Securing Debian Manual

Javier Fernández-Sanguino Peña <jfs@computer.org>

2.5 (beta) 29 augusti 2002Sat, 17 Aug 2002 12:23:36 +0200

Abstract

This document describes the process of securing and hardening the default Debian installation. It covers some of the common tasks to setup a secure network environment using Debian GNU/Linux and also gives additional information on the security tools available as well as the work done by the Debian security team.

Copyright Notice

Contents

Chapter 1

Introduction

One of the hardest things about writing security documents is that every case is unique. Two things you have to pay attention to are the threat environment and the security needs of the individual site, host, or network. For instance, the security needs of a home user are completely different from a network in a bank. While the primary threat a home user needs to face is the script kiddie type of cracker, a bank network has to worry about directed attacks. Additionally, the bank has to protect their customer's data with arithmetic precision. In short, every user has to consider the tradeoff between usability and security/paranoia.

Note that this manual only covers issues relating to software. The best software in the world can't protect you if someone can physically access the machine. You can place it under your desk, or you can place it in a hardened bunker with an army in front of it. Nevertheless the desktop computer can be much more secure (from a software point of view) than a physically protected one if the desktop is configured properly and the software on the protected machine is full of security holes. Obviously, you must consider both issues.

This document just gives an overview of what you can do to increase the security of your Debian GNU/Linux system. If you have read other documents regarding Linux security, you will find that there are common issues which might overlap with this document. However, this document does not try to be the ultimate source of information you will be using, it only tries to adapt this same information so that it is meaningful to a Debian GNU/Linux system. Different distributions do some things in different ways (startup of daemons is an usual example); here, you will find material which is appropriate for Debian's procedures and tools.

If you have comments, additions or suggestions, please mail them to Javier Fernández-Sanguino (mailto:jfs@computer.org) (alternate address: jfs@debian.org) and they will be incorporated into this manual.

1.1 Download the manual

You can download or view the newest version of the Securing Debian Manual from the Debian Documentation Project (http://www.debian.org/doc/manuals/

securing-debian-howto/). Feel free to check out the version control system through its CVS server (http://cvs.debian.org/ddp/manuals.sgml/securing-howto/?cvsroot=debian-doc).

You can download also a text version (http://www.debian.org/doc/manuals/securing-debian-howto/securing-debian-howto.txt) from the Debian Documentation's Project site. Other formats, like PDF, are not (yet) provided. However, you can download or install the harden-doc (http://packages.debian.org/harden-doc) package which provides this same document in HTML, txt and PDF formats.

1.2 Organizational Notes/Feedback

Now to the official part. At the moment I (Alexander Reelsen) wrote most paragraphs of this manual, but in my opinion this should not stay the case. I grew up and live with free software, it is part of my everyday use and I guess yours, too. I encourage everybody to send me feedback, hints additions or any other suggestions, you might have.

If you think, you can maintain a certain section or paragraph better, then write to the document maintainer and you are welcome to do it. Especially if you find a section marked as FIXME, that means the authors did not have the time yet or the needed knowledge about the topic, drop them a mail immediately.

The topic of this manual makes it quite clear that it is important to keep it up to date, and you can do your part. Please contribute.

1.3 Prior knowledge

The installation of Debian GNU/Linux is not very difficult and you should have been able to install it. If you already have some knowledge about Linux or other Unices and you are a bit familiar with basic security, it will be easier to understand this manual, as this document cannot explain every little detail of a feature (otherwise this would have been a book instead of a manual). If you are not that familiar, however, you might want to take a look at 'Be aware of general security problems' on page 17 for where to find more in-depth information.

1.4 Things that need to be written (FIXME/TODO)

- Add information on how to setup a firewall using Debian GNU/Linux. The section regarding firewalling is oriented currently towards a single system (not protecting others...) also talk on how to test the setup.
- Add information on setting up a proxy firewall with Debian GNU/Linux stating specifically which packages provide proxy services (like xfwp, xproxy, ftp-proxy, redir, smtpd, nntp-cache, dnrd, jftpgw,oops,pnsd, perdition,transproxy, tsocks).

Should point to the manual for any other info. Also note that zorp is not (yet) available as a Debian package but *is* a proxy firewall (they provide Debian packages upstream).

- Information on service configuration with file-rc
- Check all the reference URLs and remove/fix those no longer available.
- Add information on available replacements (in Debian) for common servers which are useful for limited functionality. Examples:
  - local lpr with cups (package)?
  - remote lrp with lpr
  - bind with dnrd/maradns
  - apache with dhttpd/thttpd/wn (tux?)
  - exim/sendmail with ssmtpd/smtpd/postfix
  - squid with tinyproxy
  - ftpd with oftpd/vsftp
  - ...
- More information regarding security-related kernel patches in Debian, including the ones show above and talking specifically on how to enable these patches in a Debian system.
  - Linux Intrusion Detection (`lids-2.2.19`)
  - Linux Trustees (in package `trustees`)
  - NSA Enhanced Linux (`http://www.coker.com.au/selinux/`)
  - kernel-patch-2.2.18-openwall (`http://packages.debian.org/kernel-patch-2.2.18-openwall`)
  - `kernel-patch-2.2.19-harden`
  - Linux capabilities (in package `lcap`
  - `kernel-patch-freeswan`,`kernel-patch-int`
- Details of turning off unnecessary network services (besides inetd), it is partly in the hardening procedure but could be broadened a bit.
- Information regarding password rotation which is closely related to policy.
- Policy, and educating users about policy.
- More about tcpwrappers, and wrappers in general?
- `hosts.equiv` and other major security holes.
- Issues with file sharing servers such as Samba and NFS?
- suidmanager/dpkg-statoverrides.

- lpr and lprng.
- Switching off the gnome IP things.
- Talk about pam_chroot (see http://http://lists.debian.org/debian-security/2002/debian-security-200205/msg00011.html) and its usefulness to limit users. Introduce information related to http://online.securityfocus.com/infocus/1575. `Pdmenu`, for example is available in Debian (while as flash is not).
- Talk about chrooting services, some more info on http://www.linuxfocus.org/English/January2002/aritcle225.shtml, http://www.networkdweebs.com/chroot.html and http://www.linuxsecurity.com/feature_stories/feature_story-99.html
- Talk about programs to make chroot jails. `Compartment` and `chrootuid` are waiting in incoming. Some others (makejail, jailer) could also be introduced.
- Add information provided by Karl Hegbloom regarding chrooting Bind 9, see http://people.pdxlinux.org/~karlheg/Secure_Bind9_uHOWTO/Secure_Bind_9_uHOWTO.xhtml.
- Add information provided by Pedro Zornenon to chrooting Bind 8 only for potato though :(, see http://people.debian.org/~pzn/howto/chroot-bind.sh.txt (include the whole script?).
- More information regarding log analysis software (i.e. logcheck and logcolorise).
- 'advanced' routing (traffic policing is security related)
- limiting ssh access to running certain commands.
- using dpkg-statoverride.
- secure ways to share a CD burner among users.
- secure ways of providing networked sound in addition to network display capabilities (so that X clients' sounds are played on the X server's sound hardware)
- securing web browsers.
- setting up ftp over ssh.
- using crypto loopback filesystems.
- encrypting the entire filesystem.
- steganographic tools.
- setting up a PKA for an organization.
- using LDAP to manage users. There is a HOWTO of ldap+kerberos for Debian at www.bayour.com written by Turbo Fredrikson.

1.5 Changelog/History

1.5.1 Version 2.5 (august 2002)

Changes by Javier Fernández-Sanguino Peña (me). There were many things waiting on my inbox (as far back as february) to be included, so I'm going to tag this the *back from honeymoon* release :)

- Added some information on how to setup the Xscreensaver to lock automatically the screen after the configured timeout.
- Add a note related to the utilities you should not install in the system. Including a note regarding Perl and why it cannot be easily removed in Debian. The idea came after reading Intersect's documents regarding Linux hardening.
- Added information on lvm and journaling filesystems, ext3 recommended. The information there might be too generic, however.
- Added a link to the online text version (check).
- Added some more stuff to the information on firewalling the local system triggered by a comment made by Hubert Chan in the mailing list.
- Added more information on PAM limits and pointers to Kurt Seifried's documents (related to a post by him to bugtraq on April 4th 2002 answering a person that had "discovered" a vulnerability in Debian GNU/Linux related to resource starvation)
- As suggested by Julián Muñoz, provided more information on the default Debian umask and what a user can access if he has been given a shell in the system (scary, huh?)
- Included a note in the BIOS password section due to a comment from from Andreas Wohlfeld.
- Included patches provided by Alfred E. Heggestad fixing many of the typos still present in the document.
- Added a pointer to the changelog in the Credits section since most people who contribute are listed here (and not there)
- Added a few more notes to the chattr section and a new section after installation talking about system snapshots. Both ideas were contributed by Kurt Pomeroy.
- Added a new section after installation just to remember users to change the boot-up sequence.
- Added some more TODO items provided by Korn Andras.
- Added a pointer to the NIST's guidelines on how to secure DNS provided by Daniel Quinlan.

- Added a small paragraph regarding Debian's SSL certificates infraestructure.
- Added Daniel Quinlan's suggestions regarding ssh authentication and exim's relay configuration.
- Added more information regarding securing bind including changes suggested by Daniel Quinlan and an appendix with a scrip to make some of the changes commented on that section.
- Added a pointer to another item regarding Bind chrooting (needs to be merged)
- Added a one liner contributed by Cristian Ionescu-Idbohrn to retrieve packages with tcpwrappers support.
- Added a little bit more info on Debian's default PAM setup.
- Included a FAQ question about using PAM to give services w/o shell accounts.
- Moved two FAQ items to another section and added a new FAQ regarding attack detection (and compromised systems).
- Included information on how to setup a bridge firewall (including a sample Appendix). Thanks go to Francois Bayar who sent me this on march.
- Added a FAQ regarding the syslogd's *MARK heartbeat* from a question answered by Noah Meyerhans and Alain Tesio on December 2001.
- Included information on buffer overflow protection as well as some information on kernel patches.
- Added more information (and reorganised) the firewall section. Updated the information regarding the iptables package and the firewall generators available.
- Reorganized the information regarding logchecking, moved logcheck information from host intrusion detection to that section.
- Added some information on how to prepare a static package for bind for chrooting (untested).
- Added a FAQ item (could be expanded with some of the recomendations from the debian-security list regarding some specific servers/services).
- Added some information on RPC services (and when it's necessary).
- Added some more information on capabilities (and what lcap does). Is there any good documentation on this? I haven't found any on my 2.4 kernel.
- Fixed some typos.

1.5.2 Version 2.4

Changes by Javier Fernández-Sanguino Peña.

- Rewritten part of the BIOS section.

1.5.3 Version 2.3

Changes by Javier Fernández-Sanguino Peña.

- Wrapped most file locations with the file tag.
- Fixed typo noticed by Edi Stojicevi.
- Slightly changed the remote audit tools section.
- Added some todo items.
- Added more information regarding printers and cups config file (taken from a thread on debian-security).
- Added a patch submitted by Jesus Climent regarding access of valid system users to Proftpd when configured as anonymous server.
- Small change on partition schemes for the special case of mail servers.
- Added Hacking Linux Exposed to the books section.
- Fixed directory typo noticed by Eduardo Pérez Ureta.
- Fixed /etc/ssh typo in checklist noticed by Edi Stojicevi.

1.5.4 Version 2.3

Changes by Javier Fernández-Sanguino Peña.

- Fixed location of dpkg conffile.
- Remove Alexander from contact information.
- Added alternate mail address.
- Fixed Alexander mail address (even if commented out).
- Fixed location of release keys (thanks to Pedro Zorzenon for pointing this out).

1.5.5 Version 2.2

Changes by Javier Fernández-Sanguino Peña.

- Fixed typos, thanks to Jamin W. Collins.
- Added a reference to apt-extracttemplate manpage (documents the APT::ExtractTemplate config).
- Added section about restricted SSH. Information based on that posted by Mark Janssen, Christian G. Warden and Emmanuel Lacour on the debian-security mailing list.
- Added information on anti-virus software.
- Added a FAQ: su logs due to the cron running as root.

1.5.6 Version 2.1

Changes by Javier Fernández-Sanguino Peña.

- Changed FIXME from lshell thanks to Oohara Yuuma.
- Added package to sXid and removed comment since it *is* available.
- Fixed a number of typos discovered by Oohara Yuuma.
- ACID is now available in Debian (in the acidlab package) thanks to Oohara Yuuma for noticing.
- Fixed LinuxSecurity links (thanks to Dave Wreski for telling).

1.5.7 Version 2.0

Changes by Javier Fernández-Sanguino Peña. I wanted to change to 2.0 when all the FIXMEs were, er, fixed but I run out of 1.9X numbers :(

- Converted the HOWTO into a Manual (now I can properly say RTFM)
- Added more information regarding tcp wrappers and Debian (now many services are compiled with support for them so it's no longer an inetd issue).
- Clarified the information on disabling services to make it more consistent (rpc info still referred to update-rc.d)
- Added small note on lprng.
- Added some more info on compromised servers (still very rough)

- Fixed typos reported by Mark Bucciarelli.
- Added some more steps in password recovery to cover the cases when the admin has set paranoid-mode=on.
- Added some information to set paranoid-mode=on when login in console.
- New paragraph to introduce service configuration.
- Reorganised the *After installation* section so it is more broken up into several issues and it's easier to read.
- Written information on howto setup firewalls with the standard Debian 3.0 setup (iptables package).
- Small paragraph explaining why installing connected to the Internet is not a good idea and how to avoid this using Debian tools.
- Small paragraph on timely patching referencing to IEEE paper.
- Appendix on how to setup a Debian snort box, based on what Vladimir sent to the debian-security mailing list (September 3rd 2001)
- Information on how logcheck is setup in Debian and how it can be used to setup HIDS.
- Information on user accounting and profile analysis.
- Included apt.conf configuration for read-only /usr copied from Olaf Meeuwissen's post to the debian-security mailing list
- New section on VPN with some pointers and the packages available in Debian (needs content on how to setup the VPNs and Debian-specific issues), based on Jaroslaw Tabor's and Samuli Suonpaa's post to debian-security.
- Small note regarding some programs to automatically build chroot jails
- New FAQ item regarding identd based on a discussion in the debian-security mailing list (February 2002, started by Johannes Weiss).
- New FAQ item regarding inetd based on a discussion in the debian-security mailing list (February 2002).
- Introduced note on rcconf in the "disabling services" section.
- Varied the approach regarding LKM, thanks to Philipe Gaspar
- Added pointers to CERT documents and Counterpane resources

1.5.8 Version 1.99

Changes by Javier Fernández-Sanguino Peña.

- Added a new FAQ item regarding time to fix security vulnerabilities.
- Reorganised FAQ sections.
- Started writing a section regarding firewalling in Debian GNU/Linux (could be broadened a bit)
- Fixed typos sent by Matt Kraai
- Fixed DNS information
- Added information on whisker and nbtscan to the auditing section.
- Fixed some wrong URLs

1.5.9 Version 1.98

Changes by Javier Fernández-Sanguino Peña.

- Added a new section regarding auditing using Debian GNU/Linux.
- Added info regarding finger daemon taken from the security mailing list.

1.5.10 Version 1.97

Changes by Javier Fernández-Sanguino Peña.

- Fixed link for Linux Trustees
- Fixed typos (patches from Oohara Yuuma and Pedro Zorzenon)

1.5.11 Version 1.96

Changes by Javier Fernández-Sanguino Peña.

- Reorganized service installation and removal and added some new notes.
- Added some notes regarding using integrity checkers as intrusion detection tools.
- Added a chapter regarding package signatures.

1.5.12 Version 1.95

Changes by Javier Fernández-Sanguino Peña.

- Added notes regarding Squid security sent by Philipe Gaspar.
- Fixed rootkit links thanks to Philipe Gaspar.

1.5.13 Version 1.94

Changes by Javier Fernández-Sanguino Peña.

- Added some notes regarding Apache and Lpr/lpng.
- Added some information regarding noexec and read-only partitions.
- Rewritten how can users help in Debian security issues (FAQ item).

1.5.14 Version 1.93

Changes by Javier Fernández-Sanguino Peña.

- Fixed location of mail program.
- Added some new items to the FAQ.

1.5.15 Version 1.92

Changes by Javier Fernández-Sanguino Peña.

- Added a small section on how Debian handles security
- Clarified MD5 passwords (thanks to 'rocky')
- Added some more information regarding harden-X from Stephen van Egmond
- Added some new items to the FAQ

1.5.16 Version 1.91

Changes by Javier Fernández-Sanguino Peña.

- Added some forensics information sent by Yotam Rubin.
- Added information on how to build a honeynet using Debian GNU/Linux.
- Added some more TODOS.
- Fixed more typos (thanks Yotam!)

1.5.17 Version 1.9

Changes by Javier Fernández-Sanguino Peña.

- Added patch to fix misspellings and some new information (contributed by Yotam Rubin)
- Added references to other online (and offline) documentation both in a section (see 'Be aware of general security problems' on page 17) by itself and inline in some sections.
- Added some information on configuring Bind options to restrict access to the DNS server.
- Added information on how to automatically harden a Debian system (regarding the harden package and bastille).
- Removed some done TODOs and added some new ones.

1.5.18 Version 1.8

Changes by Javier Fernández-Sanguino Peña.

- Added the default user/group list provided by Joey Hess to the debian-security mailing list.
- Added information on LKM root-kits ('LKM - Loadable Kernel Modules' on page 99) contributed by Philipe Gaspar.
- Added information on Proftp contributed by Emmanuel Lacour.
- Recovered the checklist Appendix from Era Eriksson.
- Added some new TODO items and removed other fixed ones.
- Manually included Era's patches since they were not all included in the previous version.

1.5.19 Version 1.7

Changes by Era Eriksson.

- Typo fixes and wording changes

Changes by Javier Fernández-Sanguino Peña.

- Minor changes to tags in order to keep on removing the tt tags and substitute them for prgn/package tags.

1.5.20 Version 1.6

Changes by Javier Fernández-Sanguino Peña.

- Added pointer to document as published in the DDP (should supersede the original in the near future)
- Started a mini-FAQ (should be expanded) with some questions recovered from my mailbox.
- Added general information to consider while securing.
- Added a paragraph regarding local (incoming) mail delivery.
- Added some pointers to more information.
- Added information regarding the printing service.
- Added a security hardening checklist.
- Reorganized NIS and RPC information.
- Added some notes taken while reading this document on my new Visor :)
- Fixed some badly formatted lines.
- Fixed some typos.
- Added a Genius/Paranoia idea contributed by Gaby Schilders.

1.5.21 Version 1.5

Changes by Josip Rodin and Javier Fernández-Sanguino Peña.

- Added paragraphs related to BIND and some FIXMEs.

1.5.22 Version 1.4

- Small setuid check paragraph
- Various minor cleanups
- Found out how to use `sgml2txt -f` for the txt version

1.5.23 Version 1.3

- Added a security update after installation paragraph
- Added a proftpd paragraph
- This time really wrote something about XDM, sorry for last time

1.5.24 Version 1.2

- Lots of grammar corrections by James Treacy, new XDM paragraph

1.5.25 Version 1.1

- Typo fixes, miscellaneous additions

1.5.26 Version 1.0

- Initial release

1.6 Credits and Thanks!

- Alexander Reelsen wrote the original document.
- Javier Fernández-Sanguino added more info to the original doc.
- Robert van der Meulen provided the quota paragraphs and many good ideas
- Ethan Benson corrected the PAM paragraph and had some good ideas.
- Dariusz Puchalak contributed some information to several chapters.
- Gaby Schilders contributed a nice Genius/Paranoia idea.
- Era Eriksson smoothed out the language in a lot of places and contributed the checklist appendix.
- Philipe Gaspar wrote the LKM information.
- Yotam Rubin contributed fixes for many typos as well as information regarding bind versions and md5 passwords.
- All the people who made suggestions for improvement that (eventually) got included here (see 'Changelog/History' on page 5)

- (Alexander) All the folks who encouraged me to write this HOWTO (which was later turned into a Manual).
- The whole Debian project.

Chapter 2

Before you begin

2.1 What do you want this system for?

Securing Debian is not very different from securing any other system; in order to do it properly, you must first decide what do you intend to do with it. After this, you will have to consider that the following tasks need to be taken care of if you want a really secure system.

You will find that this manual is written from the bottom up, that is, you will read some information on tasks to do before, during and after the installation of your Debian system is made. The tasks can also be thought of as:

- Decide which services you need and limit your system to those. This includes deactivating/uninstalling unneeded services, and adding firewall-like filters, or tcpwrappers.
- Limit users and permissions in your system.
- Harden offered services so that, in the event of a service compromise, the impact to your system is minimized.
- Use appropriate tools to guarantee that unauthorized use is detected so that you can take appropriate measures.

2.2 Be aware of general security problems

The following manual does not (usually) go into the details on why some issues are considered security risks. However, you might want to have a better background regarding general UNIX and (specific) Linux security. Take some time to read over security related documents in order to take informed decisions when you are encountered with different choices. Debian GNU/Linux is based on the Linux kernel, so much of the information regarding Linux, as well as from other distributions and general UNIX security also apply to it (even if the tools used, or the programs available, differ).

Some useful documents include:

- The Linux Security HOWTO (http://www.linuxdoc.org/HOWTO/Security-HOWTO.html) (also available at LinuxSecurity (http://www.linuxsecurity.com/docs/LDP/Security-HOWTO.html)) is one of the best references regarding general Linux Security.
- The Security Quick-Start HOWTO for Linux (http://www.linuxsecurity.com/docs/LDP/Security-Quickstart-HOWTO/) is also a very good starting point for novice users (both to Linux and security).
- The Linux Security Administrator's Guide (http://seifried.org/lasg/) (provided in Debian through the `lasg` package) is a complete guide that touches all the issues related to security in Linux, from kernel security to VPNs. It is somewhat obsolete (not updated since 1999) and has been superseded by the Linux Security Knowledge Base (http://seifried.org/lskb). This documentation is also provided in Debian through the `lskb` package.
- Kurt Seifried's Securing Linux Step by Step (http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html).
- In Securing and Optimizing Linux: RedHat Edition (http://www.linuxdoc.org/links/p_books.html#securing_linux) you can find a similar document to this manual but related to RedHat, some of the issues are not distribution-specific and also apply to Debian.
- IntersectAlliance has published some documents that can be used as references cards on how to harden linux servers (and their services), the documents are available at their site (http://www.intersectalliance.com/projects/index.html).
- For network administrators, a good reference for building a secure network is the Securing your Domain HOWTO (http://www.linuxsecurity.com/docs/LDP/Securing-Domain-HOWTO/).
- If you want to evaluate the programs you are going to use (or want to build up some new ones) you should read the Secure Programs HOWTO (http://www.linuxdoc.org/HOWTO/Secure-Programs-HOWTO.html).
- If you are considering installing Firewall capabilities, you should read the Firewall HOWTO (http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html) and the IPCHAINS HOWTO (http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html) (for kernels previous to 2.4).
- Finally, a good card to keep handy is the Linux Security RefenceCard (http://www.linuxsecurity.com/docs/QuickRefCard.pdf)

In any case, you have more information regarding the services here explained (NFS, NIS, SMB...) in many of the HOWTOs of the Linuxdoc Project (http://www.linuxdoc.org/), some of these documents speak on the security side of a given service, so be sure to take a look there too.

The HOWTO documents from the Linux Documentation Project are available in Debian GNU/Linux through the installation of the `doc-linux-text` (text version) or `doc-linux-html` (html version). After installation these documents will be available at the `/usr/share/doc/HOWTO/en-txt` and `/usr/share/doc/HOWTO/en-html` directories, respectively.

Other recommended Linux books:

- Maximum Linux Security : A Hacker's Guide to Protecting Your Linux Server and Network. Anonymous. Paperback - 829 pages. Sams Publishing. ISBN: 0672313413. July 1999.
- Linux Security By John S. Flowers. New Riders; ISBN: 0735700354. March 1999
- Hacking Linux Exposed (http://www.linux.org/books/ISBN_0072127732.html) By Brian Hatch. McGraw-Hill Higher Education. ISBN 0072127732. April, 2001

Other books (which might be related to general issues regarding UNIX and security and not Linux specific):

- Practical Unix and Internet Security (2nd Edition) (http://www.ora.com/catalog/puis/noframes.html) Garfinkel, Simpson, and Spafford, Gene; O'Reilly Associates; ISBN 0-56592-148-8; 1004pp; 1996.
- Firewalls and Internet Security Cheswick, William R. and Bellovin, Steven M.; Addison-Wesley; 1994; ISBN 0-201-63357-4; 320pp.

Some useful Web sites to keep uptodate regarding security:

- NIST Security Guidelines (http://csrc.nist.gov/fasp/index.html).
- Security Focus (http://www.securityfocus.com) the server that hosts the Bugtraq vulnerability database and list, and provides general security information, news and reports.
- Linux Security (http://www.linuxsecurity.com/). General information regarding Linux security (tools, news...). Most useful is the main documentation (http://www.linuxsecurity.com/resources/documentation-1.html) page.
- Linux firewall and security site (http://www.linux-firewall-tools.com/linux/). General information regarding Linux firewalls and tools to control and administrate them.

2.3 How does Debian handle security?

Just so you have a general overview of security in Debian GNU/Linux you should take note of the different issues that Debian tackles in order to provide an overall secure system:

- Debian problems are always handled openly, even security related. As the Debian Social Contract (http://www.debian.org/social_contract) states: *We Won't Hide Problems We will keep our entire bug-report database open for public view at all times. Reports that users file on-line will immediately become visible to others.* Security issues are discussed openly on the debian-security mailing list. Debian Security Advisories are sent to public mailing lists (both internal an external) and published on the public server.
- Debian follows security issues closely. The security team checks many security related sources, the most important being Bugtraq (http://www.securityfocus.com/cgi-bin/vulns.pl), on the lookout for packages with security issues that might be included in Debian.
- Security updates are the first priority. When a security problem arises in a Debian package, the security update is prepared as fast as possible and distributed for our stable and unstable releases, including all architectures.
- Information regarding security is centralized in a single point, http://security.debian.org/.
- Debian is always trying to improve the overall security of the distribution starting out new projects, like automatic package signature verification mechanisms.
- Debian tries to provide a useful number of security related tools for system administration and monitoring. Developers try to tightly integrate this tools with the distribution in order to make them better suite to enforce local security policies. Tools include: integrity checkers, auditing tools, hardening tools, firewall tools, intrusion detection tools, etc..
- Package maintainers are aware of security issues. This leads to many "secure by default" service installations which might put some limits, sometimes, to its normal use. However, Debian does try to balance security issues and ease of administration, systems are not installed de-activated, for example, like on the BSD family distributions. In any case, some special security issues, like setuid programs, are part of the Debian Policy (http://www.debian.org/doc/debian-policy/).

This same document tries to enforce, as well a better distribution security-wise, by publishing security information specific to Debian which complements other information-security documents related to the tools used by Debian or the operating system itself (see 'Be aware of general security problems' on page 17.

Chapter 3

Before and during the installation

3.1 Choose a BIOS password

Before you install any operating system on your computer, set up a BIOS password. After installation (once you have enabled bootup from the hard disk) you should go back to the BIOS and change the boot sequence to disable booting from floppy, cdrom and other devices that shouldn't boot. Otherwise a cracker only needs physical access and a boot disk to access your entire system.

Disabling booting without a password is even better. This can be very effective if you run a server, because it is not rebooted very often. The downside to this tactic is that rebooting requires human intervention which can cause problems if the machine is not easily accessible.

Note: many BIOSes have well known default master passwords, and there also exist applications to retrieve the passwords from the BIOS. Corollary: don't depend on this measure to secure console access to system.

3.2 Partitioning the system

3.2.1 Choose an intelligent partition scheme

An intelligent partition scheme depends on the how the machine is used. A good rule of thumb is to be fairly liberal with your partitions and to pay attention to the following factors:

- Any directory tree which a user has write permissions to, such as e.g. `/home` and `/tmp`, should be on a separate partition. This reduces the risk of a user DoS by filling up your "/" mount point and rendering the system unusable. (Note: this is not strictly true, since there is always some space reserved for root which a normal user cannot fill)
- Any partition which can fluctuate, e.g. `/var` (especially `/var/log`) should also be on a separate partition. On a Debian system, you should create `/var` a little bit bigger than

normal, because downloaded packages (the apt cache) are stored in `/var/cache/apt/archives`.

- Any partition where you want to install non-distribution software should be on a separate partition. According to the File Hierarchy Standard, this is `/opt` or `/usr/local`. If these are separate partitions, they will not be erased if you (have to) reinstall Debian itself.
- From a security point of view, it makes sense to try to move static data to its own partition, and then mount that partition read-only. Better yet, put the data on read-only media. See below for more details.

In the case of mail server it is important to have a separate partition for the mail spool. Remote users (either knowingly or unknowingly) can fill the mail spool (`/var/mail` and/or `/var/spool/mail`). If the spool is in a separate partition, this situation will not render the system unusable. Otherwise (if the spool directory is in the same place partition as `/var`) the system might have important problems: log entries will not be created, packages can not be installed, and some programs might even have problems starting up (if they use `/var/run`).

Also, for partitions in which you cannot be sure of the needed space, installing Logical Volume Manager (`lvm-common` and the needed binaries for your kernel, this might be either `lvm10`, `lvm6`, or `lvm5`). Using `lvm` you can create volume groups that expand multiple physical volumes.

Selecting the appropiate filesystems

During the system partitioning you also have to decide which filesystem you want to use. The default filesystem selected in the Debian installation for Linux partitions is `ext2`. However, it is recommended you swith to the a journaling filesystem, such as `ext3`, `reiserfs`, `jfs` or `xfs`, to minimize the problems derived from a system crash in the following cases:

- for laptops in all the filesystems installed. That way if you run out of battery unexpectedly or the system freezes due to a hardware issue (such as X configuration which is somewhat common) you can do a hardware reboot with less chances of losing data.
- for production systems which store big amounts of data (like mail servers, ftp servers, network filesystems...) it is recommended on these partitions. That way, in the event of a system crash, the server will take less time to recover and check the filesystems, and decrease the chances of data loss.

Leaving aside the performance issues regarding journaling filesystems (since this sometimes can turn into a religious war), it is usually better to use the `ext3` filesystem. The reason for this is that it is backwards compatible with `ext2`, so if there are any issues with the journaling you can disable it and still have a working filesystem. Also, if you need to recover the system with a bootdisk (or CDROM) you do not use a custom kernel. If the kernel in it is 2.4 `ext3` support is already available, if it is a 2.2 kernel you will be able to boot the filesystem even

if you lose journaling capabilities. If you are using other journaling filesystems you will find that you might not be able to recover unless you have a 2.4 kernel with the needed modules compiled built-in. Also, if you are stuck with a 2.2 kernel in the rescue disk it might even be more difficult to have it access `reiserfs` or `xfs`.

In any case, `ext3` might cause less data loss since it does file-data journaling whileas others do only meta-data journaling, see `http://lwn.net/2001/0802/a/ext3-modes.php3`.

3.3 Do not plug to the Internet until ready

The system you are going to install should not be immediately connected to the Internet during installation. This could sound stupid but is usually done. Since the system will install and activate services immediately, if the system is connected to the Internet and the services are not properly configured you are opening it to attack.

Also note that some services might have new security vulnerabilities not fixed in the packages you are using for installation. This is usually true if you are installing from old media (like CD-ROMs). In this case, it could even be compromised before you even finished installation!

Since Debian installation and upgrades can be done over the Internet you might think it is a good idea to use this feature on installation. If the system is going to be directly connected to the Internet (and not protected by a firewall or NAT), it is best to install without connection to the Internet and using a local packages mirror from both the Debian package sources and the security updates. You can setup package mirrors by using another system connected to the Internet and Debian-specific (if it's a Debian system) tools like `apt-move` or `apt-proxy` or other common mirroring tools to provide the archive to the installed system. If you cannot do this, you can setup firewall rules to limit access to the system while doing the update (see 'Security update protected by a firewall' on page 147).

3.4 Set a root password

Setting a good root password is the most basic requirement for having a secure system.

3.5 Activate shadow passwords and MD5 passwords

At the end of the installation, you will be asked if shadow passwords should be enabled. Answer yes to this question, so passwords will be kept in the file `/etc/shadow`. Only the root user and the group shadow have read access to this file, so no users will be able to grab a copy of this file in order to run a password cracker against it. You can switch between shadow passwords and normal passwords at any time by using `shadowconfig`. Furthermore you are queried during installation whether you want to use MD5 hashed passwords. This is generally a very good idea since it allows longer passwords and better encryption.

Read more on Shadow passwords in Shadow Password (http://www.linuxdoc.org/HOWTO/Shadow-Password-HOWTO.html) (/usr/share/doc/HOWTO/en-txt/Shadow-Password.txt.gz).

3.6 Run the minimum number of services required

Services are programmes such as ftp servers and web servers. Since they have to be *listening* for incoming connections that request the service external computers can connect to yours. Services are sometimes vulnerable (i.e. can be compromised under a given attack) and are hence a security risk.

You should not install services which are not needed on your machine. Every installed service might introduce new, perhaps not obvious (or known), security holes on your computer.

As you may already know, when you install a given service the default behavior is to activate it. In a default Debian installation, with no services installed, the footprint of running services is quite low and it's even lower when talking about services offered in the network. The footprint in Debian 2.1 wasn't as tight as in Debian 2.2 (some inetd services were enabled by default) and in Debian 2.2 the rpc portmapper is enabled upon installation. Rpc is installed by default because it is needed for many services, for example NFS, to run on a given system. It can be easily removed, however, see 'Disabling daemon services' on this page on how to disable it.

When you install a new network-related service (daemon) in your Debian GNU/Linux system it can be enabled in two ways: through the inetd superdaemon (i.e. a line will be added to /etc/inetd.conf) or through a standalone program that binds itself to your network interfaces. Standalone programs are controlled through the /etc/init.d files, which are called at boot time through the SysV mechanism (or an alternative one) by using symlinks in /etc/rc?.d/* (for more information on how this is done read /usr/share/doc/sysvinit/README.runlevels.gz).

If you still want to have some services but you use these rarely, use the update-commands, e.g. 'update-inetd' and 'update-rc.d' for removing them from the startup process.

3.6.1 Disabling daemon services

Disabling a daemon service is quite simple. There are different methods:

- remove links from /etc/rc${runlevel}.d/ or rename the links (so that they do not begin with 'S')
- move the script file (/etc/init.d/_service_name_) to another name (for example /etc/init.d/OFF._service_name_)
- remove the execution bit from the /etc/init.d/_service_name_ file.
- editing the /etc/init.d/_service_name_ script to have it stop immediately.

You can remove the links from `/etc/rc${runlevel}.d/` manually or using `update-rc.d` (see `update-rc.d(8)`). For example, you can disable a service from executing in the multi-user runlevels by doing:

```
update-rc.d stop XX 2 3 4 5 .
```

Please note that, if you are *not* using `file-rc`, `update-rc.d -f _service_ remove` will not work properly, since *all* links are removed, upon re-installation or upgrade of the package this links will be re-generated (probably not what you wanted). If you think this is not intuitive you are probably right (see Bug 67095 (`http://bugs.debian.org/67095`)). From the manpage:

```
If any files /etc/rcrunlevel.d/[SK]??name already exist then
 update-rc.d does nothing.  This is so that the system administrator
 can rearrange the  links,  provided that  they  leave  at  least one
 link remaining, without having their configuration overwritten.
```

If you are using `file-rc` all the information regarding services bootup is handled by a common configuration file and is maintained even if packages are removed from the system.

You can use the TUI (Text User Interface) provided by `rcconf` to do all this changes easily (`rcconf` works both for file-rc and normal System V runlevels).

Other (not recommended) methods of disabling services are: `chmod 644 /etc/init.d/daemon` (but that gives an error message when booting), or modifying the `/etc/init.d/daemon` script (by adding an `exit 0` line at the beginning or commenting out the `start-stop-daemon` part in it). Since init.d files are configfiles, they will not get overwritten upon upgrade.

Unfortunately, unlike other (UNIX) operating systems, services in Debian cannot be disabled by modifying files in `/etc/default/_servicename_`.

FIXME: Add more information on handling daemons using file-rc

3.6.2 Disabling inetd services

You should stop all unneeded services on your system, like echo, chargen, discard, daytime, time, talk, ntalk and r-services (rsh, rlogin and rcp) which are considered HIGHLY insecure (use ssh instead). After disabling those, you should check if you really need the inetd daemon. Many people prefer to use daemons instead of calling services via inetd. Denial of Service possibilities exist against inetd, which can increase the machine's load tremendously. If you still want to run some kind of inetd service, switch to a more configurable inet daemon like xinetd or rlinetd.

You can disable services by editing `/etc/inetd.conf` directly, but Debian provides a better alternative to do this: `update-inetd` (which comments the services in a way that it can easily be turned on again). You could remove the telnet daemon by executing this commands to change the config file and to restart the daemon (in this case the telnet service is disabled):

```
/usr/sbin/update-inetd --disable telnet
```

If you do want services listening, but do not want to have them listen on all IP addresses of your host, you might want to use some undocumented feature on inetd. . Or use an alternate inetd daemon like `xinetd`.

3.7 Install the minimum number of software required

Debian comes with *a lot* of software, for example the Debian 3.0 *woody* release includes almost 6 CD-ROMs of software and thousands of packages. With so many software, and even if the base system installation is quite reduced [1] you might get carried away and install more than it is really needed for your system.

Since you already know what the system is for (don't you?) you should only install software that is really needed for it to work. Any unnecesary tool that is installed might be used by a user that wants to compromise the system or by an external intruder that has gotten shell access (or remote code execution through a exploitable service).

The presence, for example, of development utilities (a C compiler) or interpreted languages (such as perl - but see below -, python, tcl..) may help an attacker compromise the system even further:

- allowing him to do privilege escalation. It's easier, for example, to run local exploits in the system if there is a debugger and compiler ready to compile and test them!
- providing tools that could help the attacker to use the compromised system as a *base of attack* against other systems [2]

Of course, an intruder with local shell access can download his own set of tools and execute them, and even the shell itself can be used to make complex programs. Removing unnecesary software will not help *prevent* the problem but will make it slightly more difficult for an attacker to proceed (and some might give up in this situation looking for easier targets). So, if you leave in a production system tools that could be used to remotely attack systems (see 'Remote vulnerability assesment tools' on page 91) you can expect an intruder to use them too if available.

[1] For example, in Debian woody it is around 40Mbs, try this:

```
$ size=0 $ for i in ‘grep -A 1 -B 1 "^Section: base" /var/lib/dpkg/available |
grep -A 2 "^Priority: required" |grep "^Installed-Size" |cut -d : -f 2 ‘; do
size=$(($size+$i)); done $ echo $size 34234
```

[2] Many intrusions are just made to get access to resources to do ilegitimate activity (denial of service attacks, spam, rogue ftp servers, dns pollution. . .) rather than to just obtain confidential data from the compromised system.

3.7.1 Removing Perl

You must take into account that removing `perl` might not be too easy (as a matter of fact it can be quite difficult) in a Debian system since it is used by many system utilities. Also, the `perl-base` is *Priority: required* (that about says it all). It's still doable, you just have to consider you will not be able to run any perl aplication in the system and you will also have to fool the package management system to think that the `perl-base` is installed even if it's not. [3]

Which utilities use perl? You can see it for yourself doing:

```
$ for i in /bin/* /sbin/* /usr/bin/* /usr/sbin/*; do [ -f $i ] && {
type=‘file $i | grep -il perl‘; [ -n "$type" ] && echo $i; }; done
```

These includes the following utilities in packages with priority *required* or *important*:

- `/usr/bin/chkdupexe` of package `util-linux`.
- `/usr/bin/replay` of package `bsdutils`.
- `/usr/sbin/cleanup-info` of package `dpkg`.
- `/usr/sbin/dpkg-divert` of package `dpkg`.
- `/usr/sbin/dpkg-statoverride` of package `dpkg`.
- `/usr/sbin/install-info` of package `dpkg`.
- `/usr/sbin/update-alternatives` of package `dpkg`.
- `/usr/sbin/update-rc.d` of package `sysvinit`.
- `/usr/bin/grog` of package `groff-base`.
- `/usr/sbin/adduser` of package `adduser`.
- `/usr/sbin/debconf-show` of package `debconf`.
- `/usr/sbin/deluser` of package `adduser`.
- `/usr/sbin/dpkg-preconfigure` of package `debconf`.
- `/usr/sbin/dpkg-reconfigure` of package `debconf`.
- `/usr/sbin/exigrep` of package `exim`.
- `/usr/sbin/eximconfig` of package `exim`.
- `/usr/sbin/eximstats` of package `exim`.
- `/usr/sbin/exim-upgrade-to-r3` of package `exim`.

[3] You can make (on another system) a dummy package with `equivs`

- `/usr/sbin/exiqsumm` of package `exim`.
- `/usr/sbin/keytab-lilo` of package `lilo`.
- `/usr/sbin/liloconfig` of package `lilo`.
- `/usr/sbin/lilo_find_mbr` of package `lilo`.
- `/usr/sbin/syslogd-listfiles` of package `sysklogd`.
- `/usr/sbin/syslog-facility` of package `sysklogd`.
- `/usr/sbin/update-inetd` of package `netbase`.

So, without Perl and, unless you remake these utilities in shell script, you will probably not be able to manage any packages (so you will not be able to upgrade the system, which is *not a good thing*).

If you are determined to remove Perl from the Debian base system, and you have spare time, submit bug reports to the previous packages including (as a patch) replacements for the utilities above written in shell script.

3.8 Read the debian security mailing lists

It is never wrong to take a look at either the debian-security-announce mailing list, where advisories and fixes to released packages are announced by the Debian security team, or at debian-security@lists.debian.org, where you can participate in discussions about things related to Debian security.

In order to receive important security update alerts, send an email to debian-security-announce-request@lists.debian.org (`mailto:debian-security-announce-request@lists.debian.org`) with the word "subscribe" in the subject line. You can also subscribe to this moderated email list via the web page at `http://www.debian.org/MailingLists/subscribe`

This mailing list has very low volume, and by subscribing to it you will be immediately alerted of security updates for the Debian distribution. This allows you to quickly download new packages with security bug fixes, which is very important in maintaining a secure system. (See 'Execute a security update' on page 34 for details on how to do this.)

Chapter 4

After Installation

Once the system is installed you can still secure the system more, some of the steps described in this chapter can be taken. Of course this really depends on your setup but for physical access prevention you should read 'Change the BIOS (again)' on the current page,'Set a LILO or GRUB password' on this page,'Remove root prompt on the kernel' on the next page, 'Disallow floppy booting' on page 31, 'Restricting console login access' on page 32, and 'Restricting system reboots through the console' on page 32.

Before connecting to any network, specially if it's a public one you should, at the very least: execute a security update (see 'Execute a security update' on page 34). Optionally, you could take a snapshot of your system (see 'Taking a snapshot of the system' on page 56).

4.1 Change the BIOS (again)

Remember 'Choose a BIOS password' on page 21? Well, then you should now, once you do not need to boot from removable media, to change the default BIOS setup so that it *only* boots from the hard drive. Make sure you will not lose the BIOS password, otherwise, in the event of a hard disk failure you will not be able to return to the BIOS and change the setup so you can recover it using, for example, a CD-ROM.

Another less secure but more convenient setup would change the setup to have the system boot up from the hard disk and, if it fails, try removable media. By the way, this is often done because since few people use the BIOS password that much often it's easily forgotten.

4.2 Set a LILO or GRUB password

Anybody can easily get a root-shell and change your passwords by entering "<name-of-your-bootimage> init=/bin/sh" at the boot prompt. After changing the passwords and rebooting the system, the person has unlimited root-access and can do anything he/she wants to the system. After this procedure you will not have root access to your system, as you do not know the root password.

To make sure that this can not happen, you should set a password for the boot loader. You can choose between a global password or a password for a certain image.

For LILO you need to edit the config file `/etc/lilo.conf` and add a "password" and "restricted" line as in the example below.

```
image=/boot/2.2.14-vmlinuz
   label=Linux
   read-only
   password=hackme
   restricted
```

When done, rerun lilo. Omitting the "restricted" line causes lilo to always prompt for a password, regardless of whether LILO was passed parameters. The default permissions for `/etc/lilo.conf` grant root read and write permissions, and enable read-only access for lilo.conf's group, root.

If you use GRUB instead of LILO, edit `/boot/grub/menu.lst` and add the following two lines at the top (substituting, of course 'hackme' with the desired password). This prevents users from editing the boot items. 'timeout 3' specifies a 3 second delay before grub boots the default item.

```
timeout 3
password hackme
```

To further harden the integrity of the password, you may store the password in a encrypted form. The utility grub-md5-crypt generates a hashed password which is compatible with grub's encrypted password algorithm (md5). To specify in grub that md5 format password will be used, use the following directive:

```
timeout 3
password --md5 $1$bw0ez$tljnxxKLfMzmnDVaQWgjP0
```

The –md5 parameter was added to instruct grub to perform the md5 authentication process. The provided password is the md5 en crypted version of hackme. Using the md5 password method is preferable to choosing its cleartext counterpart. More information about grub passwords may be found in the grub-doc package.

4.3 Remove root prompt on the kernel

Linux 2.4 kernels provide a way to access a root shell while booting which will be presented just after loading the cramfs filesystem. A message will appear to permit the administrator to enter an executable shell with root permissions, this shell can be used to manually load modules when autodetection fails. This behavior is the default for initrd's linuxrc. The following message will appear:

```
Press ENTER to obtain a shell (waits 5 seconds)
```

In order to remove this behavior you need to change `/etc/mkinitrd/mkinitrd.conf` and set:

```
# DELAY  The  number  of seconds the linuxrc script should wait to
# allow the user to interrupt it before the system is brought up
DELAY=0
```

Then regenerate your ramdisk image. You can do this for example with:

```
o
# cd /boot
# mkinitrd -o initrd.img-2.4.18-k7 /lib/modules/2.4.18-k7
```

or doing (preferred):

```
# dpkg-reconfigure kernel-image-2.4.x-yz
```

Note that Debian 3.0 woody allows users to install 2.4 kernels (selecting *flavors*), *however* the default kernel is 2.2 (save for some architectures for which kernel 2.2 was not ported to). If you consider this a bug consider Bug 145244 (http://bugs.debian.org/145244) before sending it.

4.4 Disallow floppy booting

The default MBR in Debian before version 2.2 did not act as a usual master boot record and left open a method to easily break into a system:

- Press shift at boot time, and an MBR prompt appears
- Then press F, and your system will boot from floppy disk. This can be used to get root access to the system.

This behavior can be changed by entering:

```
lilo -b /dev/hda
```

Now LILO is put into the MBR. This can also be achieved by adding "boot=/dev/hda" to lilo.conf. There is another solution which will disable the MBR prompt completely:

```
install-mbr -i n /dev/hda
```

On the other hand, this "back door", of which many people are just not aware, may save your skin as well if you run into deep trouble with your installation for whatever reasons.

FIXME check whether this really is true as of 2.2 or was it 2.1? INFO: The bootdisks as of Debian 2.2 do NOT install the mbr, but only LILO

4.5 Restricting console login access

Some security policies might want to force administrators to log in the system through the console with their user/password and then become superuser (with `su` or `sudo`). This policy is implemented in Debian by editing the `/etc/login.defs` file or `/etc/securetty` when using PAM. In:

- `login.defs`, editing the CONSOLE variable which defines a file or list of terminals on which root logins are allowed
- `securetty` by adding/removing the terminals to which root access will be allowed.

When using PAM other changes to the login process, which might include restrictions to users and groups at given times, can be configured at `/etc/pam.d/login`. An interesting feature that can be disabled is the possibility to login with null (blank) passwords. This feature can be limited by removing *nullok* from the line:

```
auth       required   pam_unix.so nullok
```

4.6 Restricting system reboots through the console

If you system has a keyboard attached to it anyone (yes *anyone*) can reboot the system through it without login in the system. This might, or might not, adhere to your security policy. If you want to restrict this, you must check the `/etc/inittab` so that the line that includes `ctrlaltdel` calls `shutdown` with the `-a` switch (remember to run `init q` after making any changes to this file). The default in Debian includes this switch:

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Now, in order to allow *some* users to shutdown the system, as the manpage `shutdown(8)` describes, you must create the file `/etc/shutdown.allow` and include there the name of users which can boot the system. When the *three finger salute* (a.k.a. *ctrl+alt+del*) is given the program will check if any of the users listed in the file are logged in. If none of them is `shutdown` will *not* reboot the system.

4.7 Mounting partitions the right way

When mounting an ext2 partition, you have several additional options you apply to the mount call or to `/etc/fstab`. For instance, this my fstab entry for the `/tmp` partition:

```
/dev/hda7    /tmp    ext2    defaults,nosuid,noexec,nodev    0    2
```

You see the difference in the options sections. The option `nosuid` ignores the setuid and setgid bits completely, while `noexec` forbids execution of any program on that mount point, and `nodev`, ignores devices. This sounds great, but it

- only applies to ext2 filesystems
- can be circumvented easily

The `noexec` option prevents binaries from being executed directly, but is easily circumvented:

```
alex@joker:/tmp# mount | grep tmp
/dev/hda7 on /tmp type ext2 (rw,noexec,nosuid,nodev)
alex@joker:/tmp# ./date
bash: ./date: Permission denied
alex@joker:/tmp# /lib/ld-linux.so.2 ./date
Sun Dec  3 17:49:23 CET 2000
```

However, many script kiddies have exploits which try to create and execute files in `/tmp`. If they do not have a clue, they will fall into this pit. In other words, a user cannot be tricked into executing a trojanized binary in `/tmp` e.g. when he incidentally adds `/tmp` into his PATH.

Also be forewarned, some script might depend on `/tmp` begin executable. Most notably, Debconf has (had?) some issues regarding this, for more information see Bug 116448 (http://bugs.debian.org/116448).

The following is a more thorough example. A note, though: `/var` could be set noexec, but some software like Smartlist keeps its programs in /var. The same applies to the nosuid option.

```
/dev/sda6       /usr            ext2    defaults,ro,nodev       0       2
/dev/sda12      /usr/share      ext2    defaults,ro,nodev,nosuid        0
/dev/sda7       /var            ext2    defaults,nodev,usrquota,grpquota
/dev/sda8       /tmp            ext2    defaults,nodev,nosuid,noexec,usrquota
/dev/sda9       /var/tmp        ext2    defaults,nodev,nosuid,noexec,usrquota
/dev/sda10      /var/log        ext2    defaults,nodev,nosuid,noexec    0
/dev/sda11      /var/account    ext2    defaults,nodev,nosuid,noexec    0
/dev/sda13      /home           ext2    rw,nosuid,nodev,exec,auto,nouser,asyn
/dev/fd0        /mnt/fd0        ext2    defaults,users,nodev,nosuid,noexec
/dev/fd0        /mnt/floppy     vfat    defaults,users,nodev.nosuid,noexec
/dev/hda        /mnt/cdrom      iso9660 ro,users,nodev.nosuid,noexec
```

4.7.1 Setting /tmp noexec

Be careful if setting `/tmp` noexec and you want to install new software, since some might use it for installation. Apt is one such program (see http://bugs.debian.org/116448) if not configured properly `APT::ExtractTemplates::TempDir` (see `apt-extracttemplates(1)`). You can set this variable in `/etc/apt/apt.conf` to another directory with exec privileges other than `/tmp`

Regarding noexec, please be aware that it might not offer you that much security. Consider this:

```
$ cp /bin/date /tmp
$ /tmp/date
(does not execute due to noexec)
$/lib/ld-linux.so.2 /tmp/date
(works since date is not executed directly)
```

4.7.2 Setting /usr read-only

If you set `/usr` read-only you will not be able to install new packages on your Debian GNU/Linux system. You will have, first to remount it read-write, install the packages and then remount it read-only. The latest apt version (in Debian 3.0 'woody') can be configured to run commands before and after installing packages, so you might want to configure it properly.

To do this modify `/etc/apt/apt.conf` and add:

```
DPkg
{
    Pre-Invoke  { "mount /usr -o remount,rw" };
    Post-Invoke { "mount /usr -o remount,ro" };
};
```

Note that the Post-Invoke may fail with a "/usr busy" error message. This happens mainly when you are using files during the update that got updated. Annoying but not really a big deal. Just make sure these are no longer used and run the Post-Invoke manually.

4.8 Execute a security update

As soon as new security bugs are detected in packages, Debian maintainers and upstream authors generally patch them within days or even hours. After the bug is fixed, a new package is provided on http://security.debian.org.

If you are installing a Debian release you must take in account that since the release was made there might have been security updates after it has been determined that a given package is

vulnerable. Also, there might have been minor releases (there were up to seven in Debian 2.2 *potato* release) which include these package updates.

You need to note down the date the removable media (if you are using it) was made and check the security site in order to see if there are security updates. If there are and you cannot download the packages from the security site on another system (you are not connected to the Internet yet? aren't you?) before connecting to the network you could consider (if not protected by a firewall for example) adding firewall rules so that you system could only connect to security.debian.org and then run the update. A sample configuration is shown in 'Security update protected by a firewall' on page 147.

To update the system, put the following line in your `sources.list` and you will get security updates automatically, whenever you update your system.

```
deb http://security.debian.org/debian-security stable/updates main contrib no
```

Most people, who don't live in a country which prohibits importing or using strong cryptography, should add this line as well:

```
deb http://security.debian.org/debian-non-US stable/non-US main contrib non-f
```

If you like, you can add the deb-src lines to apt as well. See `apt(8)` for further details.

You should conduct security updates frequently, the vast majority of exploitations result from known vulnerabilities that have not been patched in time, as a `http://www.cs.umd.edu/~waa/vulnerability.html` name="paper by Bill Arbaugh"> (presented on the 2001 IEEE Symposium on Security and Privacy) explains.

FIXME: Add info on how the signature of packages is done so that this can be done automatically through a cron job (big warning: DNS spoofing).

4.9 Providing secure user's access

4.9.1 User authentication: PAM

PAM (Pluggable Authentication Modules) allows system administrators to choose how applications authenticate users. Note that PAM can do nothing unless an application is compiled with support for PAM. Most of the applications that are shipped with Debian 2.2 have this support built in. Furthermore, Debian did not have PAM support before 2.2. The current default configuration for any PAM-enabled service is to emulate UNIX authentication (read `/usr/share/doc/libpam0g/Debian-PAM-MiniPolicy.gz` for more information on how PAM services *should* work in Debian).

Each application with PAM support provides a configuration file in `/etc/pam.d/` which can be used to modify its behavior:

- what backend is used for authentication.
- what backend is used for sessions.
- how do password checks behave.

The following description is far from complete, for more information you might want to read the The Linux-PAM System Administrator's Guide (http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html) (at the primary PAM distribution site (http://www.kernel.org/pub/linux/libs/pam/)), this document is also provided in the `libpam-doc`.

PAM offers you the possibility to go through several authentication steps at once, without the user's knowledge. You could authenticate against a Berkeley database and against the normal passwd file, and the user only logs in if he authenticates correct in both. You can restrict a lot with PAM, just as you can open your system doors very wide. So be careful. A typical configuration line has a control field as its second element. Generally it should be set to "requisite", which returns a login failure if one module fails.

The first thing I like to do, is to add MD5 support to PAM applications, since this helps protects against dictionary cracks (passwords can be longer if using MD5). The following two lines should be added to all files in `/etc/pam.d/` that grant access to the machine, like `login` and `ssh`.

```
# Be sure to install libpam-cracklib first or you will not be able to log in
password   required     pam_cracklib.so retry=3 minlen=12 difok=3
password   required     pam_unix.so use_authtok nullok md5
```

So, what does this incantation do? The first line loads the cracklib PAM module, which provides password strength-checking, prompts for a new password with a minimum length of 12 characters, a difference of at least 3 characters from the old password, and allows 3 retries. The second line introduces the standard authentication module with MD5 passwords and allows a zero length password. The use_authtok directive is necessary to hand over the password from the previous module.

To make sure that the user root can only log into the system from local terminals, the following line should be enabled in `/etc/pam.d/login`:

```
auth     requisite  pam_securetty.so
```

Then you should add the terminals from which the user root can log into the system into `/etc/security/access.conf`. Last but not least the following line should be enabled if you want to set up user limits.

```
session  required   pam_limits.so
```

This restricts the system resources that users are allowed (see below in 'Limiting resource usage: the limits.conf file' on the next page. For example, you could restrict the number of concurrent logins (of a given group of users, or system-wide) you may have, the number of processes, the memory size...

Now edit `/etc/pam.d/passwd` and change the first line. You should add the option "md5" to use MD5 passwords, change the minimum length of password from 4 to 6 (or more) and set a maximum length, if you desire. The resulting line will look something like:

```
password   required   pam_unix.so nullok obscure min=6 max=11 md5
```

If you want to protect su, so that only some people can use it to become root on your system, you need to add a new group "wheel" to your system (that is the cleanest way, since no file has such a group permission yet). Add root and the other users that should be able to `su` to the root user to this group. Then add the following line to `/etc/pam.d/su`:

```
auth        requisite   pam_wheel.so group=wheel debug
```

This makes sure that only people from the group wheel can use `su` to become root. Other users will not be able to become root. In fact they will get a denied message if they try to become root.

If you want only certain users to authenticate at a PAM service, this is quite easy to achieve by using files where the users who are allowed to login (or not) are stored. Imagine you only want to allow user 'ref' to login via ssh. So you put him into `/etc/sshusers-allowed` and write the following into `/etc/pam.d/ssh`:

```
auth        required    pam_listfile.so item=user sense=allow file=/etc/sshus
```

Last, but not least, create `/etc/pam.d/other` and enter the following lines:

```
auth     required       pam_securetty.so
auth     required       pam_unix_auth.so
auth     required       pam_warn.so
auth     required       pam_deny.so
account  required       pam_unix_acct.so
account  required       pam_warn.so
account  required       pam_deny.so
password required       pam_unix_passwd.so
password required       pam_warn.so
password required       pam_deny.so
session  required       pam_unix_session.so
session  required       pam_warn.so
session  required       pam_deny.so
```

These lines will provide a good default configuration for all applications that support PAM (access is denied per default).

4.9.2 Limiting resource usage: the limits.conf file

You should really take a serious look into this file. Here you can define user resource limits. If you use PAM, the file `/etc/limits.conf` is ignored and you should use `/etc/security/limits.conf` instead.

If you do not restrict resource usage, *any* user with a valid shell in your system (or even an intruder who compromised the system through a service) can use up as much CPU, memory, stack, etc. the system can provide. This *resource exhaustion* problem can only be fixed by the use of PAM. Note that there is a way to add resource limits to some shells (for example, bash has `ulimit`, see `bash(1)`), but since not all of them provide the same limits and since the user can change shells (see `chsh(1)`) it is better to place the limits on the PAM modules.

For more information read:

- PAM configuration article (`http://www.samag.com/documents/s=1161/sam0009a/0009a.htm`).
- Seifried's Securing Linux Step by Step (`http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html`) on the *Limiting users overview* section.
- LASG (`http://seifried.org/lasg/users/`) in the *Limiting and monitoring users* section.

FIXME: Get a good limits.conf up here

4.9.3 User Login actions: edit /etc/login.defs

The next step is to edit the basic configuration and action upon user login.

```
FAIL_DELAY          10
```

This variable should be set to a higher value to make it harder to use the terminal to log in using brute force. If a wrong password is typed in, the possible attacker (or normal user!) has to wait for 10 seconds to get a new login prompt, which is quite time consuming when you test passwords. Pay attention to the fact that this setting is useless if using program other than getty, such as mingetty for example.

```
FAILLOG_ENAB        yes
```

If you enable this variable, failed logins will be logged. It is important to keep track of them to catch someone who tries a brute force attack.

```
LOG_UNKFAIL_ENAB    yes
```

If you set the variable "FAILLOG_ENAB" to yes, then you should also set this variable to yes. This will record unknown usernames if the login failed. If you do this, make sure the logs have to the proper permissions (640 for example, with an appropriate group setting such as adm), because users often accidentally enter their password as the username and you do not want others to see it.

```
SYSLOG_SU_ENAB      yes
```

This one enables logging of `su` attempts to syslog. Quite important on serious machines but note that this can create privacy issues as well.

```
SYSLOG_SG_ENAB      yes
```

The same as SYSLOG_SU_ENAB but applies to the `sg` program.

```
MD5_CRYPT_ENAB      yes
```

As stated above, MD5 sum passwords greatly reduce the problem of dictionary attacks, since you can use longer passwords. If you are using slink, read the docs about MD5 before enabling this option. Otherwise this is set in PAM.

```
PASS_MAX_LEN        50
```

If MD5 passwords are activated in your PAM configuration, then this variable should be set to the same value as used there.

4.9.4 Restricting ftp: editing /etc/ftpusers

The `/etc/ftpusers` file contains a list of users who are not allowed to log into the host using ftp. Only use this file if you really want to allow ftp (which is not recommended in general, because it uses cleartext passwords). If your daemon supports PAM, you can also use that to allow and deny users for certain services.

FIXME (BUG): Is it a bug that the default ftpusers in Debian does *not* include all the administrative users (in `base-passwd`).

4.9.5 Using su

If you really need users to become the super user on your system, e.g. for installing packages or adding users, you can use the command `su` to change your identity. You should try to avoid any login as user root and instead use `su`. Actually, the best solution is to remove su and switch to `sudo`, as it has more features than `su`. However, `su` is more common as is used on many other Unices.

4.9.6 Using sudo

`sudo` allows the user to execute defined commands under another user's identity, even as root. If the user is added to `/etc/sudoers` and authenticates himself correctly, he is able to run commands which have been defined in `/etc/sudoers`. Violations, such as incorrect passwords or trying to run a program you don't have permission for, are logged and mailed to root.

4.9.7 Disallow remote adminitrative access

You should modify `/etc/security/access.conf` also so that remote administrative login is disallowed. This way the users need to use `su` (or `sudo`) so that there is always an audit trace whenever a local user wants to use administrative powers.

You need to add the following line to `/etc/security/access.conf`, the default Debian configuration file has a sample line commented out:

```
-:wheel:ALL EXCEPT LOCAL
```

4.9.8 Restricting users

Sometimes you might think you need to have users created in your local system in order to provide a given service (pop3 mail service or ftp). Before doing so, first remember that the PAM implementation in Debian GNU/Linux allows you to validate users with a wide variety of external directory services (radius, ldap, etc.) provided by the libpam packages.

If users need to be created and the system can be accessed remotely take into account that users will be able to login to the system. You can fix this by giving users a null (`/dev/null`) shell (it would need to be listed in `/etc/shells`). If you want to allow users to access the system but limit their movements, you can use the `/bin/rbash`, equivalent to adding the `-r` option in bash (*RESTRICTED SHELL* see `bash(1)`). Please note that even with restricted shell, a user that access an interactive program (that might allow execution of a subshell) could be able to bypass the limits of the shell.

Debian currently provides in the unstable release (and might be included in the next stable releases) the `pam_chroot` module. An alternative to it is to chroot the service that provides remote logging (ssh, telnet).

If you wish to restrict *when* users can access the system you will have to customize `/etc/security/access.conf` for your needs.

Restricting ssh for users

Debian's sshd will not allow you to restrict user's movement through the server since it lacks the Chroot function that the commercial (sshd2) program has (using 'ChrootGroups'

or 'ChrootUsers', see `sshd2_config(5)`). However, there is a patch available that will allow you to do this, the patch can be retrieved from Bug report 139047 (http://bugs.debian.org/139047) or http://www.cag.lcs.mit.edu/~raoul/ (and might be applied in the OpenSSH package in the future). Emanuel Lacour has ssh packages with this feature at http://debian.home-dn.net/woody/ssh/, going through the compilation step is recommended, though. A description of all the steps needed can be found at http://mail.incredimail.com/howto/openssh/ (almost all is applicable to Debian even if it talks about RedHat 7.2). After applying the patch you just need to modify the `/etc/passwd` by changing the home path of the users (with the special `/./` token):

```
joeuser:x:1099:1099:Joe Random User:/home/joe/./:/bin/bash
```

This will restrict *both* remote shell access as well as remote copy through the ssh channel.

Make sure to have all the needed binaries and libraries in the chrooted path for users. These files should be owned by root to avoid tampering by the user (so as to exit the chrooted jailed). A sample might include:

```
./bin:
total 660
drwxr-xr-x    2 root     root         4096 Mar 18 13:36 .
drwxr-xr-x    8 guest    guest        4096 Mar 15 16:53 ..
-r-xr-xr-x    1 root     root       531160 Feb  6 22:36 bash
-r-xr-xr-x    1 root     root        43916 Nov 29 13:19 ls
-r-xr-xr-x    1 root     root        16684 Nov 29 13:19 mkdir
-rwxr-xr-x    1 root     root        23960 Mar 18 13:36 more
-r-xr-xr-x    1 root     root         9916 Jul 26  2001 pwd
-r-xr-xr-x    1 root     root        24780 Nov 29 13:19 rm
lrwxrwxrwx    1 root     root            4 Mar 30 16:29 sh -> bash

./etc:
total 24
drwxr-xr-x    2 root     root         4096 Mar 15 16:13 .
drwxr-xr-x    8 guest    guest        4096 Mar 15 16:53 ..
-rw-r--r--    1 root     root           54 Mar 15 13:23 group
-rw-r--r--    1 root     root          428 Mar 15 15:56 hosts
-rw-r--r--    1 root     root           44 Mar 15 15:53 passwd
-rw-r--r--    1 root     root           52 Mar 15 13:23 shells

./lib:
total 1848
drwxr-xr-x    2 root     root         4096 Mar 18 13:37 .
drwxr-xr-x    8 guest    guest        4096 Mar 15 16:53 ..
-rwxr-xr-x    1 root     root        92511 Mar 15 12:49 ld-linux.so.2
-rwxr-xr-x    1 root     root      1170812 Mar 15 12:49 libc.so.6
```

```
-rw-r--r--    1 root     root         20900 Mar 15 13:01 libcrypt.so.1
-rw-r--r--    1 root     root          9436 Mar 15 12:49 libdl.so.2
-rw-r--r--    1 root     root        248132 Mar 15 12:48 libncurses.so.5
-rw-r--r--    1 root     root         71332 Mar 15 13:00 libnsl.so.1
-rw-r--r--    1 root     root         34144 Mar 15 16:10
libnss_files.so.2
-rw-r--r--    1 root     root         29420 Mar 15 12:57 libpam.so.0
-rw-r--r--    1 root     root        105498 Mar 15 12:51 libpthread.so.0
-rw-r--r--    1 root     root         25596 Mar 15 12:51 librt.so.1
-rw-r--r--    1 root     root          7760 Mar 15 12:59 libutil.so.1
-rw-r--r--    1 root     root         24328 Mar 15 12:57 libwrap.so.0

./usr:
total 16
drwxr-xr-x    4 root     root          4096 Mar 15 13:00 .
drwxr-xr-x    8 guest    guest         4096 Mar 15 16:53 ..
drwxr-xr-x    2 root     root          4096 Mar 15 15:55 bin
drwxr-xr-x    2 root     root          4096 Mar 15 15:37 lib

./usr/bin:
total 340
drwxr-xr-x    2 root     root          4096 Mar 15 15:55 .
drwxr-xr-x    4 root     root          4096 Mar 15 13:00 ..
-rwxr-xr-x    1 root     root         10332 Mar 15 15:55 env
-rwxr-xr-x    1 root     root         13052 Mar 15 13:13 id
-r-xr-xr-x    1 root     root         25432 Mar 15 12:40 scp
-rwxr-xr-x    1 root     root         43768 Mar 15 15:15 sftp
-r-sr-xr-x    1 root     root        218456 Mar 15 12:40 ssh
-rwxr-xr-x    1 root     root          9692 Mar 15 13:17 tty

./usr/lib:
total 852
drwxr-xr-x    2 root     root          4096 Mar 15 15:37 .
drwxr-xr-x    4 root     root          4096 Mar 15 13:00 ..
-rw-r--r--    1 root     root        771088 Mar 15 13:01
libcrypto.so.0.9.6
-rw-r--r--    1 root     root         54548 Mar 15 13:00 libz.so.1
-rwxr-xr-x    1 root     root         23096 Mar 15 15:37 sftp-server
```

4.9.9 Hand-made user auditing

If you are paranoid you might want to add users a defined `.profile` that sets the environment in a way such that they cannot remove audit capabilities from the shell (commands are dumped to `$HISTFILE`. The `.profile` could be set as follows:

```
HISTFILE=/home/_user_/.bash_history
HISTSIZE=100000000000000000
HISTFILESIZE=100000000000000000
set -o HISTFILE
set -o HISTSIZE
set -o HISTFILESIZE
export HISTFILE HISTSIZE HISTFILESIZE
```

Note: the -o attribute sets a variable read-only in bash.

For this to work the user cannot modify the `.profile` or `.bash_history` but must be able to read the first one and write in the second one. You can do this easily by changing these files and the directory where they reside to be owned by another user (root), and give write permissions to the user's group to the history file. Another option is through the use of the `chattr` program.

If you are completely paranoid and want to audit every user's command, you could take bash source code, edit it and have it send all that the user typed into another file. Or have `ttysnoop` constantly monitor any new ttys and dump the output into a file. Other useful program is Snoopy (`http://sourceforge.net/project/?group_id=2091`) which is a user-transparent program that hooks in as a library providing a wrapper around execve() calls, any command execute is logged to syslogd using the `authpriv` facility (usually stored at `/var/log/auth.log`.

Note that you cannot use the `script` command for this since it will not work as a shell (even if you add it to `/etc/shells`.

4.9.10 Complete user audit

The previous example is a simple way to configure user auditing which might be not useful for complex systems. If this is your case, you need to look at `acct`, the accounting utilities. These will log all the commands run by users or processes in the system, at the expense of disk space.

When activating accounting, all the information on processes and user is kept under `/var/account/`, more specifically in the `pacct`. The accounting package includes some tools (`sa` and `ac`) to analyse this data.

4.9.11 Reviewing user profiles

If you want to *see* what are users usually doing, when are they connecting you can use the `wtmp` database that includes all login information. This file can be processed with several utilities, amongst them `sac` which can output a profile on each user showing in which timeframe they usually log on to the system.

In case you have accounting activated, you can also use the tools provided by it in order to determine when the users access to the system and what do they execute.

4.9.12 Setting users umasks

Depending on your user policy you might want to change how information is shared between users, that is, what are the default permissions of new files created by users. This change is set by defining a proper `umask` setting for all users. You can change the UMASK setting in `/etc/limits.conf`, `/etc/profile`, `/etc/csh.cshrc`, `/etc/csh.login`, `/etc/zshrc` and probably some others (depeding on the shells you have installed on your system). Of all of these the last one that gets loaded takes precedence. The order is: PAM's limits.conf, the default system configuration for the user's shell, the user's shell (his `~/.profile`, `~/.bash_profile`...)

Debian's default `umask` setting is *022* this means that files (and directories) can be read and accessed by the user's group and by any other users in the system. If this is too permissive for your system you will have to change the umask setting for all the shells (and for PAM). Don't forget to modify the files under `/etc/skel/` since these will be a user's defaults when it gets created with the `adduser` command.

Note, however that users can modify their own umask setting if they want too, making it more permissive or more restricted.

4.9.13 Limiting what users can see/access

FIXME: Content needed. Tell of consequences of changing packages permissiones when upgrading (and admin this paranoid should chroot his users BTW).

If you need to grant users acces to the system with a shell think it very carefully. A user can, by default, and unless a severely restricted environment (like a `chroot` jail) can retrieve quite a lot of information from your system including:

- some configuration files in `/etc`. However, Debian's default permissions for some sensitive files (which might, for example, contain passwords), will prevent access to critical information. To see which files are only accesible by the root user for example `find /etc -type f -a -perm 600 -a -uid 0` as superuser.
- your installed packages, either by looking at the package database, at the `/usr/share/doc` directory or by guessing by looking at the binaries and libraries installed in your system.
- some logfiles at `/var/log`. Note also that some logfiles are only accesible to root and the `adm` group (try `find /var/log -type f -a -perm 640`) and some even are only available to the root user (try `find /var/log -type f -a -perm 600 -a -uid 0`).

What can a user see in your system? Probably quite a lot of things, try this (take a deep breath):

```
find / -type f -a -perm +006 2>/dev/null
find / -type d -a -perm +007 2>/dev/null
```

The output is the list of files that a user can *see* and the directories he has access to.

4.10 Using tcpwrappers

TCP wrappers were developed when there were no real packet filters available and access control was needed. The TCP wrappers allow you to allow or deny a service for a host or a domain and define a default allow or deny rule. If you want more informations take a look at `hosts_access(5)`.

Many services installed in Debian are either:

- launched through the tcpwrapper service (`tcpd`)
- compiled with libwrapper support built-in.

On the first hand, of services are configured in `/etc/inetd.conf`, this includes telnet, ftp, netbios, swat and finger (you will see that the configuration file executes first `/usr/sbin/tcpd`. On the other hand, even if a service is not launched by the `inetd` superdaemon can, in any case, subjected to the tcp wrappers rules by compiling its support in it. Services compiled with tcp wrappers in Debian include ssh, portmap, in.talk, rpc.statd, rpc.mountd, gdm, oaf (the GNOME activator daemon), nessus and many others.

To see which packages use tcpwrappers try:

```
$ apt-cache showpkg libwrap0 | egrep '^[[:space:]]' | sort -u | \
        sed 's/,libwrap0$//;s/^[[:space:]]\+//'
```

Take this into account when running `tcpchk`. You can add services that are linked to the wrapper library into the `host.deny` and `hosts.allow` files but `tcpchk` will warn that he is not able to find those services since it looks for them in `/etc/inetd.conf` (the manpage is not totally accurate here).

Now, here comes a small trick, and probably the smallest intrusion detection system available. In general, you should have a decent firewall policy as a first line, and tcp wrappers as the second line of defense. One little trick is to set up a SPAWN [1] command in /etc/hosts.deny that sends mail to root whenever a denied service triggers wrappers:

```
ALL: ALL: SPAWN ( \
  echo -e "\n\
  TCP Wrappers\: Connection refused\n\
  By\: $(uname -n)\n\
  Process\: %d (pid %p)\n\
  User\: %u\n\
```

[1] beware of the case here since *spawn* will not work

```
  Host\: %c\n\
  Date\: $(date)\n\
" | /usr/bin/mail -s "Connection to %d blocked" root) &
```

Beware: The above printed example can easily be DoSed by doing lots of connections in a short period of time. Many emails mean a lot of file I/O by sending only a few packets.

4.11 The importance of logs and alerts

How log and alerts are treated is an important issue in a secure system. It is easily to see that, even if the system is perfectly configured and, supposedly, 99% secure. If the 1% comes to happen, and there are no security measures in place to, first, detect this and, second, raise alarms, the system is not secure at all.

Debian GNU/Linux provides some tools to make loganalysis, most notably `logcheck` or `loganalysis` (both will need some customisation to remove unnecessary things from the report). It might be also useful, if the system is nearby, to have the system logs printed on a virtual console. This is useful since you can (from a distance) see if the system is behaving properly. Debian's `/etc/syslog.conf` comes with a commented default configuration, to enable it uncomment the lines and restart syslog (`/etc/init.d/syslogd restart`):

```
daemon,mail.*;\
        news.=crit;news.=err;news.=notice;\
        *.=debug;*.=info;\
        *.=notice;*.=warn       /dev/tty8
```

There is a lot regarding log analysis that cannot be fully covered here, a good resource for information is Counterpane's Log Analysis Resources (http://www.counterpane.com/log-analysis.html). In any case, even automated tools are no match for the best analysis tool: your brain.

4.11.1 Using and customising logcheck.

The `logcheck` package in Debian is divided into two packages `logcheck` (the main program) an `logcheck-database` (a database of regular expressions for the program). The Debian default (at `/etc/cron.d/logcheck`) is that `logcheck` is run daily at 2 am and once after each reboot.

This tool can be quite useful if properly customised to alert the administrator on unusual events in the system. `Logcheck` can be fully customised so that it can send mails from events recovered from the logs that are worthy of attention. The default installation includes profiles for ignored events and policy violations for three different setups (workstation, server and paranoid). The Debian package includes a configuration file `/etc/logcheck/logcheck.conf`, sourced by the program, that defines

which user are the checks send to. It also provides a way for packages that provide services to implement new policies in the directories: `/etc/logcheck/hacking.d/_packagename_`, `/etc/logcheck/violations.d/_packagename_`, `/etc/logcheck/violations.ignore.d/_packagename_`, `/etc/logcheck/ignore.d.paranoid/_packagename_`, `/etc/logcheck/ignore.d.server/_packagename_`, and `/etc/logcheck/ignore.d.workstation/_packagename_`. However, not many packages currently do so. If you have a policy that can be useful for other users, please send it as a bug report for the appropiate package (as a *wishlist* bug). See For more information read `/usr/share/doc/logcheck/README.Debian`

The best way to configure `logcheck` is to install it (it will ask for the user to mail reports to and generate `/etc/logcheck/logcheck.logfiles` from syslog entries). If you wish to add new logfiles just change `/etc/logcheck/logcheck.logfiles` adding them there. The package dependancy will make the `logcheck-database` package get installed too, during installation it will ask which security level is desired: workstation, server or paranoid. This will make `/etc/logcheck/ignore.d` point to the appropiate directories (through symbolic links). To change this run `dpkg-reconfigure -plow logcheck-database`. Then create the `/etc/ignore.d/local`, this file will hold all the rules to exclude messages that should not be reported. Leave it empty for the moment (a simple `cp /dev/null /etc/ignore.d/local` will work).

Once this is done you might want to check the mails that are sent, for the first few days/weeks/months, if you find are sent messages you do not wish to recive you just have to add to the `/etc/ignore.d/local` the regular expressions (see `regex(7)`) that correspond to these messages. It's an ongoing tuning process, once the messages that are sent are always relevant you can consider the tuning finished. Note that if logcheck does not find anything relevant in your system it will not mail you even if it does run (so you might get only a mail once a week, if you are lucky).

4.11.2 Configuring where alerts are sent

Debian comes with a standard syslog configuration (in /etc/syslog.conf) that logs messages to the appropriate files depending on the system facility. You should be familiar with this; have a look at the `syslog.conf` file and the documentation if not. If you intend to maintain a secure system you should be aware of where log messages are sent so they do not go unnoticed.

For example, sending messages to the console also is an interesting setup useful for many production-level systems. But for many such systems it is important to also add a new machine that will serve as loghost (i.e. it receives logs from all other systems).

Root's mail should be considered also, many security controls (like `snort`) send alerts to root's mailbox. This mailbox usually points to the first user created in the system (check `/etc/aliases`). Take care to send root's mail to some place where it will be read (either locally or remotely).

There are other role accounts and aliases on your system. On a small system, it's probably simplest to make sure that all such aliases point to the root account, and that mail to root is forwarded to the system administrator's personal mailbox.

FIXME: it would be interesting to tell how a Debian system can send/receive SNMP traps related to security problems (jfs). Check: `snmptraglogd`, `snmp` and `snmpd`.

4.11.3 Using a loghost

A loghost is a host which collects syslog data remotely over the network. If one of your machines is cracked, the intruder is not able to cover his tracks, unless he hacks the loghost as well. So, the loghost should be especially secure. Making a machine a loghost is simple. Just start the syslogd with 'syslogd -r' and a new loghost is born. In order to do this permanently in Debian, edit `/etc/init.d/sysklogd` and change the line

```
SYSLOGD=""
```

to

```
SYSLOGD="-r"
```

Next, configure the other machines to send data to the loghost. Add an entry like the following to `/etc/syslog.conf`:

```
facility.level          @your_loghost
```

See the documentation for what to use in place of *facility* and *level* (they should not be entered verbatim like this). If you want to log everything remotely, just write:

```
*.*                     @your_loghost
```

into your syslog.conf. Logging remotely as well as locally is the best solution (the attacker might presume to have covered his tracks after deleting the local log files). See the `syslog(3)`, `syslogd(8)` and `syslog.conf(5)` manpages for additional information.

4.11.4 Logfile permissions

It is not only important to decide how alerts are used, but also who has access to them, i.e. can read or modify the logfiles (if not using a remote loghost). Security alerts which the attacker can change or disable are not much worth in the event of an intrusion. Also, you have to take into account that logfiles might reveal an intruder quite a lot of information about your system and it's normal (or abnormal) operations if he has access to them.

Some logfile permissions are not perfect after the installation (but of course this really depends on your local security policy). First `/var/log/lastlog` and `/var/log/faillog` do not need to be readable by normal users. In the lastlog file you can see who logged recently, and

in the faillog you see a summary of failed logins. The author recommends chmod'ing both to 660. Take a brief look over your log files and decide very carefully which logfiles you make readable/writeable for a user with an UID other than 0 and a group other than 'adm' or 'root'. You can easily check this in your system with:

```
#  find /var/log -type f -exec ls -l {} \; | cut -c 17-35 |sort -u
(see to what users do files in /var/log belong to)
#  find /var/log -type f -exec ls -l {} \; | cut -c 26-34 |sort -u
(see to what groups do files in /var/log belong to)
# find /var/log -perm +004
(files which are readable by any user)
#  find /var/log \! -group root \! -group adm -exec ls -ld {} \;
(files which belong to groups not root or adm)
```

To customize how logfiles are created you will probably have to customize the program that generates them. If the logfile gets rotated, however, you can customize the behavior of creation and rotation

4.12 Using chroot

`chroot` is one of the most powerful possibilities to restrict a daemon or a user or another service. Just imagine a jail around your target, which the target cannot escape from (normally, but there are still a lot of conditions that allow one to escape out of such a jail). If you do not trust a user, you can create a change root environment for him. This can use quite a bit of disk space as you need to copy all needed executables, as well as libraries, into the jail. Even if the user does something malicious, the scope of the damage is limited to the jail.

A good example for this case is, if you do not authenticate against `/etc/passwd` but use LDAP or MySQL instead. So your ftp-daemon only needs a binary and perhaps a few libraries. A chrooted environment would be an excellent security improvement; if a new exploit is known for this ftp-daemon, then attackers can only exploit the UID of the ftp-daemon-user and nothing else.

Of course, many other daemons could benefit from this sort of arrangement as well.

However, be forewarned that a `chroot` jail can be broken if the user running in it is the superuser. So, you need to make the service run as a non-privileged user. By limiting its environment you are limiting the world readable/executable files the service can access, thus, you limit the possibilities of a privilege escalation by use of local system security vulnerabilities. Even in this situation you cannot be completely sure that there is no way for a clever attacker to somehow break out of the jail. Using only server programs which have a reputation for being secure is a good additional safety measure. Even minuscule holes like open file handles can be used by a skilled attacker for breaking into the system. After all, `chroot` was not designed as a security tool but as a testing tool.

As an additional note, the Debian default BIND (the Internet name service) is not shipped chrooted per default; in fact, no daemons come chrooted.

There is also some software (not currently in Debian but which might be packaged in the future) that can help setup chroot environments. `makejail` for example, can create and update a chroot jail with short configuration files. It also attempts to guess and install into the jail all files required by the daemon. More information at http://www.floc.net/makejail/. `Jailer` is a similar tool which can be retrieved from http://www.balabit.hu/downloads/jailer/.

Also useful to create chroots (or jails) is `deb.pl`, a script that analyses dependencies of a set of files.

4.12.1 Kernel configuration

4.12.2 Configuring kernel network features

FIXME: Content missing

Many features of the kernel can be modified while running by echoing something into the `/proc` file system or by using sysctl. By entering `sysctl -A` you can see what you can configure and what the options are. Only in rare cases do you need to edit something here, but you can increase security that way as well.

```
net/ipv4/icmp_echo_ignore_broadcasts = 1
```

This is a 'windows emulator' because it acts like windows on broadcast ping if this one is set to 1. Otherwise, it does nothing.

```
net/ipv4/icmp_echo_ignore_all = 0
```

If you don't want to block ICMP on your firewall, enable this.

```
net/ipv4/tcp_syncookies = 1
```

This option is a double-edged sword. On the one hand it protects your system against syn flooding; on the other hand it violates defined standards (RFCs). This option is quite dumb as it floods the other side like it floods you, so the other side is also busy. If you want to change this option you also can change it in `/etc/network/options` by setting `syncookies=yes`.

```
/proc/sys/net/ipv4/conf/all/log_martians = 1
```

Packets with impossible addresses (due to wrong routes) on your network get logged.

Here is an example to set up this and other useful stuff. You should add this information to a script in `/etc/network/interface-secure` (the name is given as an example) and call it from `/etc/network/interfaces` like this:

```
auto eth0
iface eth0 inet static
        address xxx.xxx.xxx.xxx
        netmask 255.255.255.xxx
        broadcast xxx.xxx.xxx.xxx
        gateway xxx.xxx.xxx.xxx
        pre-up /etc/network/interface-secure

# Script-name: /etc/network/interface-secure
# Modifies some default behaviour in order to secure against
# some TCP/IP spoofing & attacks
#
# Contributed by Dariusz Puchalak
#
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
                                        # broadcast echo protection enable
echo 0 > /proc/sys/net/ipv4/ip_forward     # ip forwarding disabled
echo 1 > /proc/sys/net/ipv4/tcp_syncookies # TCP syn cookie protection enable
echo 1 >/proc/sys/net/ipv4/conf/all/log_martians
                                        # Log packets with impossible addr
                          # but be careful with this on heavy loaded web serve
echo 1 > /proc/sys/net/ipv4/ip_always_defrag
                                        #  defragging protection always en
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
                                        # bad error message protection ena

# now ip spoofing protection
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
        echo 1 > $f
done

# and finally some more things:
# Disable ICMP Redirect Acceptance
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
        echo 0 > $f
done

for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
      echo 0 > $f
done

# Disable Source Routed Packets
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
        echo 0 > $f
done
```

```
# Log Spoofed Packets, Source Routed Packets, Redirect Packets
for f in /proc/sys/net/ipv4/conf/*/log_martians; do
        echo 1 > $f
done
```

4.12.3 Configuring firewall features

In order to have firewall capabilities, either to protect the local system or others *behind* it, the kernel needs to be compiled with firewall capabilities. The standard Debian 2.2 kernel (also 2.2) provides the packet filter `ipchains` firewall, Debian 3.0 standard kernel (kernel 2.4) provides the *stateful* packet filter `iptables` (netfilter) firewall. Older Debian distributions would need the appropiate kernel patch (Debian 2.1 uses kernel 2.0.34).

In any case, it is pretty easy to use a kernel different from the one provided by Debian. You can find pre-compiled kernels as packages you can easily install in the Debian system. You can also download the kernel sources using the `kernel-source-X` and build custom kernel packages using `make-kpkg`.

Setting up firewalls in Debian is discussed more thoroughly in 'Adding firewall capabilities' on page 74.

4.13 Adding kernel patches

FIXME: More content

Debian GNU/Linux provides some of the patches for the Linux kernel that enhace its security. These include:

- Linux Intrusion Detection (in package `lids-2.2.19`)
- Linux Capabilities (in package `lcap`)
- Linux Trustees (in package `trustees`)
- NSA Enhanced Linux (in package `selinux` also available from the developer's website (`http://www.coker.com.au/selinux/`))
- kernel-patch-2.2.18-openwall (`http://packages.debian.org/kernel-patch-2.2.18-openwall`).
- `kernel-patch-2.2.19-harden`
- Linux capabilities (in package `lcap`
- IPSEC kernel support (in package `kernel-patch-freeswan`)
- `kernel-patch-int`

However, some patches have not been provided in Debian yet. If you feel that some of these should be included please ask for it at the Work Needing and Prospective Packages (http://wnpp.debian.org). Some of these are:

- PaX patch (http://pageexec.virtualave.net/)
- HAP patch (http://www.theaimsgroup.com/~hlein/hap-linux/)
- Stealth patch (http://www.energymech.net/madcamel/fm/)

4.14 Protecting against buffer overflows

Buffer overflow is the name of a common attack to software which makes use of insufficient boundary checking (a common programming error) in order to execute machine code through a program's inputs. These attacks, against server software which listens to connections remotely and against local software which grant higher priviledges to users (setuid or setgid) can result in the compromise of any given system.

There are mainly four methods to protect against buffer overflows:

- patch the kernel to prevent stack execution.
- using a library, such as libsafe, to overwrite vulnerable functions and introduce proper checking (for information on how to install libsafe read this (http://www.Linux-Sec.net/harden/libsafe.uhow2.txt)).
- recompile code to introduce proper checks that prevent overflows, using, for example, stackguard.
- use tools to find and fix code that might introduce this vulnerability.

Debian GNU/Linux, as of the 3.0 release, only provides software to implement the first and last of these methods (kernel patches and tools to detect possible buffer overflows). The use of tools to detect buffer overflows requires, in any case, of programming experience in order to fix (and recompile) the code. Debian provides, for example: `bfbtester` (a buffer overflow tester that brute-forces binaries through command line and environment overflows) and `njamd`.

As for kernel patches (described in the section 'Adding kernel patches' on the preceding page), the openwall patch provides protection against buffer overflows in 2.2 linux kernels. However, for 2.4 kernels, you need to use the Grsecurity patch (in the `kernel-patch-2.4-grsecurity` which includes the Openwall patch and many more features (http://www.grsecurity.net/features.htm) (including ACLs and network randomness to make it more difficult to remote OS fingerprinting), or the Linux Security Modules (in the `kernel-patch-2.4-lsm` and `kernel-patch-2.5-lsm` packages).

In any case, be aware, that even these workarounds might not prevent buffer overflows since there are ways to circumvent these, as described in phrack's magazine issue 58 (http://packetstorm.linuxsecurity.com/mag/phrack/phrack58.tar.gz).

4.15 Secure file transfers

During normal system administration one usually needs to transfer files in and out from the installed system. Copying files in a secure manner from a host to another can be achieved by using the `sshd` server package. Another possibility is the use of `ftpd-ssl`, a ftp server which uses the *Secure Socket Layer* to encrypt the transmissions.

Any of these methods needs, of course, special clients. Debian provides clients, for example the `ssh` provides `scp`. It works like `rcp` but is encrypted completely, so the *bad guys* cannot even find out WHAT you copy. There is also a `ftp-ssl` client package for the equivalent server. You can find clients for these software even for other operating systems (non-UNIX), `putty` and `winscp` provide secure copy implementations for any version of Microsoft's operating system.

4.16 Filesystem limits and control

4.16.1 Using quotas

Having a good quota policy is important, as it keeps users from filling up the hard disk(s).

You can use two different quota systems: user quota and group quota. As you probably figured out, user quota limits the amount of space a user can take up, group quota does the equivalent for groups. Keep this in mind when you're working out quota sizes.

There are a few important points to think about in setting up a quota system:

- Keep the quotas small enough, so users do not eat up your disk space.
- Keep the quotas big enough, so users do not complain or their mail quota keeps them from accepting mail over a longer period.
- Use quotas on all user-writable areas, on `/home` as well as on `/tmp`.

Every partition/directory which users have full write access should be quota enabled. Find out those partitions and directories and calculate a workable quota size, which combines usability and security.

So, now you want to use quotas. First of all you need to check whether you enabled quota support in your kernel. If not, you will need to recompile it. After this, control whether the package 'quota' is installed. If not you will need this one as well.

Enabling quota for the respective filesystems is as easy as modifying the `defaults` setting to `defaults,usrquota` in your `/etc/fstab` file. If you need group quota, substitute `usrquota` to `grpquota`. You can also use them both. Then create empty quota.user and quota.group files in the roots of the filesystems you want to use quotas on (e.g. `touch /home/quota.user /home/quota.group` for a `/home` filesystem).

Restart quota by doing `/etc/init.d/quota stop;/etc/init.d/quota start`. Now quota should be running, and quota sizes can be set.

Editing quotas for a specific user (say 'ref') can be done by `edquota -u ref`. Group quotas can be modified with `edquota -g <group>`. Then set the soft and hard quota and/or inode quotas as needed.

For more information about quotas, read the quota man page, and the quota mini-howto(`/usr/share/doc/HOWTO/en-html/mini/Quota.html`).

You might or might not like `lshell`, since it violates the FHS. Also take into account that pam_limits.so might provide the same functionality and `lshell` is currently orphaned (http://bugs.debian.org/93894)

4.16.2 chattr/lsattr

These two commands are very useful, but they only work for the ext2 filesystem. With 'lsattr' you can list the attributes of a file and with 'chattr' you can change them. Note that attributes are not the same thing as permissions. There are many attributes, but only the most important for increasing security are mentioned here. There are two flags which can only be set by the superuser.

First there is the 'a' flag. If set on a file, this file can only be opened for appending. This attribute is useful for some of the files in `/var/log/`, though you should consider they get moved sometimes due to the log rotation scripts.

The second flag is the 'i' flag, short for immutable. If set on a file, it can neither be modified nor deleted or renamed and no link be created to it. If you do not want users to look into your config files you could set this flag and remove readability. Furthermore it can give you a little bit more security against intruders, because the cracker might be confused by not being able to remove a file. Nevertheless, you should never assume that the cracker is blind. After all, he got into your system.

You can, also, remove the `chattr` and `lsattr` programs from the system so that an intruder with root access cannot change (or list) this attributes. Since they are part of the `e2fsprogs` and it's *Required* priority you cannot simply remove it. However, you can safely delete these two applications (and probably some others) from the filesystem. Copy them before to a removable media (floppy disk?) along with they md5sums.

An intruder in the system would have to download his own copies of the binaries in the system (probably even compile them in it) which might give you a littler more time to detect and recover from the compromise before the whole system is overrun.

FIXME: This is a bug that could be reported, are any of the binaries provided by the program useful in production systems? If not, and since the libraries are needed by many packages a new package `e2fsprogs-utils` could be included with less than *Required* priority.

Remember: lsattr and chattr are only available on ext2 filesystems.

4.16.3 Checking filesystem integrity

Are you sure `/bin/login` on your hard drive is still the binary you installed there some months ago? What if it is a hacked version, which stores the entered password in a hidden file or mails it in cleartext version all over the internet?

The only method to have some kind of protection is to check your files every hour/day/month (I prefer daily) by comparing the actual and the old md5sum of this file. Two files cannot have the same md5sum (the MD5 digest is 128 bits, so the chance that two different files will have the same md5sum is roughly one in 3.4e3803), so you're on the safe site here, unless someone has also hacked the algorithm that creates md5sums on that machine. This is, well, extremely difficult and very unlikely. You really should consider this auditing of your binaries as very important, since it is an easy way to recognize changes at your binaries. Common tools used for this are `sXid`, `AIDE` (Advanced Intrusion Detection Environment), `TripWire` (non-free; the new version will be GPL), `integrit` and `samhain`.

Installing `debsums` will help to check the filesystem integrity, by comparing the md5sums of every file against the md5sums used in the Debian package archive. But beware, those files can easily be changed.

Furthermore you can replace `locate` with `slocate`. slocate is a security enhanced version of GNU locate. When using slocate, the user only sees the files he really has access to and you can exclude any files or directories on the system.

FIXME: put references to the snapshot taken after installation.

FIXME: Add a note regarding packages not providing debsums for all apps installed (not mandatory).

4.16.4 Setting up setuid check

Debian provides a cron job that runs daily in `/etc/cron.daily/standard`. This cron job will run the `/usr/sbin/checksecurity` script that will store information of this changes.

In order for this check to be made you must set `CHECKSECURITY_DISABLE="FALSE"` in `/etc/checksecurity.conf`. Note, this is the default, so unless you have changed something, this option will already be set to "FALSE".

The default behavior does not send this information to the superuser but, instead keeps daily copies of the changes in `/var/log/setuid.changes`. You should set the CHECKSECURITY_EMAIL (in `/etc/checksecurity.conf`) to 'root' to have this information mailed to him. . See `checksecurity(8)` for more configuration info.

4.17 Taking a snapshot of the system

Before putting the system into production system you culd take a snapshot of the whole system. This snapshot could be used in the event of a compromise (see 'After the compromise' on

page 105). You should remake this upgrade whenever the system is upgraded, specially if you upgrade to a new Debian release.

For this you can use a writable removable-media that can be setup read-only, this could be a floppy disk (read protected after use) or a CD on a CD-ROM unit (you could use a rewriteable CD-ROM so you could even keep backups of md5sums in different dates).

The following script creates such a snapshot:

```
#!/bin/bash
/bin/mount /dev/fd0 /mnt/floppy
/bin/cp /usr/bin/md5sum /mnt/floppy
echo "Calculating md5 database"
>/mnt/floppy/md5checksums.txt
for dir in /bin/ /sbin/ /usr/bin/ /usr/sbin/ /lib/ /usr/lib/
do
   find $dir -type f | xargs /usr/bin/md5sum >>/mnt/floppy/md5checksums-lib.t
done
/bin/umout /dev/fd0
echo "post installation md5 database calculated"
```

Note that the md5sum binary is placed on the floppy drive so it can be used later on to check the binaries of the system (just in case it gets trojaned).

The snapshot does not include the files under `/var/lib/dpkg/info` which includes the md5 hashes of installed packages (in files ended with `.md5sums`). You could copy this information along too, however you should notice:

- the md5sums provided by the Debian packages include all the files provided by them, which makes the database bigger (5 Mbs versus 600kbs in a Debian GNU/Linux system with graphical system and around 2.5 Gbs of software installed)
- not all Debian packages provide md5sums for the files installed since it is not (currently) mandated policy.

Once the snapshot is done you should make sure to set the medium read-only. You can then store it for backup or place it in the drive and use it to drive a cron check nightly comparing the original md5sums against those on the snapshot.

4.18 Other recommendations

4.18.1 Do not use software depending on svgalib

SVGAlib is very nice for console lovers like me, but in the past it has been proven several times that it is very insecure. Exploits against `zgv` were released, and it was simple to become root. Try to prevent using SVGAlib programs wherever possible.

Chapter 5

Securing services running on your system

Services can be secured in a running system in two ways:

- Making them only accessible in the access points (interfaces) they need to be in.
- Configuring them properly so that they can only be used by legitimate users in an authorised manner.

Restricting services so that they can only be accessed from a given place can be done by restricting access to them at the kernel (i.e. firewall) level, configure them to listen only on a given interface (some services might not provide this feature) or using some other methods, for example the linux vserver patch (for 2.4.16) can be used to force processes to use only one interface.

Regarding the services running from `inetd` (telnet, ftp, finger, pop3...) it is worth noting that inetd cannot be configured so that services only listen on a given interface. However, its substitute, the `xinetd` meta-daemon includes a `bind` just for this matter. See `xinetd.conf(5)`.

```
service nntp
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = news
        group           = news
        server          = /usr/bin/env
        server_args     = POSTING_OK=1 PATH=/usr/sbin/:/usr/bin:/sbin/:/bin
+/usr/sbin/snntpd logger -p news.info
        bind            = 127.0.0.1
}
```

The following sections detail how determined services can be configured properly depending on their intented use.

5.1 Securing ssh

If you are still running telnet instead of ssh, you should take a break from this manual and change this. Ssh should be used for all remote logins instead of telnet. In an age where it is easy to sniff internet traffic and get cleartext passwords, you should use only protocols which use cryptography. So, perform an `apt-get install ssh` on your system now.

Encourage all the users on your system to use ssh instead of telnet, or even better, uninstall telnet/telnetd. In addition you should avoid logging into the system using ssh as root and use alternative methods to become root instead, like `su` or `sudo`. Finally, the `sshd_config` file, in `/etc/ssh`, should be modified to increase security as well:

- `ListenAddress 192.168.0.1`

  Have ssh listen only on a given interface, just in case you have more than one (and do not want ssh available on it) or in the future add a new network card (and don't want ssh connections from it).

- `PermitRootLogin No`

  Try not to permit Root Login wherever possible. If anyone wants to become root via ssh, now two logins are needed and the root password cannot be brute forced via SSH.

- `Listen 666`

  Change the listen port, so the intruder cannot be completely sure whether a sshd daemon runs (be forewarned, this is security by obscurity).

- `PermitEmptyPasswords no`

  Empty passwords make a mockery of system security.

- `AllowUsers alex ref`

  Allow only certain users to have access via ssh to this machine.

- `AllowGroups wheel admin`

  Allow only certain group members to have access via ssh to this machine. AllowGroups and AllowUsers have equivalent directives for denying access to a machine. Not surprisingly they are called "DenyUsers" and "DenyGroups".

- `PasswordAuthentication yes`

  It is completely your choice what you want to do. It is more secure only to allow access to machine from users with ssh-keys placed in the ~/.ssh/authorized_keys file. If you want so, set this one to "no".

- Disable any forms of authentication you do not really need, if you do not use, for example `RhostsRSAAuthentication`, `HostbasedAuthenticatio`, `KerberosAuthentication` or `RhostsAuthentication` (for exaple) you should disable them, even if they are already by default (see the manpage `sshd_config(5)`).

As a final note, be aware that these directives are from a OpenSSH configuration file. Right now, there are three commonly used SSH daemons, ssh1, ssh2, and OpenSSH by the OpenBSD people. Ssh1 was the first ssh daemon available and it is still the most commonly used (there are rumors that there is even a windows port). Ssh2 has many advantages over ssh1 except it is released under an closed-source license. OpenSSH is completely free ssh daemon, which supports both ssh1 and ssh2. OpenSSH is the version installed on Debian when the package 'ssh' is chosen.

You can read more information on how to setup SSH with PAM support in the security mailing list archives (`http://lists.debian.org/debian-security/2001/debian-security-200111/msg00395.html`).

5.2 Securing Squid

Squid is one of the most popular proxy/cache server, and there are some security issues that should be taken into account. Squid's default configuration file denies all users requests. You should configure Squid to allow access to trusted users, hosts or networks defining an Access Control List on `/etc/squid.conf`, see the Squid User's Guide (`http://squid-docs.sourceforge.net/latest/html/book1.htm`) for more information about defining ACLs rules.

Also, if not properly configured, someone may relay a mail message through Squid, since the HTTP and SMTP protocols are designed similarly. Squid's default configuration file denies access to port 25. If you wish to allow connections to port 25 just add it to Safe_ports lists. However, this is *NOT* recommended.

Setting and configuring the proxy/cache server properly is only part of keeping your site secure. Another necessary task is to analyse Squid's logs to assure that all things are working as they should be working. There are some packages in Debian GNU/Linux that can help an administrator to do this. The following packages are available in woody (Debian 3.0):

- `calamaris` - Log analyzer for Squid or Oops proxy log files.
- `modlogan` - A modular logfile analyzer.
- `sarg` - Squid Analysis Report Generator.

FIXME: Add more information about security on Squid Accelerator Mode

5.3 Securing FTP

If you really have to use FTP (without wrapping it with sslwrap or inside a SSL or SSH tunnel), you should chroot ftp into the ftp users' home directory, so that the user is unable to see anything else than their own directory. Otherwise they could traverse your root filesystem just like if they had a shell. You can add the following line in your `proftpd.conf` in your global section to enable this chroot feature:

```
DefaultRoot ~
```

Restart proftpd by `/etc/init.d/proftpd restart` and check whether you can escape from your homedir now.

To prevent Proftp DoS attacks using ../../.., add the following line in `/etc/proftpd.conf`: `DenyFilter \*.*/`

Always remember that FTP sends login and authentication passwords in clear text (this is not an issue if you are providing an anonymous public service) and there are better alternatives in Debian for this. For example, `sftp` (provided by `ssh`). There are also free implementations of SSH for other operating systems: putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/) and cygwin (http://www.cygwin.com) for example.

However, if you still maintain the FTP server while making users access through SSH you migh encounter a typical problem. Users accessing Anonymous FTP servers inside SSH-secured systems is the might try to log in the *FTP server*. While the access will be refused, the password will nevertheless be sent through the net in clear form. To avoid that, ProFTPd developer TJ Saunders has created a patch that prevents users feeding the anonymous FTP server with valid SSH acounts. More information and patch available at: ProFTPD Patches (http://www.castaglia.org/proftpd/#Patches).

5.4 Securing access to the X Window System

Today, X terminals are used by more and more companies where one server is needed for a lot of workstations. This can be dangerous, because you need to allow the file server to connect to the the clients (X server from the X point of view. X switches the definition of client and server). If you follow the (very bad) suggestion of many docs, you type `xhost +` on your machine. This allows any X client to connect to your system. For slightly better security, you can use the command `xhost +hostname` instead to only allow access from specific hosts.

A much more secure solution, though, is to use ssh to tunnel X and encrypt the whole session. This is done automatically when you ssh to another machine. This has to be enabled in `/etc/ssh/ssh_config` by setting `X11Forwarding` to `yes`. In times of SSH, you should drop the xhost based access control completely.

For best security, if you do not need X access from other machines, is to switch off the binding on tcp port 6000 simply by typing:

```
$ startx -- -nolisten tcp
```

This is the default behavior in Xfree 4.1.0 (the Xserver provided in Debian 3.0). If you are running Xfree 3.3.6 (i.e. you have Debian 2.2 installed) you can edit `/etc/X11/xinit/xserverrcc` to have it something along the lines of:

```
#!/bin/sh
exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

If you are using XDM set `/etc/X11/xdm/Xservers` to: `:0 local /usr/bin/X11/X vt7 -dpi 100 -nolisten tcp`. If you are using Gdm make sure that the `-nolisten tcp` option is set in the `/etc/gdm/gdm.conf` (which is the default in Debian) such as this:

```
[server-Standard]
name=Standard Server
command=/usr/bin/X11/X -nolisten tcp
```

You can also set the default's system timeout for `xscreensaver` locks. Even if the user can override it, you should edit the `/etc/X11/app-defaults/XScreenSaver` configuration file and change the lock line:

```
*lock:                  False
```

(which is the default in Debian) to:

```
*lock:                  True
```

FIXME: add information on how to disable the screensavers which show the user desktop (which might have sensitive information).

Read more on X Window security in XWindow-User-HOWTO (`http://www.linuxdoc.org/HOWTO/XWindow-User-HOWTO.html`) (`/usr/share/doc/HOWTO/en-txt/XWindow-User-HOWTO.txt.gz`).

FIXME: Add info on thread of debian-security on how to change config files of XFree 3.3.6 to do this.

5.4.1 Check your display manager

If you only want to have a display manager installed for local usage (having a nice graphical login, that is), make sure the XDMCP (X Display Manager Control Protocol) stuff is disabled. In XDM you can do this with this line in `/etc/X11/xdm/xdm-config`:

```
DisplayManager.requestPort:     0
```

Normally, all display managers are configured not to start XDMCP services per default in Debian.

5.5 Securing printing access (The lpd and lprng issue)

Imagine, you arrive at work, and the printer is spitting out endless amounts of paper because someone is DoSing your line printer daemon. Nasty, isn't it?

In any unix printing architecture, there has to be a way to get the client's data to the host's print server. In traditional `lpr` and `lp`, the client command copies or symlinks the data into the spool directory (which is why these programs is usually SUID or SGID).

In order to avoid any issues you should keep your printer servers specially secure. This means you need to configure your printer service so it will only allow connections from a set of trusted servers. In order to do this, add the servers you want to allow printing to your `/etc/hosts.lpd`.

However, even if you do this, the `lpr` daemon accepts incoming connections on port 515 of any interface. You should consider firewalling connections from networks/hosts which are not allowed printing (the `lpr` daemon cannot be limited to listen only on a given IP address).

`Lprng` should be preferred over `lpr` since it can be configured to do IP access control. And you can specify which interface to bind to (although somewhat weirdly).

If you are using a printer in your system, but only locally, you will not want to share this service over a network. You can consider using other printing systems, like the one provided by `cups` or PDQ (`http://pdq.sourceforge.net/`) which is based on user permissions of the `/dev/lp0` device.

In `cups`, the print data is transferred to the server via the http protocol. This means the client program doesn't need any special privileges, but does require that the server be listening on a port somewhere.

However, if you want to use `cups`, but only locally, you can configure it to bind to the loopback interface by changing `/etc/cups/cupsd.conf`:

```
Listen 127.0.0.1:631
```

There are many other security options like allowing or denying networks and hosts in this config file. However, if you do not need them you might be better off just limiting the listening port. `Cups` also serves documentation through the HTTP port, if you do not want to disclose potential useful information to outside attackers (and the port is open) add also:

```
<Location />
  Order Deny,Allow
   Deny From All
    Allow From 127.0.0.1
</Locationi>
```

This configuration file can be modified to add some more features including SSL/TLS certificates and crypto. The manuals are available at http://localhost:631/ or at cups.org.

FIXME: Add more content (the article on Amateur Fortress Building (http://www.rootprompt.org) provides some very interesting views).

FIXME: Check if PDG is available in Debian, and if so, suggest this as the preferred printing system.

FIXME: Check if Farmer/Wietse has a replacement for printer daemon and if it's available in Debian.

5.6 Securing the mail daemon

If your server is not a mailing system, you do not really need to have a mail daemon listening for incoming connections, but you might want local mail delivered in order, for example, to receive mail for the root user from any alert systems you have in place.

To do this in a Debian system, you will have to remove the smtp daemon from inetd:

```
$ update-inetd --disable smtp
```

and configure the mailer daemon to only listen on the loopback interface. In exim (the default MTA) you can do this by editing the file `/etc/exim.conf` and adding the following line:

```
local_interfaces = "127.0.0.1"
```

Restart both daemons (inetd and exim) and you will have exim listening on the 127.0.0.1:25 socket only. Be careful, and first disable inetd, otherwise, exim will not start since the inetd daemon is already handling incoming connections.

For postfix edit `/etc/postfix/main.conf`:

```
inet_interfaces = localhost
```

If you only want local mail, this approach is better than tcp-wrapping the mailer daemon or adding firewalling rules to limit anybody accessing it. However, if you do need it to listen on other interfaces, you might consider launching it from inetd and adding a tcp wrapper so incoming connections are checked against `/etc/hosts.allow` and `/etc/hosts.deny`. Also, you will be aware of when an unauthorized access is attempted against your mailer daemon, if you set up proper logging for any of the methods above.

In any case, to reject mail relay attempts at the SMTP level, you can change `/etc/exim/exim.conf` to include:

```
receiver_verify = true
```

Even if you mail server will not relay the message, this kind of configuration is needed for the relay tester at http://www.abuse.net/relay.html to determine that your server is *not* relay capable.

5.7 Receiving mail securely

Reading/receiving mail is the most common cleartext protocol. If you use either POP3 or IMAP to get your mail, you send your cleartext password across the net, so almost anyone can read your mail from now on. Instead, use SSL (Secure Sockets Layer) to receive your mail. The other alternative is ssh, if you have a shell account on the box which acts as your POP or IMAP server. Here is a basic fetchmailrc to demonstrate this:

```
poll my-imap-mailserver.org via "localhost"
  with proto IMAP port 1236
      user "ref" there with password "hackme" is alex here warnings 3600
    folders
      .Mail/debian
    preconnect 'ssh -f -P -C -L 1236:my-imap-mailserver.org:143 -l ref
     my-imap-mailserver.org sleep 15 </dev/null > /dev/null'
```

The preconnect is the important line. It fires up a ssh session and creates the necessary tunnel, which automatically forwards connections to localhost port 1236 to the IMAP mail server, but encrypted. Another possibility would be to use fetchmail with the ssl feature.

If you want to provide encrypted mail services like POP and IMAP, `apt-get install stunnel` and start your daemons this way:

```
stunnel -p /etc/ssl/certs/stunnel.pem -d pop3s -l /usr/sbin/popd
```

This command wraps the provided daemon (-l) to the port (-d) and uses the specified ssl cert (-p).

5.8 Securing BIND

There are different issues that can be tackled in order to secure the Domain server daemon, which are similar to the ones considered when securing any given service:

- configure the daemon itself properly so it cannot be misused from the outside. This includes limiting possible queries from clients: zone transfers and recursive queries.
- limit the access of the daemon to the server itself so if it is used to break in, the damage to the system is limited. This includes running the daemon as a non-privileged user and chrooting it.

You should restrict some of the information that is served from the DNS server to outside clients so that it cannot be used to retrieve valuable information from your organization that you do not want to give away. This includes adding the following options: *allow-transfer*,

allow-query, *allow-recursive* and *version*. You can either limit this on the global section (so it applies to all the zones served) or on a per-zone basis. This information is documented on the `bind-doc` package, read more on this on `/usr/share/doc/bind/html/index.html` once the package is installed.

Imagine that your server is connected to the Internet and to your internal (your internal IP is 192.168.1.2) network (a basic multi-homed server), you do not want to give any service to the Internet and you just want to enable DNS lookups from your internal hosts. You could restrict it by including in `/etc/bind/named.conf`:

```
options {
     allow-query { 192.168.1/24; } ;
     allow-transfer { none; } ;
     allow-recursive { 192.168.1/24; } ;
     listen-on { 192.168.1.2; } ;
     forward { only; } ;
     forwarders { A.B.C.D; } ;
};
```

The *listen-on* option makes the DNS bind to only the interface that has the internal address, but, even if this interface is the same as the interface that connects to the Internet (if you are using NAT, for example), queries will only be accepted if coming from your internal hosts. If the system has multiple interfaces and the *listen-on* is not present, only internal users could query, but, since the port would be accessible to outside attackers, they could try to crash (or exploit buffer overflow attacks) on the DNS server. You could even make it listen only on 127.0.0.1 if you are not giving DNS service for any other systems than yourself.

The version.bind record in the chaos class contains the version of the of the currently running bind process. This information is often used by automated scanners and malicious individuals who wish to determine if one's bind is vulnerable to a specific attack. By providing false or no information in the version.bind record, one limits the probability that one's server will be attacked based on its publicized version. To provide your own version, use the *version* directive in the following manner:

```
options {
 ... various options here ...
 version "Not available.";
};
```

Changing the version.bind record does not provide actual protection against attacks, but it might be considered a useful safeguard.

A sample `named.conf` configuration file might be the following:

```
acl internal {
        127.0.0.1/32;           // localhost
```

```
        10.0.0.0/8;             // internal
        aa.bb.cc.dd;            // eth0 IP
};

acl friendly {
        ee.ff.gg.hh;            // slave DNS
        aa.bb.cc.dd;            // eth0 IP
        127.0.0.1/32;           // localhost
        10.0.0.0/8;             // internal
};

options {
        directory "/var/cache/bind";
        allow-query { internal; };
        allow-recursive { internal; };
        allow-transfer { none; };
};
// From here to the mysite.bogus zone
// is basically unmodified from the debian default
logging {
        category lame-servers { null; };
        category cname { null; };
};

zone "." {
        type hint;
        file "/etc/bind/db.root";
};

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
```

```
        file "/etc/bind/db.255";
};

// zones I added myself
zone "mysite.bogus" {
        type master;
        file "/etc/bind/named.mysite";
        allow-query { any; };
        allow-transfer { friendly; };
};
```

Please (again) check the Bug Tracking System regarding Bind, specifically Bug #94760 (regarding ACLs on zone transfers) (http://bugs.debian.org/94760). Feel free to contribute to the bug report if you think you can add useful information.

5.8.1 Changing BIND's user

Regarding limiting BIND's privileges you must be aware that if a non-root user runs BIND, then BIND cannot detect new interfaces automatically. For example, if you stick a PCMCIA card into your laptop. Check the README.Debian file in your named documentation (`/usr/share/doc/bind/README.Debian`) directory for more information about this issue. There have been many recent security problems concerning BIND, so switching the user is useful when possible. We will detail here the steps needed in order to do this, however, if you want to do this in an automatic way you might try the script provided in 'Sample script to change the default Bind installation.' on page 141.

To run BIND under a different user, first create a separate user and group for it (it is *not* a good idea to use nobody or nogroup for every service not running as root). In this example, the user and group `named` will be used. You can do this by entering:

```
addgroup named
adduser --system --home /home/named --no-create-home --ingroup named \
      --disabled-password --disabled-login named
```

Notice that the user `named` will be quite restricted. If you want, for whatever reason, to have a less restrictive setup use:

```
adduser --system --ingroup named named
```

Now edit `/etc/init.d/bind` with your favorite editor and change the line beginning with

```
start-stop-daemon --start
```

to

```
start-stop-daemon --start --quiet --exec /usr/sbin/named -- -g named -u named
```

Also, in order to avoid running anything as root, change the `reload` line commenting out:

```
reload)
       /usr/sbin/ndc reload
```

And change it to:

```
reload)
        $0 stop
        sleep 1
        $0 start
```

Note: Depending on your Debian version you might have to change the `restart` line too. This was fixed in Debian's bind version `1:8.3.1-2`.

All you need to do now is to restart bind via '/etc/init.d/bind restart', and then check your syslog for two entries like this:

```
Sep  4 15:11:08 nexus named[13439]: group = named
Sep  4 15:11:08 nexus named[13439]: user = named
```

Voilá! Your named now *does not* run as root. If you want to read more information on why BIND does not run as non-root user on Debian systems, please check the Bug Tracking System regarding Bind, specifically Bug #50013: bind should not run as root (http://bugs.debian.org/50013) and Bug #132582: Default install is potentially insecure (http://bugs.debian.org/132582), Bug #53550 (http://bugs.debian.org/53550), Bug #128120 (http://bugs.debian.org/52745), and Bug #128120 (http://bugs.debian.org/128129). Feel free to contribute to the bug reports if you think you can add useful information.

5.8.2 Chrooting the name server

To achieve maximum BIND security, now build a chroot jail (see 'Using chroot' on page 49) around your daemon. There is an easy way to do this: the `-t` option (see the `named(8)` manpage). This will make Bind chroot itself into the given directory without you needing to setup a chroot jail and worry about dynamic libraries. The only files there needs to be in these chroot jail:

```
dev/null
etc/bind/       - should hold named.conf and all the server zones
sbin/named-xfer - if you do name transfers
```

```
var/run/named/  - should hold the pid and the name server cache (if
                  any) this directory needs to be writable by named
                  user
var/log/named   - if you setup logging to a file, needs to be writable
                  for the named user
dev/log         - syslogd should be listening here if named is configure to
                  log through it
```

In order for your Bind daemon to work properly it needs permission in the named files. This is an easy task since the configuration files are always at `/etc/named/`. Take in account that it only needs read-only access to the zone files, unless it is a secondary or cache name server. If this is your case you will have to give read-write permissions to the necessary zones (so that zone transfers from the primary server work).

Also, you can find more information regarding Bind chrooting in the Chroot-BIND-HOWTO (http://www.linuxdoc.org/HOWTO/Chroot-BIND-HOWTO.html) (regarding Bind 9) and Chroot-BIND8-HOWTO (http://www.linuxdoc.org/HOWTO/Chroot-BIND8-HOWTO.html) (regarding Bind 8). This same documents should be available through the installation of the `doc-linux-text` (text version) or `doc-linux-html` (html version). Another useful document is http://www.psionic.com/papers/dns/dns-linux.

If you are setting up a full chroot jail (i.e. not just `-t`) for Bind 8.2.3 in Debian (potato), make sure you have the following files in it:

```
dev/log - syslogd should be listening here
dev/null
etc/bind/named.conf
etc/localtime
etc/group - with only a single line: "named:x:GID:"
etc/ld.so.cache - generated with ldconfig
lib/ld-2.1.3.so
lib/libc-2.1.3.so
lib/ld-linux.so.2 - symlinked to ld-2.1.3.so
lib/libc.so.6 - symlinked to libc-2.1.3.so
sbin/ldconfig - may be deleted after setting up the chroot
sbin/named-xfer - if you do name transfers
var/run/
```

And modify also `syslogd` listen on $CHROOT/dev/log so the named server can write syslog entries into the local system log.

If you want to avoid problems with dynamic libraries, you can compile bind statically. You can use `apt-get` for this, with the `source` option. It can even download the packages you need to properly compile it. You would need to do someting similar to:

```
$ apt-get --download-only source bind build-dep bind
```

```
$ cd bind-8.2.5-2
(edit the Makefile.in so CFLAGS includes the '-static' option
before the @CFLAGS@ definition substituted by autoconf)
$ dpkg-buildpackage -rfakeroot
$ cd ..
$ dpkg  -i bind-8.2.5-2*deb
```

After installation, you will need to move around the files to the chroot jail [1] you can keep the `init.d` scripts in `/etc/init.d` so that the system will automatically start the name server, but edit them to add `--chroot /location_of_chroot` in the calls to `start-stop-daemon` in those scripts.

FIXME, merge info from http://people.debian.org/~pzn/howto/chroot-bind.sh.txt, http://people.pdxlinux.org/~karlheg/ (Bind9 on Debian), http://www.cryptio.net/~ferlatte/config/ (Debian-specific), http://www.psionic.com/papers/whitep01.html, http://csrc.nist.gov/fasp/FASPDocs/NISTSecuringDNS.htm and http://www.acmebw.com/papers/securing.pdf.

5.9 Securing Apache

FIXME: Add content.

You can limit access to the Apache server if you only want to use it internally (for testing purposes, to access the `doc-central` archive, etc..) and do not want outsiders to access it. To do this use the `Listen` or `BindAddress` directivdes in `/etc/apache/http.conf`.

Using Listen:

```
Listen 127.0.0.1:80
```

Using BindAddress:

```
BindAddress 127.0.0.1
```

Then restart apache with `/etc/init.d/apache restart` and you will see that it is only listening on the loopback interface.

In any case, if you are not using all the functionality provided by Apache, you might want to take a look at other web servers provided in Debian like `dhttpd`.

The Apache Documentation (http://httpd.apache.org/docs/misc/security_tips.html) provides information regarding security measures to be taken on Apache webserver (this same information is provided in Debian by the `apache-doc` package).

[1] unless you use the `instdir` option when calling `dpkg` but then the chroot jail might be a little more complex

5.10 Securing finger

If you want to run the finger service first ask yourself if you need to do so. If you do, you will find out that Debian provides many finger daemons (output from `apt-cache search fingerd`):

- cfingerd - Configurable finger daemon
- efingerd - Another finger daemon for unix capable of fine-tuning your output.
- ffingerd - a secure finger daemon
- fingerd - Remote user information server.
- BSD-like finger daemon with qmail support.

`ffingerd` is the recommended finger daemon if you are going to use it for a public service. In any case, you are encouraged to, when setting it up through inetd, xinetd or tcpserver to: limit the number of processes that will be running at the same time, limit access to the finger daemon from a given number of hosts (using tcp wrappers) and having it only listening to the interface you need it to be in.

5.11 General chroot and suid paranoia

It is probably fair to say that the complexity of BIND is the reason why it has been exposed to a lot of attacks in recent years. (see 'Securing BIND' on page 66)

Other programs with complex features and a large installed user base include Sendmail and some ftp daemons (e.g. WUftpd). (Of course, a program with no features and no satisfied users can be just as insecure, besides being useless.)

Anyway, if you run any of these, consider similar arrangements for them — revoking root privileges, running in a chroot jail — or replacing them with a more secure equivalent.

5.12 General cleartext password paranoia

You should try to avoid any network service which sends and receives passwords in cleartext over a net like FTP/Telnet/NIS/RPC. The author recommends the use of ssh instead of telnet and ftp to everybody.

Keep in mind that migrating from telnet to ssh, but using other cleartext protocols does not increase your security in ANY way! Best would be to remove ftp, telnet, pop, imap, http and to supersede them with their respective encrypted services. You should consider moving from these services to their SSL versions, ftp-ssl, telnet-ssl, pop-ssl, https ...

Most of these above listed hints apply to every Unix system (you will find them if reading any other hardening-related document related to Linux and other Unices).

5.13 Disabling NIS

You should not use NIS, the Network Information Service, if it is possible, because it allows password sharing. This can be highly insecure if your setup is broken.

If you need password sharing between machines, you might want to consider using other alternatives. For example, you can set a LDAP server and configure PAM on your system in order to contact the LDAP server for user authentication. You can find a detailed setup in the LDAP-HOWTO (`http://www.linuxdoc.org/HOWTO/LDAP-HOWTO.html`) (`/usr/share/doc/HOWTO/en-txt/LDAP-HOWTO.txt.gz`).

Read more on NIS security in NIS-HOWTO (`http://www.linuxdoc.org/HOWTO/NIS-HOWTO.html`) (`/usr/share/doc/HOWTO/en-txt/NIS-HOWTO.txt.gz`).

FIXME (jfs): Add info on how to setup this in Debian

5.14 Disabling RPC services

You should disable RPC wherever possible, that is, when you do not need it. [2] Many security holes for both the portmapper service and RPC-based services are known and could be easily exploited. On the other hand NFS services are quite important in some networks, so find a balance of security and usability in your network. Some of the DDoS (distributed denial of service) attacks use rpc exploits to get into the system and act as a so called agent/handler. Read more on NFS security in NFS-HOWTO (`http://www.linuxdoc.org/HOWTO/NFS-HOWTO.html`) (`/usr/share/doc/HOWTO/en-txt/NFS-HOWTO.txt.gz`).

Disabling portmap is quite simple. There are different methods. The simplest one in a Debian 3.0 system is to do uninstall the `portmap` package. If you are running another version you will have to disable the service as seen in 'Disabling daemon services' on page 24, this is due to the program being a part of the `net-base` package (which cannot be de-installed without breaking the system).

This in fact removes every symlink relating to portmap in `/etc/rc${runlevel}.d/`, which is something you could also do manually. Another possibility is to `chmod 644 /etc/init.d/portmap`, but that gives an error message when booting. You can also strip off the `start-stop-daemon` part in `/etc/init.d/portmap` shell script.

5.15 Adding firewall capabilities

The Debian GNU/Linux operating system has the built-in capabilities provided by the Linux kernel. This means that if you install a potato (Debian 2.2 release) system (default kernel is 2.2) you will have `ipchains` firewalling available in the kernel, you need to install the `ipchains` which will be surely (due to its priority) be already installed. If you install a woody (Debian 3.0

[2] You only probably need it if using NFS (Network FileSystem), NIS (Network Information System) or some other RPC-based service.

release) system (default kernel is 2.4) you will have `iptables` (neftfilter) firewalling available. The main different between `ipchains` and `iptables` is that the later is based on *stateful packet inspection* which provides for more secure (and easier to build) filtering configurations.

5.15.1 Firewalling the local system

You can use firewall rules as a way to secure the access to your local system and, even, to limit the outbound communications made by it. Firewall rules can be used also to protect processes that cannot be properly configured *not* to provide services to some networks, IP addresses, etc..

However, this step is presented last in this manual basically because it is *much* better to not depend solely on firewalling capabilities in order to protect a given system. Security in a system is made up of layers, firewalling should be the last to include, once all services have been hardened. You can easily imagine a setup in which the system is solely protected by a built-in firewall and an administrator blissfully removes the firewall rules for whatever reason (problems with the setup, annoyance, human error...), this system would be wide open to an attack if there were no other hardening in the system to protect from it.

On the other hand, having firewall rules on the local system also prevents some bad things from happening. Even if the services provided are configured securely, a firewall can protect from misconfigurations or from fresh installed services that have not yet been properly configured. Also, a tight configuration will prevent trojans *calling home* from working unless the firewalling code is removed. Note that an intruder does *not* need superuser access to install a trojan locally that could be remotely controlled (since binding on ports is allowed if they are not priviledge ports and capabilities have not been removed).

Thus, a proper firewall setup would be one with a default deny policy, that is:

- incoming connections are allowed only to local services by allowed machines.
- outgoing connections are only allowed to services used by your system (DNS, web browsing, pop, email....) [3]
- the forward rule denies everything (unless you are protecting other systems, see below).
- all other incoming or outgoing connections are denied.

5.15.2 Using a firewall to protect other systems

A Debian firewall can also be installed in order to protect, with filtering rules, access to systems *behind* it, limiting their exposure to the Internet. The firewall can be configured to prevent systems outside the local network to access services (ports) that are not public. For example, on a mail server, only port 25 (where the mail service is being given) needs to be accesible from

[3] Unlike personal firewalls in other operating systems, Debian GNU/Linux does not (yet) provide firewall generation interfaces that can make rules limiting them per process or user. However, the iptables code can be configured to do this (see the owner module in the `iptables(8)` manpage)

the outside. A firewall can be configured to, even if there are other services besides the public ones, throw away packets (this is known as *filtering*) directed towards them.

You can even setup a Debian GNU/Linux box as a bridge firewall, i.e. a filtering firewall completely transparent to the network that lacks an IP address and thus cannot be attacked directly. Depending on the kernel you have installed, you might need to might to install the bridge firewall patch and then go to *802.1d Ethernet Bridging* when configuring the kernel and a new option *netfilter (firewalling) suport*. See the 'Setting up a bridge firewall' on page 135 for more information on how to setup this in a Debian GNU/Linux system).

5.15.3 Configuring the firewall

Of course, the configuration of the firewall is always system and network dependant. An administrator must know beforehand what is the network layout and the systems he wants to protect, the services that need to be accessed, and wether or not other network considerations (like NAT or routing) need to be taken into account. Be careful when configuring your firewall, as Laurence J. Lane says in the `iptables` package:

The tools can easily be misused, causing enormous amounts of grief by completely cripple network access to a computer system. It is not terribly uncommon for a remote system administrator to accidentally lock himself out of a system hundreds or thousands of miles away. One can even manage to lock himself out of a computer who's keyboard is under his fingers. Please, use due caution.

Remember this: just installing the `iptables` (or the older firewalling code) does not give you any protection, just provides the software. In order to have a firewall you need to *configure* it!

If you do not know much about firewalling, read the Firewalling-HOWTO that can be found in `doc-linux-text` package (other document formats also available). See 'Be aware of general security problems' on page 17 for more (general) pointers.

Doing it the Debian way

If you are using Debian 3.0, you will notice that you have the `iptables` package installed. This is the support for the 2.4.4+ kernels netfilter implementation. Since just after installation the system cannot *know* any firewall rules (firewall rules are too system-specific) you have to enable iptables. However, the scripts have been configured so that the administrator can setup firewall rules and then have the init scripts *learn* them and use them always as the setup for the firewall.

In order to do so you must:

- Configure the package so that it starts with the system. On newer versions (since 1.2.6a-1) this is asked for when the package is installed. You can configure it afterwards with `dpkg-reconfigure -plow iptables`. *Note*: on older versions this was done by editing `/etc/default/iptables` so that the variable `enable_iptables_initd` was set to *true*.

- create a firewall setup using iptables, you can use the command line (see `iptables(8)`) or some of the tools provided by the Debian firewall packages (see 'Using Firewall packages' on the current page). You need to create one set of firewall rules to be used when the firewall is in *active* state and another to be used when the firewall is in *inactive* state (these can be just empty rules).
- save the rules you created using `/etc/init.d/iptables save_active` and `/etc/init.d/iptables save_active` by running these scripts with the firewall rules you want enabled.

Once this is done your firewall setup is saved in the `/var/lib/iptables/` directory and will be executed when the system boots (or when running the initd script with *start* and *stop* arguments). Please notice that the default Debian setups starts the firewalling code in the multiuser runlevels (2 to 5) pretty soon (10). Also, it is stopped in singleuser runlevel (1), change this if it does not mach your local policy.

If you do not have a clue on how to setup your firewall rules manually consult the *Packet Filtering HOWTO* and *NAT HOWTO* provided by `iptables` for offline reading at `/usr/share/doc/iptables/html/`. Also, the configuration file `/etc/default/iptables` provides some more information about the issues regarding this package.

Using Firewall packages

Setting up manually a firewall can be complicated for novice (and sometimes even expert) administrators. However, the free software community has created a number of tools that can be used to easily configure a local firewall. Be forewarned that some of this tools are oriented more towards local-only protection (also known as *personal firewall*) and some are more versatile and can be used to configure complex rules to protect whole networks.

Some software that can be used to setup firewall rules in a Debian system is:

- `firestarter`
- `knetfilter`
- `fwbuilder`
- `shorewall`
- `mason`, which can propose firewall rules based on the network traffic your system "sees".
- `bastille` (among the hardening steps that can make new versions of bastille is the possibility of adding firewall rules to the system to be executed on startup)
- `ferm`
- `fwctl`
- `easyfw`

- `firewall-easy`
- `ipac-ng`
- `gfcc`
- `lokkit` or `gnome-lokkit`

The last packages: gfcc,firestarter and knetfilter are administration GUIs using either GNOME (first two) or KDE (last one) which are much more user-oriented (i.e. for home users) than the other packages in the list which might be more administrator-oriented.

Be forewarned that some of the packages outlined previousline will probably introduce firewalling scripts to be run when the system boots, this will undoubtedly conflict with the common setup (if configured) and you might undesireds effects. Usually, the firewalling script that runs last will be the one that configures the system (which might not be what you pretend). Consult the package documentation and use either one of these setups. Generally, other programs that help you setup the firewalling rules can tweak othe conriguration files.

FIXME: Add more info regarding this packages

FIXME: Check Information on Debian firewalling and what/how does it change from other distributions.

FIXME: Where should the custom firewalling code be enabled (common FAQ in debian-firewall?)

Chapter 6

Automatic hardening of Debian systems

After reading through all the information in the previous chapters you might be wondering "I have to do quite a lot of things in order to harden my system, couldn't this things be automated?". The question is yes, but be careful with automated tools. Some people believe, that a hardening tool does not eliminate the need for good administration. So do not be fooled to think that you can automate all the process and will fix all the related issues. Security is an ever-ongoing process in which the administrator must participate and cannot just stand away and let the tools do all the work since no single tool can cope: with all the possible security policy implementations, all the attacks and all the environments.

Since woody (Debian 3.0) there are two specific packages that are useful for security hardening. The `harden` which takes an approach based on the package dependencies to quickly install valuable security packages and remove those with flaws, configuration of the packages must be done by the administrator. The `bastille` that implements a given security policy on the local system based on previous configuration by the administrator (the building of the configuration can be a guided process done with simple yes/no questions).

6.1 Harden

The `harden` package tries to make it more easy to install and administer hosts that need good security . This package should be used by people that want some quick help to enhance the security of the system. To do this it conflicts with packages with known flaws, including (but not limited to): known security bugs (like buffer overflows), use of plaintext passwords, lack of access control, etc. It also automatically installs some tools that should enhance security in some way: intrusion detection tools, security analysis tools, etc. Harden installs the following *virtual* packages (i.e. no contents, just dependencies on others):

- `harden-tools`: tools to enhance system security (integrity checkers, intrusion detection, kernel patches...)

- `harden-doc`: provides this same manual and other security-related documentation packages.
- `harden-environment`: helps configure a hardened environment (currently empty).
- `harden-servers`: removes servers considered insecure for some reason.
- `harden-clients`: removes clients considered insecure for some reason.
- `harden-remoteflaws`: removes packages with known security holes that could be used by a remote attacker to compromise the system (uses versioned *Conflicts:*).
- `harden-localflaws`: removes packages with known security holes that could be used by a local attacker to compromise the system (uses versioned *Conflicts:*).
- `harden-remoteaudit`: tools to remotely audit a system.

Be careful because if you have software you need (and which you do not wish to uninstall for some reason) and it conflicts with some of the packages above you might not be able to fully use `harden`. The harden packages do not (directly) do a thing. They do have, however, intentional package conflicts with known non-secure packages. This way, the Debian packaging system will not approve the installation of these packages. For example, when you try to install a telnet daemon with `harden-servers`, apt will say:

```
# apt-get install telnetd
The following packages will be REMOVED:
 harden-servers
The following NEW packages will be installed:
telnetd
Do you want to continue (Y/n)
```

This should set off some warnings in the administrator head, who should reconsider his actions.

6.2 Bastille Linux

Bastille Linux (`http://www.bastille-linux.org`) is an automatic hardening tool originally oriented towards the RedHat and Mandrake Linux distributions. However, the `bastille` package provided in Debian (since woody) is patched in order to provide the same functionality for the Debian GNU/Linux system.

Bastille can be used with different frontends (all are documented in their own manpage in the Debian package) which enables the administrator to:

- Answer questions step by step regarding the desired security of your system (using `InteractiveBastille(8)`)

- Use a default setting for security (amongst three: Lax, Moderate or Paranoia) in a given setup (server or workstation) and let Bastille decide which security policy to implement (using `BastilleChooser(8)`)
- Take a predefined configuration file (could be provided by Bastille or made by the administrator) and implement a given security policy (using `AutomatedBastille(8)`)

Chapter 7

Package signing in Debian

This chapter could also be titled "how to upgrade/update safely your Debian GNU/Linux system" and it deserves its own chapter basically because it will not fit with any other chapter.

As of today (december 2001) Debian does not provide signed packages for the distribution and the woody release (3.0) does not integrate that feature. There is a solution for signed packages which will be, hopefully, provided in the next release (sarge).

7.1 The proposed scheme for package signature checks

The current (unimplemented) scheme for package signature checking using apt is:

- the Release file includes the md5sum of Packages.gz (which contains the md5sums of packages) and will be signed. The signature is one of a trusted source.
- This signed Release file is downloaded by 'apt-get update' and stored in the HD along with Packages.gz.
- When a package is going to be installed, it is first downloaded, then the md5sum is generated.
- The signed Release file is checked (signature ok) and it extracts from it the md5sum for the Packages.gz file, the Packages.gz checksum is generated and (if ok) the md5sum of the downloaded package is extracted from it.
- If the md5sum from the downloaded package is the same as the one in the Packages.gz file the package will be installed otherwise the administrator will be alerted and the package will be left in cache (so the administrator can decide wether to install it or not). If the package is not in the Packages.gz and the administrator has configured the system to only install checked packages it will not be installed either.

By following the chain of MD5 sums `apt` is capable of verifying that a package originates from a a specific release. This is less flexible than signing each package one by one, but can be combined with that scheme too (see below).

Package signing has been discussed in Debian for quite some time, for more information you can read: http://www.debian.org/News/weekly/2001/8/ and http://www.debian.org/News/weekly/2000/11/.

7.2 Alternative per-package signing scheme

The additional scheme of signing each and every packages allows packages to be checked when they are no longer referenced by an existing Packages file, and also third-party packages where no Packages ever existed for them can be also used in Debian but will not be default scheme.

This package signing scheme can be implemented using `debsig-verify`and `debsigs`. These two packages can sign and verify embeded signatures in the .deb itself. Debian already has the capability to do this now, but implementing the policy and tools won't be started until after woody releases (so as not to slow down its release cycle).

NOTE: Currently `/etc/dpkg/dpkg.cfg` ships with "no-debsig" as per default.

7.3 Checking packages releases

In case you want to add now the additional security checks you can use the script below, provided by Anthony Thown. This script can automatically do some new security checks to allow the user to be sure that the software s/he's downloading matches the software Debian's distributing. This stops Debian developers from hacking into someone's system without the accountability provided by uploading to the main archive, or mirrors mirroring something almost, but not quite like Debian, or mirrors providing out of date copies of unstable with known security problems.

This sample code, renamed as `apt-release-check`, should be used in the following way:

```
# apt-get update
# apt-release-check
(...results...)
# apt-get dist-upgrade
```

First you need to:

- get the keys the archive software uses to to sign Release files, http://ftp-master.debian.org/ziyi_key_2002.asc and add them to `~/.gnupg/trustedkeys.gpg` (which is what `gpgv` uses by default)

- remove any `/etc/apt/sources.list` lines that don't use the normal "dists" structure, or change the script so that it works with them.
- be prepared to ignore the fact that Debian security updates don't have signed Release files, and that Sources files don't have appropriate checksums in the Release file (yet).
- be prepared to check that the appropriate sources are signed by the appropriate keys.

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

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

>OK
>MISSING
>NOCHECK
>BAD

arch=‘dpkg --print-installation-architecture‘

am_root () {
        [ ‘id -u‘ -eq 0 ]
}

get_md5sumsize () {
        cat "$1" | awk ’/^MD5Sum:/,/^SHA1:/’ |
          MYARG="$2" perl -ne ’@f = split /\s+/; if ($f[3] eq $ENV{"MYARG"})
}
checkit () {
        local FILE="$1"
        local LOOKUP="$2"

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

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

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

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

```
fi
echo "Source: ${ty} ${url} ${dist} ${comps}"

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

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

        if [ "${dist%%/*}" != "$suitline" -a "${dist%%/*}" != "$codel
                echo "  * WARNING: asked for $dist, got $suitline/$co
        fi

        wget -q -O Release.gpg "${url}/dists/${dist}/Release.gpg"
        sigline="`gpgv --status-fd 3 Release.gpg Release 3>&1 >/dev/n
        if [ "$sigline" ]; then
                echo "  o Signed by: $sigline"
        else
                echo "  * NO VALID SIGNATURE"
                >Release
        fi
fi
okaycomps=""
for comp in $comps; do
        if [ "$ty" = "deb" ]; then
                X=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/
                Y=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/
                if [ "$X $Y" = "OK OK" ]; then
                        okaycomps="$okaycomps $comp"
                else
                        echo "  * PROBLEMS WITH $comp ($X, $Y)"
                fi
        elif [ "$ty" = "deb-src" ]; then
                X=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/
                Y=$(checkit "`echo "${baseurl}/dists/${dist}/${comp}/
```

```
                        if [ "$X $Y" = "OK OK" ]; then
                                okaycomps="$okaycomps $comp"
                        else
                                echo "  * PROBLEMS WITH component $comp ($X,
                        fi
                fi
        done
        [ "$okaycomps" = "" ] || echo "  o Okay:$okaycomps"
        echo
  done

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

allokay=true

cd /tmp/apt-release-check
diff <(cat BAD MISSING NOCHECK OK | sort) <(cd /var/lib/apt/lists && find . -

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

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

```
    echo
    sed 's/^/    /' < BAD
    echo
fi

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

if grep -q ^ NOCHECK; then
    allokay=false
    (echo "The contents of the following files in /var/lib/apt/lists could no
    echo "be validated due to the lack of a signed Release file, or the lack"
    echo "of an appropriate entry in a signed Release file. This probably"
    echo "means that the maintainers of these sources are slack, but may mean
    echo "these sources are being actively used to distribute trojans."
    if am_root; then
        echo "The files have been renamed to have the extension .FAILED and"
        echo "will be ignored by apt."
        cat NOCHECK | while read a; do
            mv /var/lib/apt/lists/$a /var/lib/apt/lists/${a}.FAILED
        done
    fi) | fmt
    echo
    sed 's/^/    /' < NOCHECK
    echo
fi

if $allokay; then
    echo 'Everything seems okay!'
    echo
fi

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

Chapter 8

Security tools in Debian

FIXME: More content needed.

Debian provides also a number of security tools that can make a Debian box suited for security testing purposes. Some of them are provided when installing the `harden-remoteaudit` package

8.1 Remote vulnerability assesment tools

The tools provided by Debian to perform remote vulnerability assesment are:

- nessus
- raccess
- whisker
- nikto (whisker's replacement)
- bass (non-free)
- satan (non-free)

By far, the most complete and up-to-date tools is `nessus` which is composed of a client (`nessus`) used as a GUI and a server (`nessusd`) which launches the programmed attacks. Nessus includes remote vulnerabilities for quite a number of systems including network appliances, ftp servers, www servers, etc. The latest releases are able even to parse a web site and try to discover which interactive pages are available which could be attacked. There are also Java and Win32 clients (not included in Debian) which can be used to contact the management server.

`Whisker` is a web-only vulnerability assessment scanner including anti-IDS tactics (most of which are not *anti-IDS* anymore). It is one of the best cgi-scanners available, being able to

detect WWW servers and launch only a given set of attacks against it. The database used for scanning can be easily modified to provide for new information.

`Bass` (Bulk Auditing Security Scanner) and `Satan` (Security Auditing Tool for Analysing Networks) must be thought of more like "proof of concept" programs than as tools to be used while performing audits. Both are quite ancient and are not kept up-to-date. However, SATAN was the first tool to provide vulnerability assesment in a simple (GUI) way and Bass is still a very high-perfomance assesment tool.

8.2 Network scanner tools

Debian does provide some tools used for remote scanning of hosts (but not vulnerability assesment). These tools are, in some cases, used by vulnerability assesment scanners as the first type of "attack" run against remote hosts in an attempt to determine remote services available. Currently Debian provides:

- nmap
- xprobe
- queso
- knocker
- hping2
- isic
- icmpush
- nbtscan

Whileas `queso` and `xprobe` provide only remote operating system detection (using TCP/IP fingerprinting), `nmap` and `knocker` do both operating system detection and port scanning of the remote hosts. On the other hand, `hping2` and `icmpush` can be used for remote ICMP attack techniques.

Designed specifically for Netbios networks, `nbtscan` can be used to scan IP networks and retrieve name information from SMB-enabled servers, including: usernames, network names, MAC addresses...

8.3 Internal audits

Currently, only the `tiger` tool used in Debian can be used to perform internal (also called white box) audit of hosts in order to determine if the filesystem is properly setup, which processes are listening on the host, etc.

8.4 Auditing source code

Debian provides two packages that can be used to audit C/C++ source code programs and find programming errors that might lead to potential security flaws:

- flawfinder
- rats

8.5 Virtual Private Networks

FIXME: Content needed

Debian provides quite a number of package to setup encrypted virtual private networks:

- `vtun`
- `tunnelv`
- `cipe`
- `vpnd`
- `tinc`
- `secvpn`
- `pptp`
- `freeswan`

IPsec (i.e. FreeSWAN) is probably the best choice overall since it promises to interoperate with most anything that runs IPsec, but these other packages can help you get a secure tunnel up in a hurry. PPTP is a Microsoft protocol for VPN. It is supported under Linux, but is known to have serious security issues.

For more information read VPN-Masquerade HOWTO (`http://www.linuxdoc.org/HOWTO/VPN-Masquerade-HOWTO.html`) (covers IPsec and PPTP) VPN HOWTO (`http://www.linuxdoc.org/HOWTO/VPN-HOWTO.html`) (covers PPP over SSH), and Cipe mini-HOWTO (`http://www.linuxdoc.org/HOWTO/mini/Cipe+Masq.html`), PPP and SSH mini-HOWTO (`http://www.linuxdoc.org/HOWTO/mini/ppp-ssh/index.html`).

8.6 Public Key Infraestructure (PKI)

When considering a PKI you are confronted to a wide variety of tools:

- a Certificate Authority that can give out certificates and can work under a given hierarchy
- a Directory to hold user's public certificates
- a Database (?) to maintain Certificate Revocation Lists
- devices that interoperate with the CA in order to print out smartcards/usb tokens/whatever to securely store certificates
- applications certificate-aware that can use certificates given out by a CA to enroll in encrypted communication and check given certificates against CRL (for authentication and full Single Sign On solutions)
- a Timestamping authority to digitally sign documents
- a management console from which all of this can be properly used (certificate generation, revocation list control, etc...)

You can use some of the software available in Debian GNU/Linux to cover some of this tools, this includes openSSL (for certificate generation), OpenLDAP (as a directory to hold the certificates), gnupg and freeswan (with X.509) support. However, the operating system does not provide (as of the woody release, 3.0) any of the freely availabe Certificate Authorities available such as pyCA, OpenCA (`http://www.openca.org`) or the CA samples from OpenSSL. For more information read the Open PKI book (`http://ospkibook.sourceforge.net/`).

8.7 SSL Infraestructure

Debian does provide some SSL certificates with the distribution so they can be installed locally. SSL certificates are distributed in the `ca-certificates`. This package provides a central repository of certificates that have been submitted to Debian and approved (that is, verified) by the package maintainer.

FIXME: read debian-devel to see if there was something added to this.

8.8 Anti-virus tools

There are not that many anti-virus tools in Debian, probably because GNU/Linux users are not that much plagued currently by virii. There have been, however, worms and virii for GNU/Linux even if there has not (yet, hopefully) been any virus that has spread on the wild over any Debian distribution. In any case, administrators might want to build up anti-virus gateways or protect themselves against them.

Debian provides currently the following tools for building anti-virus environments:

- sanitizer (http://packages.debian.org/sanitizer), a tool that can be used to filter email from procmail and remove virii.
- amavis-postfix (http://packages.debian.org/amavis-postfix), a script that provides an interface from the mail transport agent to one or more virus scanners (this package provides the postfix version).

As you can see, Debian does not currently provide any anti-virus software itself. There are, however, free software anti-virus projects which might (in the future) be included in Debian openantivirus (http://sourceforge.net/projects/openantivirus/) and jvirus (http://sourceforge.net/projects/jvirus/) (less chances for this since it is completely Java based). Also, Debian will never provide commercial anti-virus software like: Panda Antivirus (http://www.pandasoftware.com/com/linux/linux.asp), NAI Netshield (uvscan) (http://www.nai.com/naicommon/buy-try/try/products-evals.asp), Sophos Sweep (http://www.sophos.com/), TrendMicro Interscan (http://www.antivirus.com/products/), RAV (http://www.ravantivirus.com).... For more pointers see the Linux anti-virus software mini-FAQ (http://www.computer-networking.de/~link/security/av-linux_e.txt).

For more information on how to setup an a virus detection system read Dave Jones' article Building an E-mail Virus Detection System for Your Network (http://www.linuxjournal.com/article.php?sid=4882).

Chapter 9

Before the compromise

9.1 Set up Intrusion Detection.

Debian includes some tools for Intrusion Detection which you might want to setup to defend you local system (if truly paranoid of if your system is really critical) or to defend other systems in the same network.

Always be aware that in order to really improve the system's security with the introduction of any of these tools, you need to have an alert+response mechanism, so don't use Intrusion Detection if you are not going to alert anyone (i.e. don't waste your time configuring things you will not use later on).

Most of the intrusion detection tools will either log against syslog or send emails to the root user (most of them can be configured to mail instead another user) regarding the particular attack that has been detected. An administrator has to properly configure them so that false positives do not send alerts and so that alerts are take proper care of. Alerts can indicate an ongoing attack and might not be useful, say, one day later, since the attack might have been already successful. So be sure that there is a proper policy on handling alerts and that the technical mechanisms to implement it are in place.

An interesting source of information is CERT's Intrusion Detection Checklist (`http://www.cert.org/tech_tips/intruder_detection_checklist.html`)

9.1.1 Network based intrusion detection

`Snort` is a flexible packet sniffer or logger that detects attacks using an attack signature dictionary. It detects a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and much more. Snort has a real-time alerting capability. This is a tool which should be installed on every router to keep an eye on your network. Just install it via `apt-get install snort`, follow the questions and watch it log.

Snort in Debian is enabled with many security checks which you might want; however, you should customize the setup to take into account the particular services you run on your system.

You might also want to retrieve additional checks specific to these services.

You can use `snort` both to establish network intrusion detection for a range of hosts in your network as well as to detect network attacks against your own host.

There are other tools that can be used to detect network attacks (albeit more simpler). `Portsentry` is another interesting package that can tip you off when a scan is made to your site. Other tools like `ippl` or `iplogger` will also detect some IP (TCP and ICMP) attacks, even if they do not provide advanced techniques for detecting network attacks (like snort does).

You can test any of these tools with the `idswakeup` program, a false-positive generator to alert NIDSs with plenty of common attack signatures available in Debian.

9.1.2 Host based detection

`Tiger` is an old intrusion detection tool which has been ported to Debian since the woody distribution. Tiger provides check of common issues related to security break-ins, checks passwords strength, filesystem problems, communicating processes... The Debian version includes new security checks Debian-specific: MD5sums of provided binaries, and checks of installed and vulnerable packages. The default installation makes `tiger` run each day and generate a report that is sent to the superuser. The generated reports can give away information of a successful compromise of the system.

Log analysis tools, such as `logcheck` can also be used, if customised, to detect intrusion attempts. See 'Using and customising logcheck.' on page 46.

Also, any of the filesystem integrity checkers (see 'Checking filesystem integrity' on page 56) can be quite useful in order to set up detection of anomalies in a secured environment. An effective intrusion will, most surely, modify files in the local filesystem in order to circumvent local security policy, install trojans, create users... this events can be detected with them.

9.2 Useful kernel patches

FIXME: This section needs to cover how these specific patches can be installed in Debian using the kernel-2.x.x-patch-XXX packages.

There are some kernel patches, which significantly enhance system security. Here are a few of them:

- OpenWall patch by Solar Designer. This is a useful set of kernel restrictions, like restricted links, FIFOs in `/tmp`, restricted `/proc`, special file descriptor handling, non-executable user stack area and some more. Homepage: `http://www.openwall.com/linux/`
- *LIDS — Linux intrusion detection system by Huagang Xie & Philippe Biondi.* This patch makes the process of creating a hardened Linux system easier. You can restrict every process, give it rights to write or read files, or remove, by default, the ability to read files. Furthermore you can also set capabilities for certain processes. Even though it is still in

the beta phase, it is almost a must for the paranoid system administrator. Homepage: http://www.lids.org

- *POSIX Access Control Lists (ACLs) for Linux* This patch adds access control lists, an advanced method for restricting access to files, to the linux kernel. Homepage: http://acl.bestbits.at/

- *Linux trustees*. This patch adds a decent advanced permissions system to your Linux kernel. All the objects are stored in the kernel memory, which allows fast lookup of all permissions. Homepage: http://trustees.sourceforge.net/

- *International kernel patch*. This is a crypt-oriented kernel patch, therefore you have to pay attention to your local laws regarding the use of cryptography. It basically adds the possibility of using encrypted file systems. Homepage: http://www.kerneli.org

- *SubDomain*. A kernel extension to create a more secure and easier to setup chroot environment. You can specify the files needed for the chrooted service manually and do not have to compile the services statically. Homepage: http://www.immunix.org/subdomain.html

- *UserIPAcct*. This is not really a security related patch, but it allows you to create quotas for the traffic on your server per user. And you can fetch statistics about the user traffic. Homepage: http://ramses.smeyers.be/useripacct.

- *FreeS/WAN*. If you want to use IPSec with Linux, you need this patch. You can create VPNs with this quite easily, even to Windows machines, as IPSec is a common standard. Homepage: http://www.freeswan.org

9.3 Avoiding root-kits

9.3.1 LKM - Loadable Kernel Modules

LKM (Loadable Kernel Modules) are files containing dynamically loadable kernel components. They are dynamically loadable in kernel to run assigned tasks. On the GNU/Linux they are used to expand the functionality of kernel. Several advantages can be taken using LKMs, as we saw, they can dynamically be loadabled without recompiling the entire kernel, can be used to specify devices drivers (or filesystems) and other hardware drivers like soundcards, networkcards. But some crackers might use LKMs for root-kits (knark and adore) to install backdoors for GNU/Linux systems.

LKM root-kits can hide processes, files, directories and even connections without modifying the source code of binaries. For example, `ps` might have get processes information from `/proc`, a malicious LKM can subvert the kernel to hide specific processes from procfs, so not even a known good copy of the binary `ps` could list all the correct proccesses information.

9.3.2 Detecting root-kits

The detection work can be simple and painless, or difficult and tiring, depending the measure that you choose. There are two measure of defenses regarding LKM (Linux Kernel Modules) security, the proactive and reactive.

Proactive defense

The advantage of this defense is that its prevents that some lkm root-kit damages the system. The most used proactive defense is *getting there first*, that is loading a designed LKM to protect the system to be damaged by a malicious designed LKM. Another measure is to remove capabilities in the kernel, making the system more secure. For example, you can remove the capability to stop the loading and unloading of kernel modules.

On Debian systems you can find some packages that are a proactive tool:

- `kernel-patch-2.4-lsm` - LSM is the Linux Security Modules framework.
- `lcap` - A user friendly interface to remove *capabilities* (kernel-based access control) in the kernel, making the system more secure. Running `lcap CAP_SYS_MODULE` [1] will remove module loading capabilities (even for the root user). [2]

If you don't really need these many kernel features on your GNU/Linux system you might want disable loadable modules support during kernel configuration. It prevents LKM root-kits, but you couldn't use the kernel module features on your GNU/Linux. Look that disabling loadable modules you can overload the kernel, sometimes it is not necessary.

To disable loadable module support, just set CONFIG_MODULES=n on `.config`.

Reactive defense

The advangate of reactive defense is that it has low overload the system's resources. It works comparing the system call table with known clean copy in a disk file (System.map). The most obvius desavantage is that its tell to administrator only when the system have already be compromised.

[1] There are over 28 capabilities including: `CAP_BSET`, `CAP_CHOWN`, `CAP_FOWNER`, `CAP_FSETID`, `CAP_FS_MASK`, `CAP_FULL_SET`, `CAP_INIT_EFF_SET`, `CAP_INIT_INH_SET`, `CAP_IPC_LOCK`, `CAP_IPC_OWNER`, `CAP_KILL`, `CAP_LEASE`, `CAP_LINUX_IMMUTABLE`, `CAP_MKNOD`, `CAP_NET_ADMIN`, `CAP_NET_BIND_SERVICE`, `CAP_NET_RAW`, `CAP_SETGID`, `CAP_SETPCAP`, `CAP_SETUID`, `CAP_SYS_ADMIN`, `CAP_SYS_BOOT`, `CAP_SYS_CHROOT`, `CAP_SYS_MODULE`, `CAP_SYS_NICE`, `CAP_SYS_PACCT`, `CAP_SYS_PTRACE`, `CAP_SYS_RAWIO`, `CAP_SYS_RESOURCE`, `CAP_SYS_TIME`, and `CAP_SYS_TTY_CONFIG`. All of them can be activated or de-activated in or to harden your kernel.

[2] You do not need to install `lcap` to do this but it's easier than setting `/proc/sys/kernel/cap-bound` by hand.

Detection of root-kits in Debian can be accomplished with `chkrootkit`. The Chkrootkit (`http://www.chkrootkit.org`) program checks for signs of root-kits on the local system and tells whether the target computer is infected with a root-kit.

You can also use KSTAT (`http://www.s0ftpj.org/en/site.html`) (Kernel Security Therapy Anti Trolls) by the S0ftproject group. KSTAT checks the kernel memory area (`/dev/kmem`) for information about the target host, this information includes the installation of Loadable Kernel Modules.

FIXME: Add info on how to compile the kernel w/o lkm support?

9.4 Genius/Paranoia Ideas — what you could do

This is probably the most unstable and funny section, since I hope that some of the "duh. that sounds crazy" ideas might be realized. Following here you will find some ideas — it depends on the point of view whether you say they are genius, paranoid, crazy or secure — to increase your security rapidly but you will not come unscathed out of it.

- Playing around with PAM. As said in the phrack 56 PAM article, the nice thing with PAM is that "You are limited only by what you can think of." It is true. Imagine root login only possible with fingerprint or eyescan or cryptocard (why did I do an OR conjunction and not AND here?).
- Fascist Logging. I would say everything we talked about logging above is "soft logging". If you want to perform real logging, get a printer with fanfold paper and log everything hard by printing on it. Sounds funny, but it's reliable and it cannot be removed.
- CD distribution. This idea is very easy to realize and offers pretty good security. Create a hardened Debian distribution, with proper firewall rules, make an ISO image of it and burn it on CD. Make it bootable. This is a read-only distribution with about 600 MB space for services, and it is impossible for intruders to get read/write access on this system. Just make sure every data which should get written, gets written over the wires. Anyway, the intruder cannot change firewall rules, routing entries or start own daemons (he can, but reboot and he has to hack into your system again to change them).
- Switch module capability off. When you disable the usage of kernel modules at kernel compile time, many kernel based back doors are impossible to implement, since most of them are based on installing modified kernel modules.
- Logging through serial cable (contributed by Gaby Schilders). As long as servers still have serial ports, imagine having one dedicated log-machine disconnected from the net in the middle with a serial-port multiplexer (cyclades or the like). Now have all your servers log to their serial ports. Write only. The log-machine only accepts plain text as input on its serial ports and only writes it to a log-file. Hook up a cd/dvd-writer. When the log file nears 600MB it writes it to cd-rom. Now if only they would make writers with auto-changers... Not as hard-copy as the printer, but it can handle larger volumes and the cd's don't take as much storage-space.

- Set all stuff to immutable (taken from the Tips-HOWTO, written by Jim Dennis). Right after you install and configure your system go through the `/bin`, `/sbin/`, `/usr/bin`, `/usr/sbin` and `/usr/lib` (and a few of the other usual suspects) and make liberal use of the `chattr +i command`. Also add that to the the kernel files in root. Now `mkdir /etc/.dist/` copy everything from `/etc/` on down (I do this in two steps using `/tmp /etcdist.tar` to avoid recursion) into that directory. (Optionally you can just create `/etc/.dist.tar.gz`) – and mark that as immutable.

  The reason for all of this is to limit the damage that you can do when logged in as root. You won't overwrite files with a stray redirection operator, and you won't make the system unusable with a stray space in an `rm -fr` command (you might still do plenty of damage to your data — but your libs and bins will be safer.

  This also makes a variety of security and denial of service exploits either impossible or more difficult (since many of them rely on overwriting a file through the actions of some SUID program that *isn't providing an arbitrary shell command*).

  The only inconvenience of this is when building and doing your `make install` on various sorts of system binaries. On the other hand it also prevents the `make install` from over-writing the files. When you forget to read the Makefile and chattr -i the files that are to be overwritten (and the directories to which you want to add files) - the make fails, you just use the chattr command and rerun it. You can also take that opportunity to move your old bin's, libs, or whatever into a .old/ directory or rename or tar them or whatever.

  Note that this also prevents you from upgrading your system's packages. Since the files that they provide cannot be overwritten, so you might want to have a mechanism to disable the immutable flag on all binaries right before doing an `apt-get update`.

9.4.1 Building a honeypot

FIXME: More Content specific to Debian needed.

If you wish (and can also implement it and dedicate time to it) you can set a full honeypot using a Debian GNU/Linux system. You have all the tools needed in order to setup all the honeynet (i.e. the network, the honeypot is just the fake server): the firewall, the network intrusion detector and the fake server. Be careful, however, you have to be pretty sure that you will be alerted in time (see 'The importance of logs and alerts' on page 46) so that you can take appropiate measures and terminate the compromise as soon as you fill you've seen enough.

- the firewalling technology you will need (provided by the Linux kernel).
- `syslog-ng` to send the logs from the honeypot to a remote syslog server machine.
- `snort` to setup capture of all the incoming network traffic to the honeypot and detect the attacks.
- `osh` which could be used to setup a restricted shell with logging (see Lance Spitzner's article below).

- of course, all the servers for your fake server honeypot you can imagine of (but do *not* harden the honeypot)).
- and also fake services, provided by `dtk` if you want to use the honeynet also as an intrusion detection service.
- Integrity checkers (see 'Checking filesystem integrity' on page 56) and The Coroner's Toolkit (`tct`) to do post-attack audits.

You can read more about building honeypots in Lanze Spitzner's excellent article To Build a Honeypot (`http://www.net-security.org/text/articles/spitzner/honeypot.shtml`) (from the Know your Enemy series), or David Raikow's Building your own honeypot (`http://www.zdnetindia.com/techzone/resources/security/stories/7601.htm`). Also, the Honeynet Project (`http://project.honeynet.org/`) is dedicated to building honeypots and auditing attacks made to them, there is valuable information there on howto setup a honeypot and howto audit the results of an attack (check out the contest).

Chapter 10

After the compromise

10.1 General behavior

If you are physically present when an attack is happening and doing the following will not adversly affect any bussiness transactions, simply unplug the NIC until you can figure out what the intruder did and secure the box. Disabling the network at layer 1 is the only true way to keep the attacker out of the compromised box. (Phillip Hofmeister's wise advice)

If you really want to fix the compromise quickly, you should remove the compromised host from your network and re-install the operating system from scratch. This might not have any effect if you do not know how the intruder got root. In this case you must check everything: firewall/file integrity/loghost logfiles and so on. For more information on what to do following a break-in, see Sans' Incident Handling Guide (`http://www.sans.org/y2k/DDoS.htm`) or CERT's Steps for Recovering from a UNIX or NT System Compromise (`http://www.cert.org/tech_tips/root_compromise.html`).

10.2 Backing up the system

Remember that if you are sure the system has been compromised you cannot trust the software in it or any information that it gives back to you. Applications might have been troyanized, kernel modules might be installed, etc.

The best thing to do is a complete filesystem backup copy (using `dd`) after booting from a safe medium. Debian GNU/Linux Cds can be handily used for this since they provide a shell in console 2 when the installation is started (jump to it using Alt+2 and pressing Enter). The shell can be used to backup the information to another place (maybe a network file server through NFS/FTP...) for analysis while the system is offline (or reinstalled).

If you are sure that there is only a troyan kernel module you can try to run the kernel image from the CD in *rescue* mode. Make sure to startup also in *single* mode so no other trojan processes run after the kernel.

10.3 Forensics analysis

If you wish to gather more information, the `tct` (The Coroner's Toolkit from Dan Farmer and Wietse Venema) package contains utilities which perform a 'post mortem' of a system. `tct` allows the user to collect information about deleted files, running processes and more. See the included documentation for more information.

Forensics analysis should be done always on the backup copy of the data, *never* on the data itself since it might be tampered through this analysis (and lost).

FIXME: This paragraph will hopefully provide more information about forensics in a Debian system in the coming future.

FIXME: talk on how to do a debsums on a stable system with the md5sums on CD and with the recovered filesystem restored on a separate partition.

Chapter 11

Frequently asked Questions (FAQ)

This chapter will introduce some of the most common questions that usually appear in the security mailing list. You should read them before posting there or else people might tell you just to RTFM.

11.1 Security in the Debian operating system

11.1.1 Is Debian more secure than X?

A system is as secure as its administrator is capable of making it. Debian tries to install services in a *secure by default* way and might not try to be as paranoid as other operating systems which install all the services *disabled by default*. However, the system administrator needs to adapt the security of the system to his local security policy.

11.1.2 There are many Debian bugs in bugtraq does this mean that it is very vulnerable?

Debian contains quite a number of packages and different software, probably more that provided by some propietary operating systems. This means that there might be lurking more potential security issues than in systems with less software.

However, there are many advisories related to source code audits done to major software components included in Debian. Whenever such source code audit turn out a major flaw it is fixed and a advisory is sent to list such as bugtraq.

Bugs that are present in the Debian distribution usually affect other vendors and other distributions as well. Check the "Debian specific: yes/no" part on top of each advisory (DSA). If there is a "yes", it only affects Debian, if there is a "no" it probably also affects other distributions as well.

Debian contains quite a lot of packages, and nowadays there are many groups looking for security problems in software (for whichever reasons).

11.1.3 Has Debian any certification security-wise?

Simple answer: no.

Long answer: certification costs money and nobody has dedicated resources in order to certificate the Debian GNU/Linux distribution to any level of, for example, the Common Criteria. If you are interested in having a certified GNU/Linux distribution try to provide resources in order to make it possible.

11.1.4 Is there any hardening program for Debian?

Yes. Bastille Linux (`http://www.bastille-linux.org`), originally oriented towards some Linux distributions (RedHat and Mandrake) currently works for Debian. Steps are being taken to integrate the changes made to the upstream version, in any case the package in Debian is, of course, name `bastille`.

Some people believe, however, that a hardening tool does not eliminate the need for good administration.

11.1.5 I want to run XYZ service, which one should I choose?

One of the benefits of Debian, which might confuse the novice administrator, is that it provides a lot of software to offer the same service (dns servers, mail servers, ftp servers, web servers...). If you want information on what server you should be installing there's no general answer, it really depends on your feature and security needs (which might need to be balanced).

Things you should check:

- Is it maintained upstream? (when was the last release?)
- Is it mature? (the version number does *not* tell you, try to trace it's history)
- Is it bug-ridden? Have there been security advisories related to it?
- Does it provide all the functionality you need? Does it provide more than you really need?

11.1.6 How can I make service XYZ more secure in Debian?

You will find information in this document to make some services (FTP, Bind) more secure in Debian GNU/Linux. For services not covered here, however, check the program's documentation, or general Linux information. Most of the security guidelines for Unix systems apply also to Debian so securing service X in Debian is, most of the time, like securing the service for any other Linux distribution (or Unix, for that matter).

11.1.7 Are all Debian packages safe?

The Debian security team cannot analise all the packages included in Debian for potential security vulnerabilities, since there are just not enough resources to source-code audit all the project. However, Debian does benefit from the source code audits made by upstream developers or other projects like the Linux Kernel Security Audit Project (http://kernel-audit.sourceforge.net/) or the Linux Security-Audit Project (http://www.lsap.org/).

As a matter of fact, a Debian developer could distribute a trojan in a package and there is no possible way to check it out. Even if they would be introduced in Debian it would be impossible to cover all the possible situations in which the trojan would execute.

This sticks to the *no guarantees* license clause. In any case, Debian users can take confidence in that the stable code has a wide audience and most problems would be uncovered through use. It is not recommended to install untested software in a valuable system in any case (if you cannot provide the necessary code audit). And, in any case, if there were an induced security vulnerability in the distibution, the process used to include them (using digital signatures) ensures that the problem can be ultimately traced to the developer, and the Debian project has not taken this issues lightly.

11.1.8 Operating system users and groups

Are all system users necessary?

Yes and no. Debian comes with some predefined users (id < 99 as described in Debian Policy (http://www.debian.org/doc/debian-policy/)) for some services so that installing new services is easy (they are already run by the appropriate user). If you do not intend to install new services, you can safely remove those users who do not own any files in your system and do not run any services.

You can easily find users not owning any files by executing the following command (be sure to run it as root, since a common user might not have enough permissions to go through some sensitive directories):

```
cut -f 1 -d : /etc/passwd |
while read i; do find / -user "$i" | grep -q . && echo "$i"; done
```

These users are provided by `base-passwd`. You will find in its documentation more information on how these users are handled in Debian.

The list of default users (with a corresponding group) follows:

- root: Root is (typically) the superuser.
- daemon: Some unprivileged daemons that need to be able to write to some files on disk run as daemon.daemon (portmap, atd, probably others). Daemons that don't need to

own any files can run as nobody.nogroup instead, and more complex or security conscious daemons run as dedicated users. The daemon user is also handy for locally installed daemons, probably.

- bin: maintained for historic reasons.
- sys: same as with bin. However, /dev/vcs* and `/var/spool/cups` are owned by group sys.
- sync: The shell of user sync is `/bin/sync`. Thus, if its password is set to something easy to guess (such as ""), anyone can sync the system at the console even if they have no account on the system.
- games: Many games are sgid to games so they can write their high score files. This is explained in policy.
- man: The man program (sometimes) runs as user man, so it can write cat pages to `/var/cache/man`
- lp: Used by printer daemons.
- mail: Mailboxes in `/var/mail` are owned by group mail, as is explained in policy. The user and group are used for other purposes as well by various MTA's.
- news: Various news servers and other associated programs (such as suck) use user and group news in various ways. Files in the news spool are often owned by user and group news. Programs such as inews that can be used to post news are typically sgid news.
- uucp: The uucp user and group is used by the UUCP subsystem. It owns spool and configuration files. Users in the uucp group may run uucico.
- proxy: Like daemon, this user and group is used by some daemons (specifically, proxy daemons) that don't have dedicated user id's and that need to own files. For example, group proxy is used by pdnsd, and squid runs as user proxy.
- majordom: Majordomo has a statically allocated uid on Debian systems for historical reasons. It is not installed on new systems.
- postgres: Postgresql databases are owned by this user and group. All files in `/var/lib/postgresql` are owned by this user to enforce proper security.
- www-data: Some web browsers run as www-data. Web content should *not* be owned by this user, or a compromised web server would be able to rewrite a web site. Data written out by web servers, including log files, will be owned by www-data.
- backup: So backup/restore responsibilities can be locally delegated to someone without full root permissions.
- operator: Operator is historically (and practically) the only 'user' account that can login remotely, and doesn't depend on NIS/NFS.

- list: Mailing list archives and data are owned by this user and group. Some mailing list programs may run as this user as well.
- irc: Used by irc daemons. A statically allocated user is needed only because of a bug in ircd – it setuid()s itself to a given UID on startup.
- gnats.
- nobody, nogroup: Daemons that need not own any files run as user nobody and group nogroup. Thus, no files on a system should be owned by this user or group.

Other groups which have no associated user:

- adm: Group adm is used for system monitoring tasks. Members of this group can read many log files in `/var/log`, and can use xconsole. Historically, `/var/log` was `/usr/adm` (and later `/var/adm`), thus the name of the group.
- tty: Tty devices are owned by this group. This is used by write and wall to enable them to write to other people's tty's.
- disk: Raw access to disks. Mostly equivalent to root access.
- kmem: /dev/kmem and similar files are readably by this group. This is mostly a BSD relic, but any programs that need direct read access to the system's memory can thus be made sgid kmem.
- dialout: Full and direct access to serial ports. Members of this group can reconfigure the modem, dial anywhere, etc.
- dip: The group's name stands for "Dialup IP". Being in group dip allows you to use a tool such as `ppp`, `dip`, `wvdial`, etc. to dial up a connection. The users in this group cannot configure the modem, they can just run the programs that make use of it.
- fax: Allows members to use fax software to send / receive faxes.
- voice: Voicemail, useful for systems that use modems as answering machines.
- cdrom: This group can be used locally to give a set of users access to a cdrom drive.
- floppy: This group can be used locally to give a set of users access to a floppy drive.
- tape: This group can be used locally to give a set of users access to a tape drive.
- sudo: Members of this group do not need to type their password when using sudo. See `/usr/share/doc/sudo/OPTIONS`.
- audio: This group can be used locally to give a set of users access to an audio device.
- src: This group owns source code, including files in `/usr/src`. It can be used locally to give a user the ability to manage system source code.

- shadow: `/etc/shadow` is readable by this group. Some programs that need to be able to access the file are set gid shadow.
- utmp: This group can write to `/var/run/utmp` and similar files. Programs that need to be able to write to it are sgid utmp.
- video: This group can be used locally to give a set of users access to an video device.
- staff: Allows users to add local modifications to the system (`/usr/local`, `/home`) without needing root privileges. Compare with group "adm", which is more related to monitoring/security.
- users: While Debian systems use the user group system by default (each user has their own group), some prefer to use a more traditional group system. In that system, each user is a member of the 'users' group.

What is the difference between the adm and the staff group?

'adm' are administrators and is mostly useful to allow them to read logfiles without having to `su`. 'staff' is useful for more helpdesk/junior sysadmins type of people and gives them the ability to do things in `/usr/local` and create directories in `/home`.

11.1.9 Question regarding services and open ports

Why are all services activated on installation?

That's just an approach to the problem of being, on one side, security conscious and on the other side user friendly. Unlike OpenBSD, which disables all services unless activated by the administrator, Debian GNU/Linux will activate all installed services unless deactivated (see 'Disabling daemon services' on page 24 for more information). After all you installed the service, didn't you?

There has been a lot of discussion on Debian mailing lists (both at debian-devel and at debian-security) regarding which should be the standard setup. However, there is not a consensus as of this writting (march 10th 2002) on how it should be tackled.

Can I remove inetd?

Inetd is not easy to remove since `netbase` depends on the package that provides it (`netkit-inetd`. If you want to remove it you can either disable it (see 'Disabling daemon services' on page 24 or remove the package by using the `equivs` package.

Why do I have port 111 open?

Port 111 is sunrpc's portmapper, it is installed by default in all base installations of a Debian system since there is no need to know when a user's program might need RPC to work out correctly. In any case, it is used mostly for NFS. If you do not need it, remove it as explained in 'Disabling RPC services' on page 74.

What use is identd (113) for?

Identd is used for administrators to provide userid details to remote systems who want to know who's responsible for a given connection from your machine. Notably this includes mail, FTP and IRC servers, however, it can also be used to trace down which user in your local system is attacking a remote system.

There has been extensive discussion for this see the mailing list archives (`http://lists.debian.org/debian-security/2001/debian-security-200108/msg00297.html`), basicly if you do not know what it is for, don't activate it. But if you firewall it, *please* make it a reject rule and not a deny rule, otherwise communications might hang until a timeout expires (see reject or deny issues (`http://logi.cc/linux/reject_or_deny.php3`)).

I have checked I have the following port (XYZ) open, can I close it?

Of course you can, the ports you are leaving open should adhere to your site's policy regarding public services available to other systems. Check if they are open by inetd (see 'Disabling inetd services' on page 25) or by other installed packages and take appropriate measures (configure inetd, remove the package, avoid it running on bootup...)

I have removed services from `/etc/services`, am I ok?

No, `/etc/services` just provides a mapping from a virtual name to a given port number, removing names from there will not (usually) prevent services from being started. Some daemons might not run if `/etc/services` is modified but that's not the norm, and it's not the recommended way to do it, see 'Disabling daemon services' on page 24.

11.1.10 Common security issues

I have lost my password and cannot access the system!!

The steps you need to take in order to recover from this depends on whether or not you have applied the suggested procedure for limiting access to Lilo and BIOS.

If you have limited both. You need to disable the BIOS features (only boot from hard disk) before proceeding, if you also forgot your BIOS password, you will have to open your system and manually remove the BIOS battery.

If you have bootup of CD-ROM or diskette enable, you can:

- bootup from a rescue disk and start the kernel
- go to the virtual console (Alt+F2)
- mount the hard disk where your /root is
- edit (Debian 2.2 rescue disk comes with `ae`, Debian 3.0 comes with `nano-tiny` which is similar to `vi`) `/etc/shadow` and change the line:

  ```
  root:asdfjl290341274075:XXXX:X:XXXX:X::: (X=any number)
  ```

  to:

  ```
  root::XXXX:X:XXXX:X:::
  ```

This will remove the root password. You can startup the system and root from the login: prompt as root (with an empty password). This will work unless you have configured the system more tightly, i.e. if you have allowed users with null passwords and root can login from the console.

If you have introduced also this features you will need to enter in single mode. LILO needs not to be restricted, if you have done this too you will need to rerun `lilo` just after the root reset above. This is quite tricky since your `/etc/lilo.conf` will need to be tweaked due to having a / filesystem that is a ramdisk and not the real harddisk.

Once LILO is not restricted it. You can:

- Press the Alt, shift or Control key just before the system BIOS finishes, you should get the LILO prompt.
- Type 'linux single', 'linux init=/bin/sh' or 'linux 1' in the prompt.
- you should get to a shell prompt in singleuser mode (it will ask for password but you already know it)
- remount read/write the / partition

  ```
  mount -o remount,rw /
  ```

- change the superuser password with `passwd` (since you are superuser it will not ask for the previous password).

11.1.11 I want to setup a service for my users bot not give shell accounts.

If you want to give a POP service, for example, you do not need to setup a user account for each user accessing it. It's best to setup a directory-based authentication through an external service (like Radius, LDAP or an SQL database). Just install the appropiate PAM library (`libpam-radius-auth`, `libpam-ldap`, `libpam-pgsql` or `libpam-mysql`), read the documentation (for starters, see 'User authentication: PAM' on page 35 and configure the PAM-enabled service to use the backend you have chosen. This is done by editing the files under `/etc/pam.d/` for your service and modify the

```
auth    required        pam_unix_auth.so shadow nullok use_first_pass
```

to, for example, ldap:

```
auth    required        pam_ldap.so
```

In the case of LDAP directories, some services provide LDAP schemas to be included in your directory that need to be included in order to use LDAP autentication for it. If you are using a relational database, a useful trick is to use the *where* clause when configuring the PAM modules. For example if you have a database with the following table:

```
(user_id,user_name,realname,shell,password,uid,gid,homedir,sys,pop,imap,ftp)
```

You can use the last (boolean) values to enable or disable access to the different services just by inserting the appropiate lines in the following files:

- `/etc/pam.d/imap`:`where=imap=1`.
- `/etc/pam.d/qpopper`:`where=pop=1`.
- `/etc/nss-mysql*.conf`:`users.where_clause = user.sys = 1;`.
- `/etc/proftpd.conf`:`SQLWhereClause "ftp=1"`.

11.2 My system is vulnerable! (are you sure?)

11.2.1 I've seen an attack on my system logs: am I compromised?

A trace of an attack does not always mean that you've been cracked into, you should take the usual steps to determine if the system is compromised (see 'After the compromise' on page 105). Also notice that sometimes, the fact that you see the attacks in the log might mean you system is already vulnerable to it (a determined attacker might have used some other besides the ones you have seen, however).

11.2.2 I have found strange 'MARKs' in my logs: Am I compromised?

You might find the following in your system logs. If your system does not have a lot of load (and services) they probably be one of the few appearing in your logs:

```
Dec 30 07:33:36 debian -- MARK --
Dec 30 07:53:36 debian -- MARK --
Dec 30 08:13:36 debian -- MARK --
```

This does not indicate any cand of compromise, and users changing between Debian releases might find it strange. It is, a matter of fact an indication of `syslogd` running properly. From `syslogd(8)`:

```
       -m interval
              The syslogd logs a mark timestamp  regularly.   The
              default interval between two -- MARK -- lines is 20
              minutes.  This can be  changed  with  this  option.
              Setting the interval to zero turns it off entirely.
```

11.2.3 I found users doing 'su' in my logs: Am I compromised?

You might find lines in your logs like:

```
Apr  1 09:25:01 server su[30315]: + ??? root-nobody
Apr  1 09:25:01 server PAM_unix[30315]: (su) session opened for user nobody
```

Don't worry too much, check out if this is due to a job running through the cron (usually `/etc /cron.daily/find` or `logrotate`):

```
$ grep 25 /etc/crontab
25 6    * * *   root    test -e /usr/sbin/anacron || run-parts --report
/etc/cron.daily
$ grep nobody /etc/cron.daily/*
find:cd / && updatedb --localuser=nobody 2>/dev/null
```

11.2.4 I have suffered a break-in what do I do?

Read this document and take the appropiate measures outlined here. If you need assistance you might use the debian-security@lists.debian.org to ask for advice on how to recover/patch your system.

11.2.5 How can I trace an attack?

Watching the logs (if they have not been changed), using intrusion detection systems (see 'Set up Intrusion Detection.' on page 97), `traceroute`, `whois` and similar tools (including forensic analysis) you can trace an attack to the source. The way you should react on this information depends, solely, on your appropiate security policy, and what *you* consider an attack is. Is a remote scan an attack? Is a vulnerability probe an attack?

11.2.6 Program X in Debian is vulnerable, what do I do?

Take a moment, first, to see if the vulnerability has been announced in public security mailing lists (like Bugtraq) or other forums, the Debian Security Team keeps up to date with this lists, so they might already be aware of the problem. Do not take any further actions if you see an announcement already at `http://security.debian.org`.

If you do not see any of this, please send mail on the affected packages as well as a description of the vulnerability as detailed as possible (proof of concept code is also ok) to security@debian.org which will get you in touch with the security team.

11.2.7 The version number for a package indicates that I am still running a vulnerable version!

Instead of upgrading to a new release we backport security fixes to the version that was shipped in the stable release. The reason we do this is to make sure that a release changes as little as possible so things will not change or break unexpectedly as a result of a security fix. You can check if you are running a secure version of a package by looking at the package changelog, or comparing its exact (upstream version -slash- debian release) version number with the version indicated in the Debian Security Advisory.

11.2.8 Specific software

Proftpd is vulnerable to a Denial of Service attack

Add `DenyFilter \*.*/` to your configuration file, for more information see `http://www.proftpd.org/critbugs.html`.

11.3 Questions regarding the Debian security team

11.3.1 What is a Debian Security Advisory (DSA)

It is the information sent by the Debian Security Team (see below) informing of a fix of a security related vulnerability available for the Debian operating system. Signed DSAs are sent to

public mailing lists and posted in Debian's web site (both in the front page and in the security area (http://www.debian.org/security/)).

DSAs include information on the affected package/s, the bug discovered and where to retrieve the updated packages (and their MD5 sums).

11.3.2 The signature on Debian advisories does not verify correctly!

This is most likely a problem on your end. The debian-security-announce list has a filter that only allows messages with a correct signature from one of the security team members to be posted.

Most likely some piece of mail software on your end slightly changes the message that breaks the signature. Make sure your software does not do any MIME encoding or decoding, or tab/space conversions.

Known culprits are fetchmail (with the mimedecode option enabled) and formail (from procmail 3.14 only).

11.3.3 How are security incidents handled in Debian?

Once the Security Team receives a notification of an incident, one or more members review it and consider Debian/stable vulnerable or not. If our system is vulnerable, it is worked on a fix for the problem. The package maintainer is contacted as well, if he didn't contact the Security Team already. Finally the fix is tested and new packages are prepared, which then are compiled on all stable architectures and uploaded afterwards. After all this tasks are done a Debian Security Advisory (DSA) is sent to public mailing lists.

11.3.4 How much time will it take Debian to fix vulnerability XXXX?

Analysis of the time it takes the Debian security team to send an advisory and produce fixed packages once a vulnerability is known show that it does not take much time for vulnerabilities to be fixed in the stable distribution.

A report published in the debian-security mailinglist (http://lists.debian.org/debian-security/2001/debian-security-200112/msg00257.html) showed that in the year 2001, it took the Debian Security Team an average of 35 days to fix security-related vulnerabilites. However, over 50% of the vulnerabilities where fixed in a 10-days time frame, and over 15% of them where fixed the *same day* the advisory was released.

However, when asking this question people tend to forget that:

- DSAs are not sent until:
  - packages are available for *all* architectures supported by Debian (this takes some time for packages that are part of the system core, specially considering the number of architectures supported in the stable release).

  - new packages are thoroughly tested in order to ensure that no bugs are introduced

- Packages might be available before the DSA is sent (in the incoming queue or in the mirrors).
- Debian is a volunteer-based project.
- there is a "no guarantees" clause, that is part of the license with which Debian is given.

11.3.5 How is security handled for `testing` and `unstable`?

The short answer is: it's not. Testing and unstable are rapidly moving targets and the security team does not have the resources needed to properly support those. If you want to have a secure (and stable) server you are strongly encouraged to stay with stable.

However, as a matter of fact, unstable usually gets fixed really quick since security updates sometimes get they are usually available upstream faster (other versions, like those in stable need usually to be backported).

11.3.6 Why are there no official mirrors for security.debian.org?

A: The purpose of security.debian.org is to make security updates available as quickly and easily as possible. Mirrors would add extra complexity that is not needed and can cause frustration if they are not up to date.

11.3.7 I've seen DSA 100 and DSA 102, now where is DSA 101?

Several vendors (mostly of GNU/Linux, but also of BSD derivatives) coordinate security advisories for some incidents and agree to a particular timeline so that all vendors are able to properly determine if they are (or not) vulnerable and create the patchs needed.

The Debian Security Team holds, in this case, the numbers before the advisory can be released, and hence temporarily leaving out one or more advisories by number.

11.3.8 How can I reach the security team?

A: Security information can be sent to security@debian.org, which is read by all Debian developers. If you have sensitive information please use team@security.debian.org which only the members of the security team read. If desired email can be encrypted with the Debian Security Contact key (key ID 363CCD95).

11.3.9 What different is there between security@debian.org and debian-security@lists.debian.org?

When you send messages to security@debian.org these are sent to the developers mailing list (debian-private) to which all Debian developers are subscribed to, posts to this list are kept private (i.e. are not archived at the public website). Debian-security@lists.debian.org is a public mailinglist, open to anyone that wants to subscribe to it, and there are searchable archives available at the web site.

11.3.10 How can I contribute with the Debian security team?

- By contributing to this document, fixing FIXMEs or providing new content. Documentation is important and reduces the overload of answering common issues. Translation of this documentation into other languages is also of great help.
- By packaging applications that are useful for providing/checking security in/using a Debian system. If you are not a developer, file a WNPP bug (http://www.debian.org/devel/wnpp/) and ask for software you think might be useful and is not currently provided.
- Audit applications in Debian or help solve security bugs and report issues to security@debian.org. Other projects' work like the Linux Kernel Security Audit Project (http://kernel-audit.sourceforge.net/) or the Linux Security-Audit Project (http://www.lsap.org/) increase the security of Debian GNU/Linux since contributions will eventually help here too.

In any case, please review each problem before reporting it to security@debian.org. If you are able to provide patches, that would speed up the process. Do not simply forward bugtraq mails, since they are received already. Providing additional information, however, is always a good idea.

11.3.11 Who is the Security Team composed of?

The Debian Security Team currently consists of five members and two secretaries. The Security Team itself appoints people to join the team.

11.3.12 Does the Debian Security team check every new package in Debian?

No, the Debian security team does not check every new package and neither is there an automatic (lintian) check in order to detect malicious new packages, since those checks are rather impossible to detect automatically. Maintainers, however, are fully responsible for the software that is introduced in Debian and no software is introduced that is not first signed by an authorised developers. They are in charge of analysing the software they maintain and are security-aware.

11.3.13 I have an older version of Debian, is it security-supported?

Unfortunately no, the Debian Security Team cannot handle both the stable release (unofficially also the unstable) and other older releases. However, you can expect security updated for a limited period of time (usually several months) just after a new Debian distribution is released.

Appendix A

The hardening process step by step

A procedure is always useful, since it allows you to see the entire process of hardening the system and enables you to take decisions. A possible approach for such a procedure in Debian 2.2 GNU/Linux is shown below. This is a post-installation procedure, for a checklist of measures to be taken, step by step, during configuration see 'Configuration checklist' on page 127. Also, this procedure is (for the moment) more oriented towards hardening of network services.

- Do an installation of the system (take into account information in this howto regarding partitioning). After base installation go into custom install, do not select task packages but select shadow passwords.
- go through `dselect` and remove unneeded but selected packages before doing [I]nstall. Leave the bare minimum software in the server.
- Update all software from latest packages available at security.debian.org as explained previously in 'Execute a security update' on page 34.
- implement the suggested issues presented in this manual regarding user quotas, login definitions and lilo
- in order to do a service hardening, make a list of services currently awake in your system.

  ```
  $ ps -aux
  $ netstat -pn -l -A inet
  # /usr/sbin/lsof -i | grep LISTEN
  ```

  You will need to install `lsof-2.2` for the third command to work (run it as root). You should be aware that lsof can translate the word LISTEN to your locale settings.
- in order to remove unnecessary services, first determine how is it started and which package provides it. You can do this easily by checking the program that listens in the socket, the following example will tell you using this tools and `dpkg`

```
#!/bin/sh
# FIXME: this is quick and dirty; replace with a more robust script sni
for i in ‘sudo lsof -i | grep LISTEN | cut -d " " -f 1 |sort -u‘ ; do
        pack=‘dpkg -S $i |grep bin |cut -f 1 -d : | uniq‘
        echo "Service $i is installed by $pack";
        init=‘dpkg -L $pack |grep init.d/ ‘
        if [ ! -z "$init" ]; then
                echo "and is run by $init"
        fi
done
```

- Once you find unwanted services, remove the package (with `dpkg -purge`) or, if useful but should not be enabled on startup, use `update-rc.d` (see ‘Disabling daemon services’ on page 24) in order to remove them from the system startup.
- For inetd services (launched by the superdaemon) you can just check the enabled services, for example with:

  ```
  $ grep -v "^#" /etc/inetd.conf | sort -u
  ```

  and disable those not needed by commenting the line that includes them, removing the package, or using `update-inetd`
- If you have wrapped services (those using `/usr/sbin/tcpd`) check that the `/etc/hosts.allow` and `/etc/hosts.deny` are configured according to your service policy.
- If possible, and depending on each service, you might want to limit services when using more than one external interface to listen only on one of them. For example, if you want internal FTP access make the FTP daemon listen only on your management interface, not on all interfaces (i.e, 0.0.0.0:21).
- Reboot the machine, or switch it to single user and back to multiuser with

  ```
  $ init 1
  (....)
  $ init 2
  ```

- Check the services now available, and, if necessary, repeat steps above.
- Install now the needed services if you have not done so already, and configure them properly.
- Check what users are being used to run available services for example with:

  ```
  $ for i in ‘/usr/sbin/lsof -i |grep LISTEN |cut -d " " -f 1 |sort -u‘;
  ```

and consider changing these services to a given user/group and maybe also chrooting them for increased security. You can do this by changing the `/etc/init.d` scripts, where the service starts. Most services in Debian use `start-stop-daemon` so you can use the –change-uid option and the –chroot option to setup those services. Chrooting services is beyond the scope of this document but a word of warning is necessary: you might need to put all the files installed by the service package using dpkg -L and the packages it depends on in the chrooted environment.

- Repeat steps above in order to check that only desired services are running and that they are run as the desired user/group combination.
- Test the installed services in order to see if they work as expected.
- Check the system using a vulnerability assessment scanner (like `nessus`) in order to determine vulnerabilities in the system (misconfigurations, old services or unneeded services).
- Install network intrusion measures and host intrusion measures (like `snort` and `logsentry`).
- Repeat the network scanner step and verify that the intrusion detection systems work fine.

For the truly paranoid, consider also the following:

- Add firewalling capabilities to the system, accepting incoming connections only to offered services and limiting outgoing connections to those authorized.
- Recheck the installation with a new vulnerability assessment with a network scanner.
- Check outgoing connections using a network scanner from the system to a host outside and verify that unwanted connections do not find their way out.

FIXME: this procedure considers service hardening but not system hardening at the user level, include information regarding checking user permissions, setuid files and freezing changes in the system using the ext2 filesystem.

Appendix B

Configuration checklist

This appendix briefly reiterates points from the other sections of this manual in a condensed checklist format. This is intended as a quick summary for someone who has already read the manual. There are other good checklists available, Kurt Seifried has one setup based on a course at Securing Linux Step by Step (`http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html`).

FIXME: This is based on v1.4 of the manual and might need to be updated.

- Limit physical access and booting capabilities
  - Enable BIOS password
  - Disable floppy/cdrom/...booting
  - Set a LILO or GRUB password (`/etc/lilo.conf` or `/boot/grub/menu.lst`, respectively); check that the LILO or GRUB configuration file is read-protected.
  - Disallow MBR floppy booting back door by overwriting the MBR (maybe not?)
- Partitioning
  - Separate user-writable data, non-system data, and rapidly changing run-time data to their own partitions
  - Set `nosuid,noexec,nodev` mount options in `/etc/fstab` on ext2 partitions such as `/tmp`.
- Password hygiene and login security
  - Set a good root password
  - Enable password shadowing and MD5
  - Install and use PAM
    - Add MD5 support to PAM and make sure that (generally speaking) entries in `/etc/pam.d/` files which grant access to the machine have the second field in the pam.d file set to "requisite" or "required".

    - Tweak `/etc/pam.d/login` so as to only permit local root logins.
    - Also mark authorized tty:s in `/etc/security/access.conf` and generally set up this file to limit root logins as much as possible.
    - Add pam_limits.so if you want to set per-user limits
    - Tweak `/etc/pam.d/passwd`: set minimum length of passwords higher (6 characters maybe) and enable md5
    - Add group wheel to `/etc/group` if desired; add pam_wheel.so group=wheel entry to `/etc/pam.d/su`
    - For custom per-user controls, use pam_listfile.so entries where appropriate
    - Have an `/etc/pam.d/other` file and set it up with tight security
  - Set up limits in `/etc/security/limits.conf` (note that `/etc/limits` is not used if you are using PAM)
  - Tighten up `/etc/login.defs`; also, if you enabled MD5 and/or PAM, make sure you make the corresponding changes here, too
  - Disable root ftp access in `/etc/ftpusers`
  - Disable network root login; use `su(1)` or `sudo(1)`. (consider installing `sudo`)
  - Use PAM to enforce additional constraints on logins?

- Other local security issues
  - Kernel tweaks (see 'Configuring kernel network features' on page 50)
  - Kernel patches (see 'Useful kernel patches' on page 98)
  - Tighten up logfile permissions (`/var/log/{last,fail}log`, Apache logs)
  - Verify that setuid checking is enabled in `/etc/checksecurity.conf`
  - Consider making some log files append-only and configuration files immutable using chattr (ext2 filesystems only)
  - Set up file integrity (see 'Checking filesystem integrity' on page 56). Install `debsums`
  - Consider replacing locate with slocate
  - Log everything to a local printer?
  - Burn your configuration on a bootable CD and boot off that?
  - Disable kernel modules?

- Limit network access
  - Install and configure `ssh` (suggest PermitRootLogin No in `/etc/ssh/sshd_config`, PermitEmptyPasswords No; note other suggestions in text also)
  - Consider disabling or removing in.telnetd
  - Generally, disable gratuitous services in `/etc/inetd.conf` using `update-inetd --disable` (or disable inetd altogether, or use a replacement such as xinetd or rlinetd)

  - Disable other gratuitous network services; mail, ftp, DNS, www etc should not be running if you do not need them and monitor them regularly.
  - For those services which you do need, do not just use the most common programs, look for more secure versions shipped with Debian (or from other sources). Whatever you end up running, make sure you understand the risks.
  - Set up chroot jails for outside users and daemons.
  - Configure firewall and tcpwrappers (i.e. `hosts_access(5)`); note trick for `/etc/hosts.deny` in text.
  - If you run ftp, set up your ftpd server to always run chrooted to the user's home directory
  - If you run X, disable xhost authentication and go with ssh instead; better yet, disable remote X if you can (add -nolisten tcp to the X command line and turn off XDMCP in `/etc/X11/xdm/xdm-config` by setting the requestPort to 0)
  - Disable outside access to printers
  - Tunnel any IMAP or POP sessions through SSL or ssh; install stunnel if you want to provide this service to remote mail users
  - Set up a loghost and configure other machines to send logs to this host (`/etc/syslog.conf`)
  - Secure BIND, Sendmail, and other complex daemons (run in a chroot jail; run as a non-root pseudo-user)
  - Install snort or a similar logging tool.
  - Do without NIS and RPC if you can (disable portmap).

- Policy issues

  - Educate users about the whys and hows of your policies. When you have prohibited something which is regularly available on other systems, provide documentation which explains how to accomplish similar results using other, more secure means.
  - Prohibit use of protocols which use cleartext passwords (telnet, rsh and friends; ftp, imap, http, ...).
  - Prohibit programs which use SVGAlib.
  - Use disk quotas.

- Keep informed about security issues

  - Subscribe to security mailing lists
  - Subscribe to security updates – add to `/etc/apt/sources.list` an entry (or entries) for http://security.debian.org/debian-security
  - Also remember to periodically run `apt-get update ; apt-get upgrade` (perhaps install as a cron job?) as explained in 'Execute a security update' on page 34.

Appendix C

Setting up a stand alone IDS

You can easily setup a Debian dedicated box as an standalone Intrusion Detection System using `snort`.

Some guidelines:

- Install a base Debian system and select no additional packages.
- Download and manually (with dpkg) install necessary packages (see installed packages list below).
- Download and install ACID (Analysis Console for Intrusion Databases).

ACID is currently packaged for Debian with the `acidlab`, it provides a graphical WWW interface to snort's output. It can be downloaded from `http://www.cert.org/kb/acid/`, `http://acidlab.sourceforge.net` or `http://www.andrew.cmu.edu/~rdanyliw/snort/`. You might want to read the Snort Statistics HOWTO (`http://www.linuxdoc.org/HOWTO/Snort-Statistics-HOWTO/index.html`).

You can setup this system with, at least, two interfaces: one interface connected to a management lan (to access the results and maintain he sytem), one interface with no ip-address attached to the network segment to analyse.

In order to configure the network cards without an ip-address you cannot use the standard's Debian `/etc/network/interfaces` since the `ifup` and `ifdown` expect some more information there than it is needed. You have to (simple `ifconfig eth0 up`)

You need, besides the standard Debian installation, Apache, MySQL and PHP4 for ACID to work. Downloaded packages (Note: versions might vary depending on which Debian distribution you are using, this are from Debian *woody* september 2001):

```
ACID-0.9.5b9.tar.gz
adduser_3.39_all.deb
apache-common_1.3.20-1_i386.deb
```

```
apache_1.3.20-1_i386.deb
debconf_0.9.77_all.deb
dialog_0.9a-20010527-1_i386.deb
fileutils_4.1-2_i386.deb
klogd_1.4.1-2_i386.deb
libbz2-1.0_1.0.1-10_i386.deb
libc6_2.2.3-6_i386.deb
libdb2_2.7.7-8_i386.deb
libdbd-mysql-perl_1.2216-2_i386.deb
libdbi-perl_1.18-1_i386.deb
libexpat1_1.95.1-5_i386.deb
libgdbmg1_1.7.3-27_i386.deb
libmm11_1.1.3-4_i386.deb
libmysqlclient10_3.23.39-3_i386.deb
libncurses5_5.2.20010318-2_i386.deb
libpcap0_0.6.2-1_i386.deb
libpcre3_3.4-1_i386.deb
libreadline4_4.2-3_i386.deb
libstdc++2.10-glibc2.2_2.95.4-0.010703_i386.deb
logrotate_3.5.4-2_i386.deb
mime-support_3.11-1_all.deb
mysql-client_3.23.39-3_i386.deb
mysql-common_3.23.39-3.1_all.deb
mysql-server_3.23.39-3_i386.deb
perl-base_5.6.1-5_i386.deb
perl-modules_5.6.1-5_all.deb
perl_5.6.1-5_i386.deb
php4-mysql_4.0.6-4_i386.deb
php4_4.0.6-1_i386.deb
php4_4.0.6-4_i386.deb
snort_1.7-9_i386.deb
sysklogd_1.4.1-2_i386.deb
zlib1g_1.1.3-15_i386.deb
```

Installed packages (dpkg -l):

```
ii  adduser        3.39
ii  ae             962-26
ii  apache         1.3.20-1
ii  apache-common  1.3.20-1
ii  apt            0.3.19
ii  base-config    0.33.2
ii  base-files     2.2.0
ii  base-passwd    3.1.10
ii  bash           2.03-6
```

```
ii  bsdutils       2.10f-5.1
ii  console-data   1999.08.29-11.
ii  console-tools  0.2.3-10.3
ii  console-tools- 0.2.3-10.3
ii  cron           3.0pl1-57.2
ii  debconf        0.9.77
ii  debianutils    1.13.3
ii  dialog         0.9a-20010527-
ii  diff           2.7-21
ii  dpkg           1.6.15
ii  e2fsprogs      1.18-3.0
ii  elvis-tiny     1.4-11
ii  fbset          2.1-6
ii  fdflush        1.0.1-5
ii  fdutils        5.3-3
ii  fileutils      4.1-2
ii  findutils      4.1-40
ii  ftp            0.10-3.1
ii  gettext-base   0.10.35-13
ii  grep           2.4.2-1
ii  gzip           1.2.4-33
ii  hostname       2.07
ii  isapnptools    1.21-2
ii  joe            2.8-15.2
ii  klogd          1.4.1-2
ii  ldso           1.9.11-9
ii  libbz2-1.0     1.0.1-10
ii  libc6          2.2.3-6
ii  libdb2         2.7.7-8
ii  libdbd-mysql-p 1.2216-2
ii  libdbi-perl    1.18-1
ii  libexpat1      1.95.1-5
ii  libgdbmg1      1.7.3-27
ii  libmm11        1.1.3-4
ii  libmysqlclient 3.23.39-3
ii  libncurses5    5.2.20010318-2
ii  libnewt0       0.50-7
ii  libpam-modules 0.72-9
ii  libpam-runtime 0.72-9
ii  libpam0g       0.72-9
ii  libpcap0       0.6.2-1
ii  libpcre3       3.4-1
ii  libpopt0       1.4-1.1
ii  libreadline4   4.2-3
ii  libssl09       0.9.4-5
ii  libstdc++2.10  2.95.2-13
```

```
ii  libstdc++2.10- 2.95.4-0.01070
ii  libwrap0       7.6-4
ii  lilo           21.4.3-2
ii  locales        2.1.3-18
ii  login          19990827-20
ii  makedev        2.3.1-46.2
ii  mawk           1.3.3-5
ii  mbr            1.1.2-1
ii  mime-support   3.11-1
ii  modutils       2.3.11-13.1
ii  mount          2.10f-5.1
ii  mysql-client   3.23.39-3
ii  mysql-common   3.23.39-3.1
ii  mysql-server   3.23.39-3
ii  ncurses-base   5.0-6.0potato1
ii  ncurses-bin    5.0-6.0potato1
ii  netbase        3.18-4
ii  passwd         19990827-20
ii  pciutils       2.1.2-2
ii  perl           5.6.1-5
ii  perl-base      5.6.1-5
ii  perl-modules   5.6.1-5
ii  php4           4.0.6-4
ii  php4-mysql     4.0.6-4
ii  ppp            2.3.11-1.4
ii  pppconfig      2.0.5
ii  procps         2.0.6-5
ii  psmisc         19-2
ii  pump           0.7.3-2
ii  sed            3.02-5
ii  setserial      2.17-16
ii  shellutils     2.0-7
ii  slang1         1.3.9-1
ii  snort          1.7-9
ii  ssh            1.2.3-9.3
ii  sysklogd       1.4.1-2
ii  syslinux       1.48-2
ii  sysvinit       2.78-4
ii  tar            1.13.17-2
ii  tasksel        1.0-10
ii  tcpd           7.6-4
ii  telnet         0.16-4potato.1
ii  textutils      2.0-2
ii  update         2.11-1
ii  util-linux     2.10f-5.1
ii  zlib1g         1.1.3-15
```

Appendix D

Setting up a bridge firewall

This information was contributed by Francois Bayart in order to help users setup a linux bridge/fireall with the 2.4.x kernel and `iptables`. The only features need are the bridge firewall patch, available at sourceforge download page (http://bridge.sourceforge.net/download.html).

For example, if you are using a 2.4.18 kernel you need to download the patch (http://bridge.sourceforge.net/devel/bridge-nf/bridge-nf-0.0.6-against-2.4.18.diff) and apply it after downloading the `kernel-source-2.4.1` package and then apply the patch:

```
Zipowz:/usr/src# apt-get install kernel-source-2.4.18
Zipowz:/usr/src# cd kernel-source-2.4.18
Zipowz:/usr/src/kernel-source-2.4.18# patch -p1 < ../bridge-nf-0.0.6-against-
patching file include/linux/netfilter.h
patching file include/linux/netfilter_ipv4.h
patching file include/linux/skbuff.h
patching file net/bridge/br.c
patching file net/bridge/br_forward.c
patching file net/bridge/br_input.c
patching file net/bridge/br_netfilter.c
patching file net/bridge/br_private.h
patching file net/bridge/Makefile
patching file net/Config.in
patching file net/core/netfilter.c
patching file net/core/skbuff.c
patching file net/ipv4/ip_output.c
patching file net/ipv4/netfilter/ip_tables.c
patching file net/ipv4/netfilter/ipt_LOG.c
```

Now, run the configuration for the kernel (with your favorite method: make menuconfig, make xconfig ...). In the section *Networking option* enable this options:

```
[*] Network packet filtering (replaces ipchains)
[ ]   Network packet filtering debugging (NEW)
<*> 802.1d Ethernet Bridging
[*]   netfilter (firewalling) support (NEW)
```

Caution you must disable this if you want can to apply some firewalling rules else iptables doesn't work.

```
[ ]   Network packet filtering debugging (NEW)
```

After you must add the correct options in the section *IP: Netfilter Configuration*. Then, compile and install the kernel. If you want to do it the *Debian way* install the `kernel-package` and run `make-kpkg` to create a new Debian package you can install on your server (or use it on other systems). One the new kernel is installed and compiled you need to install `bridge-utils`.

Now, you can examine two different configurations which show how configuration can be done once all this steps have been taken. Both configurations are shown with a network map and the commands neccesary to configure the bridge.

D.1 A bridge providing nat and firewall capabilities

The first one uses a bridge as a firewall with network address translation that protects a server and internal LAN clients.

```
Internet ----- router ( 62.3.3.25 ) ----- bridge ( 62.3.3.26 gw 62.3.3.25 / 1
                                                |
                                                |
                                                |---- WWW Server ( 62.3.3.27 gw 6
                                                |
                                                |
                                               LAN --- Zipowz ( 192.168.0.2 gw 19
```

This commands show how the bridge can be configured.

```
# That create the interface br0
/usr/sbin/brctl addbr br0

# Add the ethernet interface to use with the bridge
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 eth1

# Just up the interface ethernet
/sbin/ifconfig eth0 0.0.0.0
```

```
/sbin/ifconfig eth1 0.0.0.0

# Configure the bridge ethernet
# The bridge will be correct and invisible ( transparent firewall ).
# It's hidden in a traceroute and you keep your real gateway on the
# other computers. Now if you want you can config a gateway on your
# bridge and choose it as your new gateway for the other computers.

/sbin/ifconfig br0 62.3.3.26 netmask 255.255.255.248 broadcast 62.3.3.32

# I have added this internal IP to create my NAT
ip addr add 192.168.0.1/24 dev br0
/sbin/route add default gw 62.3.3.25
```

D.2 A bridge providing firewall capabilities

This system is setup as a transparent firewall for a LAN with a Public IP address space.

```
Internet ----- router ( 62.3.3.25 ) ----- bridge ( 62.3.3.26 )
                                                  |
                                                  |
                                                  |---- WWW Server ( 62.3.3.28 gw 6
                                                  |
                                                  |
                                                  |---- Mail Server ( 62.3.3.27 gw
```

This commands show how the bridge can be configured.

```
# That create the interface br0
/usr/sbin/brctl addbr br0

# Add the ethernet interface to use with the bridge
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 eth1

# Just up the interface ethernet
/sbin/ifconfig eth0 0.0.0.0
/sbin/ifconfig eth1 0.0.0.0

# Configure the bridge ethernet
# The bridge will be correct and invisible ( transparent firewall ).
# It's hidden in a traceroute and you keep your real gateway on the
# other computers. Now if you want you can config a gateway on your
```

```
# bridge and choose it as your new gateway for the other computers.

/sbin/ifconfig br0 62.3.3.26 netmask 255.255.255.248 broadcast 62.3.3.32
```

If you traceroute the Linux Mail Server you don't see the bridge, if you want to access to the bridge with ssh you must have an gateway or you must to connect to another server such as the "Mail Server" and then connect to the bridge through the internal network card.

D.3 Iptables basic rules

This is an example of the basic rules that could be used for any of these setups.

```
iptables -F FORWARD
iptables -P FORWARD DROP
iptables -A FORWARD -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -m state --state IN
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Some funny rules but not in a classic Iptables sorry ...
# Limit ICMP
# iptables -A FORWARD -p icmp -m limit --limit 4/s -j ACCEPT
# Match string, a good simple method to block some VIRUS very Quickly
# iptables -I FORWARD -j DROP -p tcp -s 0.0.0.0/0 -m string --string "cmd.exe

# Block all MySQL connection just to be sure
iptables -A FORWARD -p tcp -s 0/0 -d 62.3.3.0/24 --dport 3306 -j DROP

# Linux Mail Server Rules
#

# Allow FTP-DATA ( 20 ) , FTP ( 21 ) , SSH ( 22 )
iptables -A FORWARD -p tcp -s 0.0.0.0/0 -d 62.3.3.27/32 --dport 20:22 -j ACCE

# Allow the Mail Server to connect to the outside
# Note: This *not* needed for the previous connections
# (remember: stateful filtering) and could be removed.
iptables -A FORWARD -p tcp -s 62.3.3.27/32 -d 0/0 -j ACCEPT

# WWW Server Rules
#

# Allow HTTP ( 80 ) connections with the WWW server
iptables -A FORWARD -p tcp -s 0.0.0.0/0 -d 62.3.3.28/32 --dport 80 -j ACCEPT

# Allow HTTPS ( 443 ) connections with the WWW server
```

```
iptables -A FORWARD -p tcp -s 0.0.0.0/0 -d 62.3.3.28/32 --dport 443 -j ACCEPT

# Allow the WWW server to go out
# Note: This *not* needed for the previous connections
# (remember: stateful filtering) and could be removed.
iptables -A FORWARD -p tcp -s 62.3.3.28/32 -d 0/0 -j ACCEPT
```

Appendix E

Sample script to change the default Bind installation.

This script automates the procedure of changing the name server's default installation so that it does *not* run as the superuser. Use with extreme care since it has not been tested thoroughly. The script will create the user and groups to be used for the name server.

```
#!/bin/sh
# Change the default Debian bind configuration to have it run
# with a non-root user and group.
#
# WARN: This script has not been tested throughly, please
# verify the changes made to the INITD script

# (c) 2002 Javier Fernandez-Sanguino Peña
#
#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 1, or (at your option)
#    any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#     Please see the file 'COPYING' for the complete copyright notice.
#

restore() {
# Just in case, restore the system if the changes fail
 echo "WARN: Restoring to the previous setup since I'm unable to properly cha
```

```
 echo "WARN: Please check the $INITDERR script."
 mv $INITD $INITDERR
 cp $INITDBAK $INITD
}


USER=named
GROUP=named
INITD=/etc/init.d/bind
INITDBAK=$INITD.preuserchange
INITDERR=$INITD.changeerror
START="start-stop-daemon --start --quiet --exec /usr/sbin/named -- -g $GROUP
AWKS="awk ' /start-stop-daemon --start/ { print \"$START\"; noprint = 1; }; /

[ `id -u` -ne 0 ] && {
 echo "This program must be run by the root user"
 exit 1
}

RUNUSER=`ps -eo user,fname |grep named |cut -f 1 -d " "`

if [ "$RUNUSER" = "$USER" ]
then
 echo "WARN: The name server running daemon is already running as $USER"
 echo "ERR:  This script will not many any changes to your setup."
 exit 1
fi
if [ ! -f $INITD ]
then
        echo "ERR:  This system does not have $INITD (which this script tries
        RUNNING=`ps -eo fname |grep named`
         [ -z "$RUNNING" ] && \
     echo "ERR:  In fact the name server daemon is not even running (is it in
         echo "ERR:  No changes will be made to your system"
 exit 1
fi

# Check if named group exists
if [ -z "`grep $GROUP /etc/group`" ]
then
 echo "Creating group $GROUP:"
 addgroup $GROUP
else
 echo "WARN: Group $GROUP already exists. Will not create it"
fi
# Same for the user
```

```
if [ -z "‘grep $USER /etc/passwd‘" ]
then
 echo "Creating user $USER:"
 adduser --system --home /home/$USER \
 --no-create-home --ingroup $GROUP \
 --disabled-password --disabled-login $USER
else
 echo "WARN: The user $USER already exists. Will not create it"
fi

# Change the init.d script

# First make a backup (check that there is not already
# one there first)
if [ ! -f $INITDBAK ]
then
 cp $INITD $INITDBAK
fi

# Then use it to change it
cat $INITDBAK |
eval $AWKS > $INITD

echo "WARN: The script $INITD has been changed, trying to test the changes."
echo "Restarting the named daemon (check for errors here)."

$INITD restart
if [ $? -ne 0 ]
then
 echo "ERR:  Failed to restart the daemon."
 restore
 exit 1
fi

RUNNING=‘ps -eo fname |grep named‘
if [ -z "$RUNNING" ]
then
 echo "ERR:  Named is not running, probably due to a problem with the changes
 restore
 exit 1
fi

# Check if it’s running as expected
RUNUSER=‘ps -eo user,fname |grep named |cut -f 1 -d " "‘

if [ "$RUNUSER" = "$USER" ]
```

```
then
 echo "All has gone well, named seems to be running now as $USER."
else
 echo "ERR:  The script failed to automatically change the system."
 echo "ERR:  Named is currently running as $RUNUSER."
 restore
 exit 1
fi

exit 0
```

The previous script, run on woody's (Debian 3.0) custom's bind will produce the following initd file after creating the 'named' user and group:

```
#!/bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin

test -x /usr/sbin/named || exit 0

start () {
 echo -n "Starting domain name service: named"
 start-stop-daemon --start --quiet \
     --pidfile /var/run/named.pid --exec /usr/sbin/named
 echo "."
}

stop () {
 echo -n "Stopping domain name service: named"
 # --exec doesn't catch daemons running deleted instances of named,
 # as in an upgrade.  Fortunately, --pidfile is only going to hit
 # things from the pidfile.
 start-stop-daemon --stop --quiet  \
     --pidfile /var/run/named.pid --name named
 echo "."
}

case "$1" in
    start)
 start
    ;;

    stop)
 stop
    ;;
```

```
    restart|force-reload)
 stop
 sleep 2
 start
    ;;

    reload)
 /usr/sbin/ndc reload
    ;;

    *)
 echo "Usage: /etc/init.d/bind {start|stop|reload|restart|force-reload}" >&2
 exit 1
    ;;
esac

exit 0
```

Appendix F

Security update protected by a firewall

After doing a standard installation a system might still have security vulnerabilities, if so, there might be updates available from Debian for the release. However, if you cannot download the packages for the upgrade on another system (or mirror security.debian.org yourself for local use) you need to connect to the Internet to do a security update.

However, when connecting yourself to the Internet you are exposing yourself. If one of your local services is vulnerable you might be compromised even before the update is finished! You might find this paranoid but, in fact, analysis from the Honeynet Project (http://www.honeynet.org) show than systems can be compromised in less than three days even if the system is not known publicly (i.e. not published in dns records).

When doing an update on a system not protected by an external system (a firewall) you can, however, properly configure your local firewall to only allow the security update itself. See the example below to see how the local firewall capabilities to provide a restricted setup in which only connections to security.debian.org are allowed whileas the rest are logged.

FIXME: add IP address for security.debian.org (since otherwise you need DNS up to work) on /etc/hosts.

FIXME: test this setup to see if it works properly

FIXME: this will only work with http urls since ftp might need the ip_conntrack_ftp module, or use passive mode.

```
# iptables -F
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
```

```
target     prot opt source               destination
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT DROP
# iptables -A OUTPUT -d security.debian.org -p 80 -j ACCEPT
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -p icmp -j ACCEPT
# iptables -A INPUT -j LOG
# iptables -A OUTPUT -j LOG
# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0          state RELATED,EST
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
LOG        all  --  anywhere             anywhere           LOG level warning

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     80   --  anywhere             security.debian.org
LOG        all  --  anywhere             anywhere           LOG level warning
```