Securing Debian HOWTO

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

v1.92 6 noviembre 2001Tue Oct 23 00:59:57 CEST 2001

Abstract

This document describes the process of securing and hardening the default Debian installation. It covers some of the common taks to setup a secure network environment using Debian GNU/Linux.

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 HOWTO 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 Alexander Reelsen (mailto:alex@rhwd.owl.de) and Javier Fernández-Sanguino (mailto:jfs@computer.org) and they will be incorporated into this HOWTO.

1.1 Download the HOWTO

You can download or view the newest version of the Securing Debian HOWTO 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).

1.2 Organizational Notes/Feedback

Now to the official part. At the moment I (Alexander Reelsen) wrote most paragraphs of this HOWTO, 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 HOWTO 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 HOWTO, as this document cannot explain every little detail of a feature (otherwise this would have been a book instead of a HOWTO). If you are not that familiar, however, you might want to take a look at ‘Be aware of general security problems’ on page 9 for where to find more in-depth information.

1.4 Things that need to be written (TODO)

- 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 HOWTO for any other info.
- 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
  - ...

- Information on Debian firewalling and what/how does it change from other distributions. Specifically, could mention firewalling script creators available in Debian (`mason`, `easyfw`...) and where should the firewalling code be enabled (common FAQ in debian-firewall?)

- 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`)
  - NSA Enhanced Linux (`http://www.coker.com.au/lsm/`)
  - 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`

- Add info on how the Security Team (`http://security.debian.org`) works in Debian. For example: security bugs are backported to the stable distribution if possible (even if the version has not changed), alerts are sent to mailing lists (including bugtraq)...

- 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.

1.5 Changelog

1.5.1 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

1.5.2 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!)

1.5.3 Version 1.9

Changes by Javier Fernández-Sanguino Peña.

- Added patch to fix mispellings 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 9) 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.

1.5.4 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 rootkits (‘LKM - Loadable Kernel Modules’ on page 43) 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.

1.5.5 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.

1.5.6 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.

1.5.7 Version 1.5

Changes by Josip Rodin and Javier Fernández-Sanguino Peña.

- Added paragraphs related to BIND and some FIXMEs.

1.5.8 Version 1.4

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

1.5.9 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

1.5.10 Version 1.2

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

1.5.11 Version 1.1

- Typo fixes, miscellaneous additions

1.5.12 Version 1.0

- Initial release

1.6 Credits

- Alexander Reelsen wrote the original document.
- Javier Fernández-Sanguino added more info to the original doc.
- Robert van der Meulen with 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.
- All the folks who encouraged me (Alexander) to write this HOWTO.
- 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 though 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 many 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) is one of the best references regarding general Linux Security.
- The Linux Security Administrator's Guide (http://www.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 (currently not available online, used to be at http://www.securityportal.com/lskb/ which is also provided in Debian through the `lksb` package.
- In Securing and Optimizing Linux: RedHat Edition (http://www.linuxdoc.org/links/p_books.html\hyper@hashsecuring_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.
- For network administrators, a good reference for building a secure network is the Securing your Domain HOWTO (http://www.linuxsecurity.com/docs/HOWTO/Securing-Domain-HOWTO/Securing-Domain-HOWTO.html).
- 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-H 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).

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 Proyect 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

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:

- 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...).
- 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_conract) 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 arquitectures.
- Information regarding security is centralized in a single point, http://security.debian.org/ is it.
- 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 9.

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 and change the boot sequence to disable booting from a floppy. 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.

3.2 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/apt/cache/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 an elaboration.

3.3 Set a root password

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

3.4 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`).

3.5 Run the minimum number of services required

You should not install services which are not needed on your machine. Every installed service introduces new, perhaps not obvious, but real security holes on your machine. If you still want to have some services but you use these rarely, use the update-commands, e.g. 'update-inetd' for removing them from the startup process.

FIXME: This section needs a list of services, and information about what they do and the security risk level involved, for newbies who don't have a clue.

3.6 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 17 for details on how to do this.)

Chapter 4

After Installation

4.1 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 they want 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 crypted form. The utility grub-md5-crypt generates a hashed password which is compatible with grub's crypted 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 paramater was added to instruct grub to perform the md5 authentication process. The provided password is the md5 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.

4.2 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 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

4.3 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.

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
```

4.4 Execute a security update

As soon as new security bugs are revealed 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. 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 con-
trib non-free
```

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 con-
trib non-free
```

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

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).

4.5 PAM — Pluggable Authentication Modules

PAM 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. For each application there is a configuration file in `/etc/pam.d/`.

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. 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. For example, you could restrict the number of concurrent logins users may have.

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 we want to protect su, so that only some people can use it to become root on your system, we 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
allowed onerr=fail
```

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).

4.6 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.

FIXME: Get a good limits.conf up here

4.7 Customize /etc/inetd.conf

You should stop all unneeded services on your system, like echo, chargen, discard, daytime, time, talk, ntalk and the HIGHLY insecure considered r-services (rsh, rlogin and rcp. 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 an alternative to this: `update-inetd`. 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`.

4.8 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 Editing /etc/ftpusers

This 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.

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)`.

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:

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

```
ALL: ALL: SPAWN ( \
  echo -e "\n\
  TCP Wrappers\: Connection refused\n\
  By\: $(uname -n)\n\
  Process\: %d (pid %p)\n\
  User\: %u\n\
  Host\: %c\n\
  Date\: $(date)\n\
" | /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.

4.11.1 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 wary 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 SNMP traps related to security problems (jfs). Check: `snmptraglogd`, `snmp` and `snmpd`.

4.11.2 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.

4.11.3 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.

Some logfile permissions are not perfect after the installation. 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 another UID than 0 and a group other than 'adm' or 'root'.

I want to emphasize that the apache logfile permissions are really screwed due to the fact that the apache user owns the apache log files. If a user gets a shell with a back door in apache, they can easily remove the logfiles.

4.12 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.

4.13 Using su

If you really need 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 Unixes.

4.14 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.

4.15 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 as well.

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. This might change in the woody (3.0) release.

4.16 Configuring some kernel 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 protec-
tion enabled
   echo 0 > /proc/sys/net/ipv4/ip_forward     # ip forwarding disabled
   echo 1 > /proc/sys/net/ipv4/tcp_syncookies # TCP syn cookie protec-
tion enabled
   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 al-
ways enabled
   echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
                                              # bad error message pro-
tection enabled

   # 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
```

Some users might want to set up firewalling rules in this script as well. However, check what firewalling programs/features you might be using since they can tweak other files and change the definitions you add on startup. For example, firewalk, for one, will use another configuration file for firewall setup.

FIXME: I remember a thread about this in debian-security, should add the information posted there here (jfs).

4.17 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.

4.18 Secure file transfers

Copying files in a secure manner from a host to another can be achieved by using 'scp' which is included in the ssh package. It works like rcp but is encrypted completely, so the bad guys cannot even find out WHAT you copy.

4.19 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`).

FIXME. Should we add some comments regarding `lshell` here. Is it useful? (jfs)

4.20 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.

Note that lsattr and chattr are only available on ext2 filesystems.

4.21 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 day/hour/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, except someone hacked the algorithm to create md5sums on that machine, what is, well, pathological. 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.

Chapter 5

Securing services running on your system

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. 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: `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. `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”.

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.

5.2 Securing FTP

If you really have to use FTP (without wrapping it inside a SSL 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 \*.*/`

5.3 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`

NOTE: This is the default behavior in Xfree 4.0.

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. This is done for XDM by setting `/etc/X11/xdm/Xservers` to: `:0 local /usr/bin/X11/X vt7 -dpi 100 -nolisten tcp`

5.3.1 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.

5.4 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? So 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).

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 PDQ (`http://feynman.tam.uiuc.edu/pdq`) which is based on user permissions of the `/dev/lp0` device.

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.

5.5 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.

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.

5.6 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).

5.7 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 into 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 server 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 should be considered a useful safeguard.

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.

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 --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
```

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. To achieve maximum BIND security, now build a chroot jail (see 'Using chroot' on page 24) around your daemon.

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`).

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).

FIXME (jfs): I'm not sure about this, shouldn't bind files be chown'ed to the groups created? Some files might need rw permissions in order for bind to work correctly; for example: if the name server is being used as a cache the cache files need to be written on hard disk. Also, if the DNS server is secondary, it might need to transfer zones from the primary and write them on hard disk too. This should be clarified.

5.8 Securing Apache

FIXME. Add content.

The Apache Documentation (http://www.apache.org/docs/mics/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).

5.9 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 34)

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.

5.10 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 Unixes).

5.11 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

5.12 Disabling RPC services

Last, but not least, disable RPC wherever possible. Many security holes for this service are known and can 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. Most 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 system is to do `update-rc.d portmap remove`.

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.

5.13 Automatic hardening of Debian systems

After reading through all the information in the following 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 siple yes/no questions).

5.13.1 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 know security holes that could be used by a remote attacker to compromise the system (uses versioned *Conflicts:*).
- `harden-localflaws`: removes packages with know 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 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.

5.13.2 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 6

Before the compromise

6.1 Set up Intrusion Detection.

FIXME: Write more about this.

Debian includes some tools for Intrusion Detection which you might want to setup (if truly paranoid of if your system is really critical).

Always be aware that in order 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).

6.1.1 Network based intrusion detection: Using snort

`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

6.1.2 Host based detection

`Tiger` is an old intrusion detection tool which has been ported to Debian since woody. Tiger provides check of common issues related to security breakins, 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.

Other audit tools, on-site, like `logcheck`, `portsentry` or any of the filesystem integrity checkers (see 'Checking filesystem integrity' on page 29) can be quite useful in order to set up detection of anomalies in a secured environment.

6.2 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://www.braysystems.com/linux/trustees.html`
- *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://rsmeyers.3ti.org/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`

6.3 Avoiding rootkits

6.3.1 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 rootkits (knark and adore) to install backdoors for GNU/Linux systems.

LKM rootkits can hide processes, files, directories and even connections without modifying the source code of binaries.

The adore rootkit

You can find it at `http://packetstorm.securify.com/groups/teso/adore-0.38.tar.gz`. See the README for compiling and configuring adore. Once you have already ran `./startadore` you can run `./ava`. The startadore script load the lkm rootkit and make it invisible on `lsmod` command. Now you are free to use adore, the usage of ava is very easy to understand. Just take a look at the output from ./ava.

```
Usage: ./ava {h,u,r,R,i,v,U} [file, PID or dummy (for U)]

 h hide file
 u unhide file
 r execute as root
 R remove PID forever
 U uninstall adore
 i make PID invisible
 v make PID visible
```

The knark rootkit

You can find it at `http://packetstorm.securify.com/UNIX/penetration/rootkits/knark-0.59.tar.gz` Get the source code from the address above, then compile it typing “make”. So you are ready to load the lkm rootkits, use

```
insmod knark.o
```

A hidden directory /proc/knark is created, which includes some files that will define what things will be hidden from the system, strings in /proc/net, files, pids, redirects..

The rkit toolkit

You can find it at http://packetstorm.securify.com/UNIX/penetration/rootkits/Rkit-1.01.tgz Put your UID in the rkit.c "#define magik_UID" statement as in:

```
#define magik_UID 500
```

Compile the rkit.c and load the kernel module. All the processes made by the defined UID will be hidden from the system.

6.3.2 Detecting rootkits

Detection of rootkits in Debian can be accomplished with `chkrootkit`, which can detect some of them.

You can also use SKAT (http://s0ftpj.org/en/site.html). SKAT 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?

6.4 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`.

6.4.1 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: 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 22) 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 29) 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 Proyect (`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 7

After the compromise

7.1 General behavior

If you really want to clean up residual waste, you should remove the compromised host from your network and re-install the OS 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 breakin, see Sans' Incident Handling Guide (`http://www.sans.org/y2k/DDoS.htm`)

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 mortum' of a system. `tct` allows the user to collect information about deleted files, running processes and more. See the included documentation for more information.

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

FIXME: continue the list, maybe?.

Chapter 8

Frequently asked Questions

FIXME: write them, extract from mailing list

8.1 Is Debian more secure than X?

A system is as secure as its administrator is capable of making it.

8.2 Is there are 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.

8.3 How can I make service XYZ more secure?

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).

8.4 Questions regarding users and groups

8.5 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/manuals/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 is 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 man 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.

8.5.1 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`.

8.6 Question regarding open ports

8.6.1 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 38.

8.6.2 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 ‘Customize /etc/inetd.conf’ on page 20) or by other installed packages and take appropriate measures (configure inetd, remove the package, avoid it running on bootup. . .)

8.7 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 rescue disk comes with `ae`) `/etc/shadow` and change the line:

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

  to:

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

If you are using LILO and have 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
- 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).

8.8 Questions regarding the Debian security team

8.8.1 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).

8.8.2 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.

8.8.3 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.

8.8.4 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).

8.8.5 How can I help with security?

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.

8.8.6 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.

8.8.7 How is the Security Team composed?

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

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 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 61. 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 17.
- 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 second command to work (run it as root).
- 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` 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 0
  (....)
  $ 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‘; do user=‘ps -ef |grep $i |grep -v grep |cut -f 1 -d " "‘ ; echo "Ser-
  vice $i is running as user $user"; done
  ```

and consider changing these services to a give 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 HOWTO in a condensed checklist format. This is intended as a quick summary for someone who has already read the HOWTO.

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

- Limit physical access and booting capabilities
  - Enable BIOS password
  - Disable floppy 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 some kernel features' on page 25)
  - Kernel patches (see 'Useful kernel patches' on page 42)
  - 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 29). 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`, 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 17.