of the food chain are the intrusion tools that actually attempt to execute exploits, and report back if they worked or not, lastly I include the exploits category, while not strictly an intrusion tool per se they do exist and you should be aware of them.

### Host scanners

Cops

Cops is extremely obsolete and it’s original home on CERT’s ftp site is gone.

Tiger

Tiger is obsolete but I thought I’d mention it for historical accuracy, Texas Agricultural and Mechanical University used to require that a UNIX host pass tiger before it was allowed to connect to the network from offsite. You can get it from: ftp://net.tamu.edu/pub/security/TAMU/.

SBScan

SBScan is an up and coming host based scanner, it looks for a variety of problems such as bad rhosts files, open ports, passwordless accounts, and scans the network for other types of naughtiness. SBScan is available from: http://sy.tsx.org/.

check.pl

check.pl is a nice perl program that checks file and directory permissions, and will tell you about any suspicious or ‘bad’ ones (setuid, setgid, writeable directories, etc). Very useful but it tends to find a lot of false positives. It’s available at: http://opop.nols.com/proggie.html.

### Network scanners

Strobe

Strobe is one of the older port scanning tools, quite simply it attempts to connect to various ports on a machine(s) and reports back the result (if any). It is simple to use and very fast, but doesn't have any of the features newer port scanners have. Strobe is available for almost all distributions as part of it, or as a contrib package, the source is available at: ftp://suburbia.net/pub/.

Nmap

Nmap is a newer and much more featured host scanning tool. It features advanced techniques such as TCP-IP fingerprinting, a method by which the returned TCP-IP packets are examined and the host OS is deduced based on various quirks present in all TCP-IP stacks. Nmap also supports a number of scanning methods, from normal TCP scans (simply trying to open a connection as normal) to stealth scanning and half open SYN scans (great for crashing unstable TCP-IP stacks). This is arguably one of the best port scanning programs available, commercial or otherwise. Nmap is available at: http://www.insecure.org/nmap/index.html.

MNS

http://www.thegrid.net/gravitino/products.html

Bronc Buster vs. Michael Jackson

http://www.thegrid.net/gravitino/products.html

Leet scanner

http://www.thegrid.net/gravitino/products.html

Soup scanner

http://www.thegrid.net/gravitino/products.html

Portscanner

Portscanner is a nice little portscanner (surprise!), that has varying levels of outputs making it easy to use in scripts, and by humans. It's opensource and free to use, you can get it at: http://www.ameth.org/~veilleux/portscan.html.

## Intrusion Scanners

Nessus

Nessus is relatively new, but is fast shaping up to be the best intrusion scanning tool. It has a client/server architecture, the server currently only runs on Linux, and clients are available for Linux, Windows and there is a Java client. Communication between the server and client is ciphered for added security, all in all a very slick piece of code. Nessus supports port scanning, and attacking, based on IP addresses or host name(s). It can also search through network DNS information and attack related hosts at your bequest. Nessus is relatively slow in attack mode, which is hardly surprising, but it currently has over 200 attacks, and a plug in language so you can write your own. Nessus is available from http://www.nessus.org/.

Saint

Saint is the sequel to Satan, a network security scanner made (in)famous by the media a few years ago (there were great worries that bad people would take over the Internet using it). Saint also uses a client/server architecture, but uses a www interface instead of a client program. Saint produces very easy to read and understand output, with security problems graded by priority (although not always correctly), and also supports add in scanning modules making it very flexible. Saint is available from: http://www.wwdsi.com/saint/.

Cheops

While not a scanner per se, it is useful for detecting a hosts OS and dealing with a large number of hosts quickly. Cheops is a "network neighborhood" on steroids, it builds a picture of a domain, or IP block, what hosts are running and so on. It is extremely useful for preparing an initial scan as you can locate interesting items (HP printers, Ascend routers, etc) quickly. Cheops is available at: http://www.marko.net/cheops/.

Ftpcheck / Relaycheck

Two simple utilities that scan for ftp servers and mail servers that allow relaying, good for keeping tabs on naughty users installing services they shouldn’t (or simply misconfiguring them), available from: http://david.weekly.org/code/.

SARA

Security Auditor’s Research Assistant (SARA) is a tool similar in function to SATAN and Saint. SARA supports multiple threads for faster scans, stores it’s data in a database for ease of access and generates nice HTML reports. SARA is free for use and is available from: http://home.arc.com/sara/.

### Firewall scanners

Firewalk

Firewalk is a program that uses a traceroute style of packets to scan a firewall and attempt to deduce the rules in place on that firewall. By sending out packets with various time to lives and seeing where they die or are refused a firewall can be tricked into revealing rules. There is no real defense against this apart from silently denying packets instead of sending a rejection message which hopefully will reveal less. I would advise utilizing this tool against your systems as the results can help you tighten up security. Firewalk is available from: http://www.packetfactory.net/firewalk/.

### Exploits

I won't cover exploits specifically, since there are hundreds if not thousands of them floating around for Linux. I will simply cover the main archival sites.

http://www.rootshell.com/

One of the primary archive sites for exploits, it has almost anything and everything, convenient search engine and generally complete exploits.

## Scanning and intrusion detection tools

If the last section has you worried you should be. There are however many defenses, active and passive against those types of attacks. The best ways to combat network scans are keep software up to date, only run what is needed, and heavily restrict the rest through the use of firewalls and other mechanisms. Luckily in Linux these tools are free and easily available, again I will only cover opensource tools, since the idea of a proprietary firewall/etc is rather worrying. The first line of defense should be a robust firewall, followed by packet filters on all Internet accessible machines, liberal use of TCP-WRAPPERS, logging and more importantly automated software to examine the logs for you (it is unfeasible for an administrator to read log files nowadays).

### Logging Tools

Port Sentry (beta)

The third component to the Abacus suite, it detects and logs port scans, including stealthy scans (basically anything nmap can do it should be able to detect). Port Sentry can be configured to block the offending machine (in my opinion a bad idea as it could be used for a denial of service attack on legitimate hosts), making completion of a port scan difficult. As this tool is in beta I would recommend against using it, however with some age it should mature into a solid and useful tool. Port Sentry is available at: http://www.psionic.com/abacus/portsentry/.

### Host based attack detection

Firewalling

Most firewalls support logging of data, and ipfwadm/ipchains are no exception, using the -l switch you get a syslog entry for each packet, using automated filters (Perl is good for this) you can detect trends/hostile attempts and so on. Since most firewalls (UNIX based, and Cisco in any case) log via the syslog facility, you can easily centralize all your firewall packet logging on a single host (with a lot of harddrive space hopefully).

TCP-WRAPPERS

Wietse's TCP-WRAPPERS allow you to restrict connections to various services based on IP address and so forth, but even more importantly it allows you to configure a response, you can have it email you, finger the offending machine, and so on (use with caution however). TCP_WRAPPERS comes standard with most distributions and is available at: ftp://ftp.porcupine.org/pub/security/.

Klaxon

While mostly obsoleted by TCP-WRAPPERS and firewall logging, klaxon can still be useful for detecting port scans/etc if you don't want to totally lock down the machine. Klaxon is available at: ftp://ftp.eng.auburn.edu/pub/doug/.

Host Sentry (pre release software)

While this software is not yet ready for mass consumption I thought I would mention it anyways as it is part of a larger project (the Abacus project, http://www.psionic.com/abacus/). Basically Host Sentry builds a profile of user accesses and then compares that to current activity in order to flag any suspicious activity. Host Sentry is available at: http://www.psionic.com/abacus/hostsentry/.

### Pikt

Pikt is an extremely interesting tool, it is actually more of a scripting language aimed at system administration then a simple program. Pikt allows you to do things such as killing off idle user processes, enforcing mail quotas, monitor the system for suspicious usage patterns (off hours, etc), and much more. About the only problem with Pikt will be a steep learning tools, as it uses it's own scripting language, but ultimately I think mastering this language will pay off if you have many systems to administer (especially since Pikt runs on Solaris, Linux and FreeBSD currently). Pikt is available at: http://pikt.uchicago.edu/pikt/.

## Network based attack detection

### NFR

NFR (Network Flight Recorder) is much more then a packet sniffer, it actually logs data and in real time detects attacks, scans and so on. This is a very powerful tool and requires a significant investment of time, energy and machine power to run, but it is at the top of the food chain for detection. NFR is available at: http://www.nfr.com/.

## **Host monitoring tools**

Monitoring your server(s) and host(s) is important for a variety of reasons, from tracking down break-ins to legal requirements. Remember, an ounce of prevention is worth a pound of cure. There are a variety of general monitoring tools available in the Logging section from syslog to auditd (which allows you to audit users opening files, running programs, etc). I strongly suggest you use them. As well there are a number of more specific programs to monitor system status and prevent users from doing things they shouldn't.

bgcheck

bgcheck runs in background and checks the process table for items that should be running (i.e. irc bots, password crackers, etc.). You can download it from: http://blue.dhs.org/bgcheck/.

Sxid

Sxid checks setuid and setgid for changes, generates MD5 signatures of the files and generally allows you to track any changes made. You can get it at: ftp://marcus.seva.net/pub/sxid/.

Viperdb

Viperdb checks setuid/setgid programs and folders and will notify you of any changes as well as restricting access to them if any changes are made, it's available for download from: http://www.resentment.org/projects/viperdb/.

Pikt

Pikt is covered in the previous section and can be used to monitor user activity. I would recommend it for large installations as it is extremely flexible and powerful. Pikt is available at: http://pikt.uchicago.edu/pikt/.

DTK

The Deception ToolKit is a set of programs that emulate well known services in order to provide a false set of readings to attackers. The hope is to confuse and slow down attackers by leading them to false conclusions, you can download DTK from: http://all.net/dtk/.

## Packet sniffers

Packet sniffing is the practice of capturing network data not destined for your machine, typically for the purpose of viewing confidential/sensitive traffic such as telnet sessions or people reading their email. Unfortunately there is no real way to detect a packet sniffer since it is a passive activity, however by utilizing network switches and fiber optic backbones (which are very difficult to tap) you can minimize the threat.

### tcpdump

The granddaddy of packet sniffers for Linux, this tool has existed as long as I can remember, and is of primary use for debugging network problems. It is not very configurable and lacks advanced features of newer packet sniffers, but it can be useful. Most distributions ships with tcpdump.

### sniffit

My favorite packet sniffer, sniffit is very robust, has nice filtering capabilities, will convert data payloads into ASCII text for easy reading (like telnet sessions), and even has a graphical mode (nice for monitoring overall activity/connections). Sniffit is available at: http://sniffit.rug.ac.be/sniffit/sniffit.html.

### Ethereal

A nice looking network protocol analyzer (a.k.a. a souped up sniffer), that has an interface very similar to NT’s network monitor, and always for easy viewing of data payloads for most network protocols (tftp, http, Netbios, etc). It is based on GTK, thus meaning you will probably have to be running gnome to use it. I haven'’ tested it yet (but intend to), and it is available at: http://ethereal.zing.org/.

### Other sniffers

There are a variety of packet sniffers for Linux, based on the libpcap library among others, here is a short list:

http://www.mtco.com/~whoop/ksniff/ksniff.html - KSniff
http://ksniffer.veracity.nu/ - Ksniffer
http://mojo.calyx.net/~btx/karpski.html - karpski
http://www.ozemail.com.au/~peterhawkins/gnusniff.html - Gnusniff
http://elektra.porto.ucp.pt/snmpsniff/ - SNMP Sniffer

## Virii, Trojan Horses, Worms, and Social Engineering

Linux is not susceptible to virii in the same ways that a Dos/Windows or Mac platform is. In UNIX security controls are a fundamental part of the operating system, things like not allowing users to write promiscuously to any location in memory that they choose to, something that Dos/Windows and the Mac allow. To be fair there are viruses for UNIX, however the only Linux one I have seen was called "bliss", had an uninstall option ("--uninstall-please") and had to be run as root to be effective. Or to quote an old Unix favorite "if you don't know what an executable does, don't run it as root". Worms are much more prevalent in the UNIX world, the first major occurrence being the Morris Internet worm which exploited a vulnerability in sendmail. Current worms for Linux exploit broken versions of imapd, sendmail, WU-FTPD and other daemons, the simplest fix is to keep up to date, and not make daemons accessible unless necessary. These attacks can be very successful especially if they find a network(s) that are not up to date, but typically their effectiveness fades out as people upgrade their daemons. In general I would not specifically worry about these two items, and there is definitely no need to buy anti virus software for Linux.

Social engineering on the other hand can be very effective, as no matter how many safe guards, security probes and patches you apply, humans can provide a wonderfully weak link to exploit. Case in point:

A customer bet me I couldn't crash his NT server (the bet was pizza and beer). The main problem in attacking his NT server was I had no idea what it's name or IP address was. Now I could have scanned the ISP's network using tools like ntinfosec, nbtstat and the like to look for likely netbios names, this would have taken several hours and annoyed the ISP's security officer (who I know quite well). The simplest solution was to ask the helpdesk. "Well you know I  shouldn't be telling you this" and I was given his IP address after claiming I needed to fix the server. One mangled ping packet after business hours and I got my free pizza and beer. "Hi this is Bob from the IT department, we need to reinitialize all the accounts so please change your password to "temporary"". You get the idea. This is by far one of the more difficult threats to protect against, the only real answer is user education, which doesn't work very well (case in point the Melissa macro virus, many users have been told not to open documents with macros yet they did).

Worms have a long and proud tradition in the UNIX world, by exploiting known security holes (generally, very few exploit new/unknown holes), and replicating they can quickly mangle a network(s). There are several worms currently making their way around Linux machines, mostly exploiting old Bind 4.x and old IMAP software, defeating them is as easy as keeping software up to date.

Trojan horses are also popular, recently ftp.win.tue.nl was broken into and the tcp_wrappers package (among others) was modified to email passwords to an anonymous account. This was detected when someone checked the PGP signature of the package and found that it wasn't quite kosher. Moral of the story? Use software from trusted sites, and check the PGP signature(s).

## Disinfection of virii / worms / trojans

Back up your data, format and reinstall the system from known good media. Once an attacker has root on a Linux system they can literally do anything, from compromising gcc/egcs to loading interesting kernel modules at boot time. Do not run untrusted software as root. Check the PGP signatures on files you download, etc. An ounce of prevention will pretty much block the spread of virii, worms and trojans under Linux.

The easiest method for dealing with virii and the like is to use system integrity tools such as tripwire, L5, and Gog&Magog, you will be able to easily find which files have been compromised and restore/replace/update them.

Virus Scanners

As stated above virii aren’t a real concern in the Linux world, however virus scanners that run on Linux can be useful. Filtering email / other forms of content at the gateways to your network can provide an extra line of strong defense since the platforms providing the defense against the threat cannot be compromised by that threat.

AMaViS

AMaViS uses third party scanning software (such as McAfee) to scan incoming email for virii. You can get AMaViS at: http://aachalon.de/AMaViS/.

## Password storage

This is something many people don’t think about much. How can you securely store passwords? The most obvious method is to memorize them, this however has it’s drawbacks, if you administer 30 different sites you generally want to have 30 different passwords, and a good password is 8+ characters in length and generally not the easiest thing to remember. This leads to many people using the same passwords on several systems (come on, admit it). One of the easiest methods is to write passwords down, this a BIG NO-NO, you’d be surprised what people find lying around, and what they find if they are looking for it. A better option is to store passwords in an encrypted format, usually electronically on your computer or palm pilot, this way you only have to remember one password to unlock the rest which you can then use. Something as simple as PGP or GnuPG can be used to accomplish this.

### Gpasman

Gpasman is an application that requires gtk (relatively easily to install on a non Gnome system, just load up the gtk libs), and encrypts your passwords using the rc2 algorithm. Upon startup of the program you type in your master password, and assuming it is correct) you are presented with a nice list of your user accounts, sites, passwords and a comment field. Gpasman is available at: http://www.student.wau.nl/~olivier/gpasman/.

## Conducting baselines / system integrity

One major oversight made by a lot of people when securing their machines is that they forget to create a baseline of the system, that is a profile of the system, it's usage of resources, and so on in normal operation. For example something as simple as a "`netstat -a -n > netstat-output`" can give you a reference to latter check against and see if any ports are open that should not be. Memory usage and disk usage are also good things to keep an eye on, a sudden surge in memory usage could result in the system being starved of resources, likewise for disk usage, it might be a user accident, a malicious user, or a worm program that has compromised your system and is now scanning other systems. Various tools exist to measure memory and disk usage: `vmstat`, `free`, `df`, `du`, all of which are covered by their respective man pages.

At the very minimum make a full system backup, and regularly backup config files and log files, this can also help you pinpoint when an intrusion occurred (user account "rewt" was added after the April 4th backup, but isn't in the March 20th backup). Once a system is compromised typically a "rootkit" is installed, these consist of trojaned binaries, and are near impossible to remove safely, you are better of formatting the disk and starting from scratch. There is of course a notable exception to this rule, if you were diligent and used file/directory integrity tools such as L5 you will be able to pinpoint the affected files easily and deal with them.

Tripwire

Tripwire is no longer a open source tool, I have absolutely NO problems with commercial software, but when you expect me to rely on a program to provide security, when I (nor anyone else really) can easily view the source (it is available under some special license agreement, probably an NDA) I must decline. Tripwire costs $70 approximately for Linux, and is only available as an RPM package aimed at (tripwire is $500 for other operating systems), which is rather on the high side for a piece of software that can easily be replaced with alternatives such as L5. Tripwire is available at: http://www.tripwiresecurity.com/.

L5

There is an alternative to tripwire however, L5, available at: ftp://avian.org/src/hacks/, it is completely free and very effective. I would definitely recommend this tool.

Gog&Magog

Gog&Magog creates a list of system file properties, owner, permissions, an MD5 signature of the file and so (similar to tripwire). You can then have it automatically compare this and ensure any changed files/etc come to your attention quickly. As well it makes recovering from a break in simpler as you'll know which files were compromised. You can download Gog&Magog from: http://www.multimania.com/cparisel/gog/.

confcollect

confcollect is a simple script that collects system information such as routing tables, rpm's installed and the like. You can download it from: http://www.skagelund.com/confcollect/

## Backups

Something people forget about, but you can compare the current files to old backups, many backup formats (Tape, floppy, CDR, etc.) can be made read only, so a backup of a newly installed system provides a good benchmark to compare things to. The utility “diff” and “cmp” can be used to compare files against each other. See the backup session for a full listing of free and commercial software.

## Conducting audits

So you've secured your machines, and done all the things that needed to be done. So how do you make sure it's actually doing what it is supposed to do, or prove to someone that it is as secure as you say it is? Well you conduct an audit. This can be as simple as reviewing the installed software, configuration files and other settings, or as complex as putting together or hiring a tiger team (or ethical hackers, or whatever buzzword(s) you prefer) to actively try and penetrate your security. If they can't then you did your job well (or they suck), and if they do get in, you know what needs to be fixed (this is also a good method to get an increased security budget, show how vulnerable you are to the CIO).

There are also many free tools and techniques you can use to conduct a self audit and ensure that the systems react as you think they should (we all make errors, but catching them quickly and correcting them is part of what makes a great administrator). Tools such as nmap, nessus, crack, and so forth can be quickly employed to scan your network(s) and host(s), finding any obvious problems quickly. I also suggest you go over your config files every once in a while (for me I try to 'visit' each server once a month, sometimes I discover a small mistake, or something I forgot to set previously). Keeping systems in a relative state of synchronization (I just recently finished moving ALL my customers to Kernel 2.2.x, ipchains) which will save you a great deal of time and energy.

Using the tools mentioned earlier in “ Conducting baselines” you can check file integrity using tripwire, L5, backups or other methods. Another tool that is useful for check binaries is the “strings” commands, it shows readable information in binary files, and is especially useful if someone forgot to run “strip” on their binaries after compiling them (people have gotten lucky and gotten the directory from which the exploit was compiled, allowing them to trace down the exact user).

## Backups

I don't know how many times I can tell people, but it never ceases to amaze me how often people are surprised by the fact that if they do not backup their data, if the drive craters out on them, or they hit 'delete' without thinking it will be gone. Always backup your system, even if it's just the config files, you'll save yourself time and money in the long run.
To backup your data under Linux there are many solutions, all with various pro's and con's. There are also several industrial strength backup programs, the better ones support network backups which are a definite plus in a large non-homogenous environment .

### Tar and Gzip

Oldies but still goldies, tar and gzip. Why? Because like vi you can darn near bet the farm on the fact that any UNIX system will have tar and gzip. They may be slow, klunky and starting to show their age, but it's a universal tool that will get the job done. I find with Linux the installation of a typical system takes 15-30 minutes depending on the speed of the network/cdrom, configuration another 5-15 (assuming I have backups or it is very simple) and data restoration takes as long as it takes (definitely not something you should rush). Good example: I recently backed up a server and then proceeded to blow the filesystem away (and remove 2 physical HD's that I no longer needed), I then installed RedHat 5.2, and reconfigured all 3 network cards, apache (for about 10 virtual sites), Bind and several other services in about 15 minutes. If I had done it from scratch it would have taken me several hours. Simply:

```
tar -cvf archive-name.tar dir1 dir2 dir3....
```

to create the tarball of all your favorite files (typically /etc, /var/spool/mail/, /var/log/, /home, and any other user/system data), followed by a:

```
gzip -9 archive-name.tar
```

to compress it as much as possible (granted harddrive space is cheaper then a politicians promise but compressing it makes it easier to move around). You might want to use bzip, which is quite a bit better then gzip at compressing text, but it is quite a bit slower. I typically then make a copy of the archive on a remote server, either by ftping it or emailing it as an attachment if it's not to big (i.e. the backup of a typical firewall is around 100k or so of config files).

### Non Commercial Backup Programs for Linux

Amanda

Amanda is a client/server based network backup programs with support for most unices and Windows (via SAMBA). Amanda is BSD style licensed and available from: http://www.amanda.org/.

afbackup

Afbackup is another client/server with a generally GPL license with one minor exception, development of the server portion on Windows is forbidden. Afbackup has server support for Linux, HP-UX and Solaris, and has clients for that and windows. You can download it at: ftp://ftp.zn-gmbh.com/pub/linux/.

Burt

Burt is a Tcl/Tk based set of extensions that allow for easy backups of Unix workstations, this allows it to run on pretty much any system. Burt is a client/server architecture and appears pretty scalable, it is available at: http://www.cs.wisc.edu/~jmelski/burt/.

## Commercial Backup Programs for Linux

BRU

RedHat ships with a nice backup program (well a demo version anyways) called BRU (**B**ackup and **R**estore **U**tility), this thing has been in the Linux world since as long as Linux Journal (they have had ads in there since the beginning). This program affords a relatively complete set of tools in a nice unified format, with command line and a graphical front end (easy to automate in other words). It supports full, incremental and differential backups, as well as catalogs, and can write to a file or tape drive, basically a solid, simple, easy to use backup program, and it ships with RedHat Linux (one license), so if you bought RedHat you should have a copy to play with. BRU is available at http://www.estinc.com/features.html.

Quickstart

Quickstart is more aimed at making an image of the system so that when the hard drive fails/etc. you can quickly re-image a blank disk and have a working system. It can also be used to 'master' a system and then load other systems quickly (as an alternative to say RedHat's kickstart). It's reasonably priced as well and garnered a good revue in Linux Journal (Nov 1998, page 50). You can get it at: http://www.estinc.com/qsdr.html.

Backup Professional

http://www.unitrends.com/bp.html

CTAR

http://www.unitrends.com/ctar.html

CTAR:NET

http://www.unitrends.com/ctarnet.html

PC ParaChute

http://www.unitrends.com/pcpara.html

Arkeia

Arkeia is a very powerful backup program with a client - server architecture that supports many platforms. This is an 'industrial' strength product and appropriate for heterogeneous environments, it was reviewed in Linux Journal (April 1999, page 38) and you can download a shareware version online and give it a try, the URL is: http://www.arkeia.com/.

Legato Networker

Legato Networker is another enterprise class backup program, with freely available (but unsupported) Linux clients. Legato Networker is available at: http://www.legato.com/Products/html/legato_networker.html and the Linux clients are available from: http://feral.com/networker.html.

## Pro's and Con's of Backup Media

There are more things to back data up onto then you can drive a range rover over but here are some of the more popular/sane alternatives:

| Name of Media | Pro's | Cons |
|---|---|---|
| Hard Drive | It's fast. It's cheap. It's pretty reliable. ($20-$30 USD per gig) | It might not be big enough, and they do fail, usually at the worst possible time. Harder to take offsite as well. RAID is a viable option though. 20 gig drives are $350 USD now. |
| CDROM | Not susceptible to EMP, and everyone in the developed world has a CDROM drive. Media is also pretty sturdy and cheap ($2 USD per 650 Megs or so) | CDROM's do have a finite shelf life of 5-15 years, and not all recordables are equal. Keep away from sunlight, and make sure you have a CDROM drive that will read them. |
| Tape | It's reliable, you can buy BIG tapes, tape carousels and tape robots, and they're getting cheap enough for almost everyone to own one. | Magnetic media, finite life span and some tapes can be easily damaged (you get what you pay for), also make sure the tapes can be read on other tape drives (in case the server burns down....). |
| Floppies | I'm not kidding, there are rumors some people still use these to backup data. | It's a floppy. Whaddya think? |
| Zip Drives | I have yet to damage one, nor have my cats. They hold 100 megs which is good enough for most single user machines. | Not everyone has a zip drive, and they are magnetic media. |
| Jazz Drives | 1 or 2 gig removable harddrives, my SCSI one averages 5 meg/sec writes. | They die. I'm on my third drive. The platters also have a habit of going south if used heavily. |
| Syquest | 1.6 gigs, sealed platter, same as above. | Sealed cartridges are more reliable. Company did recently declare bankruptcy though. No warranty service. |
| LS120 | 120 Megs, and cheap, gaining in popularity. | Slooooooooow. I'm not kidding. 120 megs over a floppy controller to something that is advertised as "up to 3-4 times faster then a floppy drive". |

| | | |
|---|---|---|
| Printer | Very long shelf life. requires a standard Mark 1 human being as a reading device. Handy for showing consultants and as reference material. Cannot be easily altered. | You wanna retype a 4000 entry password file? OCR is another option as well. |

## Dealing with attacks

Dealing with an attack depends on several factors, is the attack in progress? Did you discover your company plan being sent out by the mail server to a hotmail address? Did you get called in to find a cluster of dead servers? What are your priorities? Restoring service? Ensuring confidential data is safe? Prosecuting the attacker(s)? Several things to keep in mind:

- Most admins will not respond very positively to news their site is being used as a base of attack, i.e. it's a waste of time usually.
- Most sites usually don't want to report attacks (successful or not) due to the potential embarrassment and related public relations problems.
- Most quick attacks, denial of service attacks and the like are spoofed. Tracking down the real attacker is very difficult and resource intensive.
- Even if all goes well there is a chance law enforcement will seize your equipment as evidence, and hold it, not something to be taken lightly.
- Do you know how the attacker got in (i.e. NFR recorded it), if so you might just want to plug the holes and go on.

Also before you deal with an attack, you should consult your company policy, if you don't have one consult your manager, the legal department, etc. It's also a good idea to have a game plan to deal with attacks (i.e. the mail server is first priority, checking fileservers is number two, who to notify, etc) will prevent a lot of problems when it happens (be prepared). The book “Practical Unix and Internet Security” covers this topic in great detail so I’m not going to rehash it, go buy the book.

An excellent whitepaper on this is also available, see Appendix D, “How to Handle and Identify Network Probes”.

## Denial of service attacks

One mess I didn’t want to venture into but must it seems. Denial of service (DOS) attacks are by far the most annoying and troublesome attacks, since in general they cannot be blocked, no matter how fast you flush your SYN connection tables, or limit CPU time for users, enough attacks fast enough will cause grief. Generally speaking there are two types of DOS attacks, local and remote, local are the easiest to deal with since once you figure out which user account is responsible you may terminate it/etc. Remote DOS attacks are typically spoofed from a wide variety of addresses, making firewalling ineffective (or they simply spoof as from a location you actually need to talk to, such as a customer’s site).

## **Examples of attacks**

Without going into to much detail and helping the black hats I want to give a few examples of attacks, to show how innocuous looking things can be problematic and other complicate your life.

Ping flooding (a.k.a. smurfing)

Simply flooding a network with data is an old fashioned but effective tactic, made worse by the fact most networks have faulty firewall configurations. By pinging the network address of a remote network (say a cablemodem ISP) you can receive several hundred ping replies for each ping packet you send. Now if you spoof the IP address and label the outgoing packets as from a network you do not like you can have someone else's improperly setup network do the dirty work and flood the victim.

DNS cache poisoning

Since so many services rely on DNS to work properly it provides a wonderful part of the network to attack. Subverting the information in DNS servers is easier then it should be, and you can insert false data if successful. For example if I convinced your name server that updates.redhat.com actually pointed at updates.badpeople.com, I could probably trick you into downloading and installing my software. This of course is negated by the fact RedHat PGP signs their packages, but do you check those signatures? As well, if you use an automated tool such as autorpm it will happen without user intervention, the compromised packages are downloaded and installed, all I have to do is watch my ftp log and then exploit the sites that download packages. If I managed to convince your mail server that othercompany.com was actually one of my servers I could not only receive email you send to othercompany.com, I could read the email, and perhaps send it along with minor modifications (like add an extra 0 to the cost of your bid).

## Distribution specific tools

### SuSE

One of SuSE's employees (Marc Heuse) has written a few useful utilities for SuSE Linux, available at: http://www.suse.de/~marc/. The first one is called "Harden SuSE" and basically goes about removing sharp objects, tightening up file permissions, turning off daemons and so on. The second one "SuSE security check" is a set of shell scripts that check the password file for sanity, lists out all installed packages once a month and so on.

## Distribution specific errata and security lists

### RedHat

Errata
http://www.redhat.com/support/docs/errata.html
Security
http://www.redhat.com/support/docs/errata.html
Mailing lists
http://archive.redhat.com/

### Debian

Errata
http://www.debian.org/distrib/packages/
Security
http://www.debian.org/security/
Mailing lists
http://www.debian.org/MailingLists/subscribe/

### Slackware

Errata
ftp://ftp.cdrom.com/pub/linux/slackware-current/ChangeLog.txt
Security
ftp://ftp.cdrom.com/pub/linux/slackware-current/ChangeLog.txt
Mailing Lists
NO URL

### Caldera

Errata
http://www.calderasystems.com/support/download.html
Security
http://www.calderasystems.com/news/security/index.html
Mailing Lists
http://www.calderasystems.com/support/forums.html

### SuSE

Errata
http://www.suse.de/e/patches/
Security
http://www.suse.de/security/
Mailing Lists
http://www.suse.com/Mailinglists/index.html

### TurboLinux

Errata
http://www.turbolinux.com/support/solutions.html
Security
http://www.turbolinux.com/support/solutions.html
Mailing Lists
NO URL

### Stampede GNU/Linux

Errata
ftp://ftp.stampede.org/current/README.CHANGES
Security
ftp://ftp.stampede.org/current/README.CHANGES
Mailing Lists
http://www.stampede.org/maillists.php3

### Mandrake

Errata
http://www.linux-mandrake.com/en/fupdates.html
Security
http://www.linux-mandrake.com/en/fupdates.html
Mailing Lists
http://www.linux-mandrake.com/en/flists.html

### LinuxPPC

Errata
NO URL
Security
NO URL
Mailing Lists
http://lists.linuxppc.org/

### Linux Pro

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

**LinuxWare**

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

**MKLinux**

Errata
NO URL
Security
NO URL
Mailing Lists
http://www.mklinux.org/mailinglists.html

**Yggdrasil**

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

**Connectiva**

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

### DLD

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

### Eagle Linux M68K

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

### Eurielec

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

### Kheops Linux

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

### MNIS Linux

Errata
NO URL
Security
NO URL
Mailing Lists
NO URL

## Internet connection checklist

The following is a checklist for people attaching as computer to the Internet (PPP, *DSL, Cablemodem, etc). This is by no means comprehensive, but it should help.

Turn off all unneeded daemons and network software – disable telnet, ftp, ntalk, auth, pop, imap, etc. unless you plan to use it. If it's turned off it's much less of a risk.

Use ipfwadm/ipchains to firewall services if possible, if you firewall by default this will also greatly slow down any scans people execute on your machines. Generally speaking services such as NFS / Samba, imap and pop will only need to be accessible to internal users, blocking external access greatly simplifies matters.

Use TCP_WRAPPERS to secure the services you leave on, where possible restrict access to internal clients or certain Internet clients for services such as imap, pop, and ftp. Also Remember that most distributions ship with nfs configured to use TCP_WRAPPERS, and SSH can use TCP_WRAPPERS as well. This allows you to easily centralize access control to services.

Update all software (especially network software) to the latest versions, check the errata / security pages that apply to your distribution.

Use a system integrity tool such as L5 of Gog&Magog to establish a list and signatures of your current (known good) files. If you are broken into later this will make your life easier.

Run penetration tests from an external site, i.e. run nmap / strobe, nessus and similar tools against your machine to ensure it is properly locked up. Remember, the bad guys have these tools, so you should use them to.

Keep software / access lists / etc. up to date. Audit your log files using tools like logcheck on a regular basis.

Use tools like Pikt if possible to monitor and prevent system problems (i.e. users with huge mail boxes).

If you need a truly detailed audit log install auditd and enable kernel auditing for events such as the opening of files and running of programs. Make sure you have plenty of disk space.

## Appendix A: Books and magazines

Sendmail - http://www.oreilly.com/catalog/sendmail2/
Linux Network Admin Guide (NAG) - http://www.oreilly.com/catalog/linag/
Running Linux - http://www.oreilly.com/catalog/runux2/noframes.html
DNS & BIND - http://www.oreilly.com/catalog/dns3/
Apache - http://www.oreilly.com/catalog/apache2/
Learning The Bash Shell - http://www.oreilly.com/catalog/bash2/
Building Internet Firewalls - http://www.oreilly.com/catalog/fire/
Computer Crime - http://www.oreilly.com/catalog/crime/
Computer Security Basics - http://www.oreilly.com/catalog/csb/
Cracking DES - http://www.oreilly.com/catalog/crackdes/
Essential System Administration - http://www.oreilly.com/catalog/esa2/
Linux in a nutshell - http://www.oreilly.com/catalog/linuxnut2/
Managing NFS and NIS - http://www.oreilly.com/catalog/nfs/
Managing Usenet - http://www.oreilly.com/catalog/musenet/
PGP - http://www.oreilly.com/catalog/pgp/
Practical Unix and Internet Security - http://www.oreilly.com/catalog/puis/
Running Linux - http://www.oreilly.com/catalog/runux2/
Using and Managing PPP - http://www.oreilly.com/catalog/umppp/
Virtual Private Networks - http://www.oreilly.com/catalog/vpn2/

RedHat/SAMS also publish several interesting books:
Maximum RPM (available as a postscript document on www.rpm.org)
RedHat User's Guide (available as HTML on ftp.redhat.com)

SNMP, SNMPv2 and RMON - W. Stallings (ISBN: 0-201-63479-1)

Magazines:
Linux Journal (of course, monthly)
Sys Admin (intelligent articles, monthly)
Perl Journal (quarterly)
Information Security - http://www.infosecuritymag.com/

## Appendix B: URL listing for programs

### Table of contents

### License

### Preface

### Forward by the author

### Contributing

### What this guide is and isn't

### How to determine what to secure and how to secure it

### Safe installation of Linux

### General concepts, server verses workstations, etc

### Physical / Boot security

### The Linux kernel

ftp://ftp.kernel.org/ - Linux kernel source code

### Administrative tools

ftp://ftp.replay.com/ - SSH
http://www.net.lut.ac.uk/psst/ - LSH
ftp://ftp.replay.com/ - SSL Telnet
http://violet.ibs.com.au/slush/ - Slush
http://www.networkshell.com/ - NSH
http://www.lysator.liu.se/fsh/ - Fsh
http://www.leenux.com/scripts/ - secsh
http://www.courtesan.com/sudo/ - sudo
ftp://ftp.ucolick.org/pub/users/will/ - super
http://www.webmin.com/ - Webmin
http://www.solucorp.qc.ca/linuxconf/ - Linuxconf
http://www.coas.org/ COAS

### System Files

### Log files and other forms of monitoring

http://www.core-sdi.com/ssyslog/ - secure syslog
http://www.balabit.hu/products/syslog-ng.html - next generation syslog
http://www.psionic.com/abacus/logcheck/ - logcheck
http://www.resentment.org/projects/colorlogs/ - colorlogs
http://www.vcpc.univie.ac.at/~tc/tools/ - WOTS
ftp://ftp.stanford.edu/general/security-tools/swatch/ - swatch
ftp://ftp.hert.org/pub/linux/auditd/ - auditd

**Shadow passwords**
http://www.false.com/security/john/ - Jack the ripper
http://www.users.dircon.co.uk/~crypto/ - Crack
http://www.thegrid.net/gravitino/products.html - Saltine cracker
http://wilter.com/wf/vcu/ - VCU

**PAM**
http://www.sun.com/software/solaris/pam/ - PAM

**Software Management**
http://www.rpm.org/ - RPM
http://www.gnu.ai.mit.edu/software/stow/stow.html - stow
ftp://ftp.replay.com/ - PGP
ftp://ftp.kaybee.org/pub/linux/ - AutoRPM
ftp://missinglink.darkorb.net/pub/rhlupdate/ - rhlupdate
http://www.iaehv.nl/users/grimaldo/info/scripts/ - RpmWatch
http://www.debian.org/Packages/stable/admin/apt.html - apt
http://datanord.datanord.it/~pdemauro/installwatch/ - installwatch
http://hal.csd.auth.gr/~vvas/instmon/ - instmon
http://kitenet.net/programs/alien/ - alien

**File / Filesystem security**
http://gsu.linux.org.tr/wipe/ - wipe (durakb@crit2.univ-montp2.fr)
http://users.erols.com/thomassr/zero/download/wipe/ - wipe (thomassr@erols.com)

**TCP-IP and network security**
http://www.bieringer.de/linux/IPv6/ - Ipv6
http://www.cri.cz/kra/index.html – HUNT Project

**PPP security**
http://www.interweft.com.au/other/ppp-howto/ppp-howto.html - PPP HOWTO

**Basic network service security**
ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/ - lsof

**Basic network services config files**

**Network services**
http://www.massconfusion.com/ssh/ - Fresh Free FiSSH
http://hp.vector.co.jp/authors/VA002416/teraterm.html - TeraTerm
http://www.zip.com.au/~roca/ttssh.html - TeraTerm SSH DLLs
ftp://rak.isternet.sk/mnt/rhcd/misc/putty/ - putty
http://www.mindbright.se/mindterm/ - mindterm
http://www.webmin.com/webmin/ - Webmin
http://www.apacheweek.com/features/userauth/ - Apache User Authentication
http://squid.nlanr.net/Squid/FAQ/FAQ.html - SQUID FAQ
http://www.sendmail.org/ - Sendmail
http://www.qmail.org/ - Qmail
http://www.postfix.org/ - Postfix

http://www.zmailer.org/ - Zmailer
http://netwinsite.com/dmail_first.htm – DMail
http://www.washington.edu/imap/ - WU IMAPD (stock popd)
http://andrew2.andrew.cmu.edu/cyrus/imapd/ - Cyrus
http://www.nodomainname.net/software/ids-pop/ - IDS POP
http://www.washington.edu/imap/ - WU IMAPD (stock imapd)
http://andrew2.andrew.cmu.edu/cyrus/imapd/ - Cyrus
http://www.horde.org/imp/ - IMP
http://www.nodomainname.net/software/atdot/ - AtDot
http://netwinsite.com/dmailweb/index.htm - DmailWeb
http://netwinsite.com/webimap/index.htm - WebImap
http://www.isc.org/bind.html - BIND
http://www.dents.org/ - Dents
http://www.isc.org/inn.html - INN
http://netwinsite.com/dnews.htm - DNews
http://www.isc.org/dhcp.html – DHCPD
http://www.nrw.net/uwe/utftpd.html - utftpd
http://ftp.bitgate.com/cfingerd/ - Configurable Finger Daemon
http://insecurity.net/ - Paul's secure identd written in perl
http://www.ojnk.nu/~odin/ - ojnk identd
http://www.tildeslash.org/nullidentd.html - null identd
http://www.ajk.tele.fi/~too/sw/ - fake identd
http://p8ur.op.het.net/midentd/ - midentd
http://www.eecis.udel.edu/~ntp/ NTP
http://cuba.xs4all.nl/~tim/scvs/ - Secure CVS documentation
http://cblack.mokey.com/cvsd/ - CVSD
http://rsync.samba.org/ - Rsync
http://www.astart.com/lprng/LPRng.html - LP Next Generation
http://feynman.tam.uiuc.edu/pdq/ - pdq
http://csociety.ecn.purdue.edu/~sigos/projects/ssh/forwarding/ - Forwarding X via ssh
http://www.samba.org/ - SAMBA
http://anu.samba.org/cgi-bin/swat/ - SWAT Demo

**File sharing methods**
http://www.coda.cs.cmu.edu/ - CODA
http://www.edlund.org/projects/drall/index.html - Drall
http://www.angelfire.com/hi/plutonic/afs-faq.html – AFS

**Network based authentication**
http://metalab.unc.edu/LDP/HOWTO/NIS-HOWTO.html - NIS / NIS+ Howto
http://srp.stanford.edu/srp/ - SRP
http://web.mit.edu/kerberos/www/ - Kerberos

**Encrypting services / data**
http://www.openssl.org/ - OpenSSL
http://www.apache-ssl.org/ - Apache SSL
ftp://ftp.uni-mainz.de/pub/internet/security/ssl/ - SSL Telnet
ftp://ftp.replay.com/pub/replay/linux/redhat/ - SSL Telnet RPMs
ftp://ftp.uni-mainz.de/pub/internet/security/ssl/ - SSL FTP
ftp://ftp.replay.com/pub/replay/linux/redhat/ - SSL FTP RPMs

http://www.xs4all.nl/~freeswan/ - FreeS/WAN, Linux IPSec
http://www.moretonbay.com/vpn/pptp.htm - Linux PPTP
http://sites.inka.de/~W1011/devel/cipe.html - CIPE
http://eclipt.uni-klu.ac.at/projects/est/ - ECLiPt
ftp://ftp.replay.com/ - PGP
http://www.gnupg.org/ - GnuPG
http://www.cryptography.org/ - CFS
ftp://ftp.replay.com/pub/replay/linux/redhat/ - CFS RPMs
http://www.debian.org/Packages/stable/otherosfs/cfs.html - CFS debs

**Firewalling**
http://www.rustcorp.com/linux/ipchains/ - IPCHAINS
http://users.dhp.com/~whisper/ipfwadm2ipchains/ - IPFWADM 2 IPCHAINS
http://users.dhp.com/~whisper/mason/ - Mason
http://devplanet.fastethernet.net/ - firewall.sh
http://www.madhouse.org.uk/~red/framepage.phtml?/mklinuxfw/index.html - Mklinuxfw

**Scanning / intrusion testing tools**
ftp://net.tamu.edu/pub/security/TAMU/ - Tiger
http://sy.tsx.org/ - SBScan
http://opop.nols.com/proggie.html - check.pl
ftp://suburbia.net/pub/ - Strobe
http://www.insecure.org/nmap/index.html - Nmap
http://www.thegrid.net/gravitino/products.html - MNS
http://www.thegrid.net/gravitino/products.html - Bronc Buster vs. Michael Jackson
http://www.thegrid.net/gravitino/products.html - Leet scanner
http://www.thegrid.net/gravitino/products.html - Soup scanner
http://www.ameth.org/~veilleux/portscan.html - Portscanner
http://www.nessus.org/ - Nessus
http://www.wwdsi.com/saint/ - Saint
http://www.marko.net/cheops/ - Cheops
http://david.weekly.org/code/ - Ftpcheck & Relaycheck
http://home.arc.com/sara/ - SARA
http://www.packetfactory.net/firewalk/ - Firewalk
http://www.rootshell.com/ - Exploit code

**Scanning and intrusion detection tools**
http://www.psionic.com/abacus/portsentry/ - Port Sentry
ftp://ftp.porcupine.org/pub/security/ - TCP_WRAPPERS
ftp://ftp.eng.auburn.edu/pub/doug/ - Klaxon
http://www.psionic.com/abacus/hostsentry/ - Host Sentry
http://pikt.uchicago.edu/pikt/ - Pikt
http://www.nfr.com/ - NFR

**Host monitoring tools**
http://blue.dhs.org/bgcheck/ - bgcheck
ftp://marcus.seva.net/pub/sxid/ - Sxid
http://www.resentment.org/projects/viperdb/ - Viperdb
http://pikt.uchicago.edu/pikt/ - Pikt
http://all.net/dtk/ - DTK

**Packet sniffers**
http://sniffit.rug.ac.be/sniffit/sniffit.html - sniffit
http://ethereal.zing.org/ - Ethereal
http://www.mtco.com/~whoop/ksniff/ksniff.html - KSniff
http://ksniffer.veracity.nu/ - Ksniffer
http://mojo.calyx.net/~btx/karpski.html - karpski
http://www.ozemail.com.au/~peterhawkins/gnusniff.html - Gnusniff
http://elektra.porto.ucp.pt/snmpsniff/ - SNMP Sniffer

**Virii, Trojan Horses, Worms, and Social Engineering**
http://aachalon.de/AMaViS/ - AMaViS

**Password storage**
http://www.student.wau.nl/~olivier/gpasman/ - Gpasman

**Conducting baselines / system integrity**
http://www.tripwiresecurity.com/ - Tripwire
ftp://avian.org/src/hacks/ - L5
http://www.multimania.com/cparisel/gog/ - Gog&Magog
http://www.skagelund.com/confcollect/ - confcollect

**Conducting audits**

**Backups**
http://www.amnda.org/ - Amanda
ftp://ftp.zn-gmbh.com/pub/linux/ - afbackup
http://www.cs.wisc.edu/~jmelski/burt/ - Burt
http://www.estinc.com/features.html - BRU
http://www.estinc.com/qsdr.html - Quickstart
http://www.unitrends.com/bp.html - Backup Professional
http://www.unitrends.com/ctar.html - CTAR
http://www.unitrends.com/ctarnet.html - CTAR:NET
http://www.unitrends.com/pcpara.html - PC ParaChute
http://www.arkeia.com/ - Arkeia
http://www.legato.com/Products/html/legato_networker.html - Legato Networker Linux client
http://feral.com/networker.html - Legato Networker server

**Dealing with attacks**

**Denial of service attacks**

**Examples of attacks**

**Distribution specific tools**
http://www.suse.de/~marc/ - Harden SuSE
http://www.suse.de/~marc/ - SuSE Security Check

**Distribution specific errata and security lists**
URLs listed in section clearly.

## Appendix C: Other Linux security documentation

Firewalling and Proxy Server HOWTO
http://metalab.unc.edu/LDP/HOWTO/Firewall-HOWTO.html

Linux IPCHAINS HOWTO
http://metalab.unc.edu/LDP/HOWTO/IPCHAINS-HOWTO.html

Linux Security HOWTO
http://metalab.unc.edu/LDP/HOWTO/Security-HOWTO.html

Linux Shadow Password HOWTO
http://metalab.unc.edu/LDP/HOWTO/Shadow-Password-HOWTO.html

The Linux CIPE + Masquerading mini-HOWTO
http://metalab.unc.edu/LDP/HOWTO/mini/Cipe+Masq.html

Firewall Piercing mini-HOWTO
http://metalab.unc.edu/LDP/HOWTO/mini/Firewall-Piercing.html

Quota mini-HOWTO
http://metalab.unc.edu/LDP/HOWTO/mini/Quota.html

Secure POP via SSH mini-HOWTO
http://metalab.unc.edu/LDP/HOWTO/mini/Secure-POP+SSH.html

The VPN HOWTO (using SSH)
http://metalab.unc.edu/LDP/HOWTO/mini/VPN.html

RedHat Knowledge Base
http://www.redhat.com/cgi-bin/support?faq

Linux Security Site
http://www.linux-security.org/

## Appendix D: Online security documentation

Bugtraq Archives
http://www.geek-girl.com/bugtraq/

CERT Incident Reporting Guidelines
http://www.cert.org/tech_tips/incident_reporting.html

SECURITY RISK ANALYSIS AND MANAGEMENT
http://www.norman.com/local/whitepaper.htm

An Introduction to Information Security
http://www.certicom.com/ecc/wecc1.htm

Site Security Handbook
http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/21xx/2196

How to Handle and Identify Network Probes
http://www.network-defense.com/papers/probes.html

IANA Port Numbers
http://rlz.ne.mediaone.net/linux/papers/port_numbers

Free Firewall and related tools (large)
http://sites.inka.de/sites/lina/freefire-l/index_en.html

Internet FAQ Consortium (You want FAQ's? We got FAQ's!)
http://www.faqs.org/

## Appendix E: General security sites

SecurityPortal, has a Linux section
http://www.securityportal.com/

Open Security Solutions
http://www.opensec.net/

SANS
http://www.sans.org/

Security Mailing Lists
http://www.iss.net/vd/mail.html

Computer Security Information
http://www.alw.nih.gov/Security/security.html

## Appendix F: General Linux sites

Linux.com
http://www.linux.com/

Linux.org
http://www.linux.org/

## Version history

0.0.1 Initial draft, structure decided, basic information from older writings inserted. 04/01/1999

0.0.2 Major spell checking, formatting changes. The book covers most major topics generally and with RedHat specific instructions. Added ipchains examples where ipfwadm examples are given. 04/03/1999

0.0.3 Started Appendix B (www sites/etc.) and the glossary. Polished up some sections, added some new ones. Table of contents created. 04/07/1999

0.0.4 General cleanup, added sections on Encryption, IPSec and the like. 04/11/1999

0.0.5 Intrusion scanning tools and detection, packet sniffing, added some new sections. Edited some older section in need of cleanup. Got rid of unnecessary graphics which removed about 200k (currently half the bulk). 04/12/1999

0.0.6 Added audits and baselines, inn, CVS, rsync, added some new tools in various sections (ssh, etc.). 04/16/1999

0.0.7 Added some new section titles (Network Based Authentication, X Window System, PAM, etc.), finished some other sections (PPP, Linuxconf, etc) and added some more commercial backup programs. Added a whole whack of Linux vendors. Tools like YaST, super, Linuxconf also added. 04/19/1999

0.0.8 Added some more sections (dpkg, tarballs/tgz's, tftp, etc.), added some more tools. Actually installed Debian 2.1 and spent a few days playing with it. Now to experience Slackware for 0.0.9. Some minor reorganization (SAMBA is it's own subject now as well as Novell connectivity). 04/22/1999

0.0.9 Major reorganization, finished up most sections. Major grammar and spell checking. Put in consistent firewall examples. Added page numbers (wheee). Skipped the release to get 0.1.0 out. New license, polishing, etc. New sections on dealing with attacks, and types of attacks, etc. Officially released to the world. 04/27/1999

0.0.95 Removed the Security Consultants sections, possible conflict of interest and no method of vetting them. Added a whole lot of new tools. Removed the word “RedHat” many times and made the guide somewhat more neutral. 05/08/1999

0.0.98 Added many more tools, started adding some commercial tools in areas other then the backup section. More spell and grammar checking. Added some new sections (password storage, denial of service attacks, etc.) and did some tidying up of the structure. Major updates to squid and the admin tools. Rewrote the administrative tools section. 15/5/1999

0.1.0 Fixed the URL section at the end, tidied up various sections. Added the “Internet connection” checklist. Added about 60+ new programs and apps. Added secure file deletion, software management tools, several new scanning and probe detection tools. New whitepaper section listing good security documents, and a URL listing by section at the end. Added several new appendices listing Linux sites, security sites, whitepapers and more. 29/5/1999