

# Linux Administrator' s Security Guide

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org), Copyright Kurt Seifried 2001. [License](#) is here. Chapters with listed sections are "completed", the rest is still being rewritten.

Latest major update: [Introduction to computer security](#)

---

## Table of Contents

Linux Administrator' s Security Guide.....	1
License and Warranty.....	15
Preface.....	16
About this book.....	16
Acknowledgements .....	17
Contacts and mailing lists.....	17
Typographical conventions.....	17
Chapter Title - H1.....	17
Section Title H2 .....	17
SubSection Title H3.....	17
Bugs and errata.....	18
Dates, times and other conventions.....	18
Dates.....	18
Times.....	18
Encryption keys.....	19
Introduction to computer security.....	20
What is computer security?.....	20
Security Policy.....	21
Monitoring and Enforcement .....	21
Acceptable Use Policy.....	22
Privacy Policy.....	22
Security is a process .....	22

Defense in depth.....	23
Technical problems.....	23
Network Connectivity.....	23
Insecure defaults.....	23
Legitimate access vs.s a break in.....	23
Minimizing access and privilege.....	24
Installation.....	24
Verifying packages and files.....	24
Installation Media.....	25
CD' .s.....	25
Harddrive.....	26
Network (FTP / NFS).....	26
Automating installs.....	26
Red Hat Kickstart.....	26
Filesystem layout and structuring.....	27
Hardening your installation.....	29
Bastille Linux.....	30
Harden SuSE.....	30
Summary.....	30
Physical and console security.....	30
Physical security.....	31
Console security.....	32
BIOS / Open Firmware security.....	32
LILO security.....	33
Grub security.....	36
Rebooting the system.....	36
Summary.....	37
Administrative tools.....	37
Local tools.....	37
YaST.....	38
sudo.....	38
Super.....	38
WWW based tools.....	39
Webmin.....	39
Linuxconf.....	39
Other network based tools.....	40

Pikt.....	40
VNC.....	40
cfengine.....	40
Non-commercial backup programs for Linux.....	41
Tar and Gzip or Bzip2.....	41
rsync.....	42
Amanda.....	42
Commercial backup programs for Linux.....	42
BRU.....	42
Quickstart.....	43
Backup Professional.....	43
CTAR.....	43
CTAR:NET.....	43
PC ParaChute.....	43
Legato Networker.....	43
Backup media.....	44
Basic file commands.....	45
Secure file deletion.....	47
wipe .....	47
fwipe.....	47
Access control lists (ACL's).....	48
NSA SELinux.....	48
RSBAC.....	48
Critical system configuration files.....	48
/etc/ directory.....	48
File encryption.....	50
PGP.....	50
GnuPG.....	51
Filesystem encryption.....	53
TCFS.....	53
BestCrypt.....	53
PPDD.....	55
Hiding data.....	56
StegHide.....	57
StegFS.....	57
OutGuess.....	57

RubberHose.....	57
Authentication.....	58
PAM.....	58
PAM Cryptocard Module.....	59
Pam Smart Card Module.....	59
Pam module for SMB.....	59
Authentication services.....	59
Passwords .....	60
Use a better hash.....	60
Use shadow passwords.....	60
Cracking passwords.....	61
VCU.....	61
Password storage.....	61
Strip.....	62
Log files and other forms of monitoring.....	62
General log security.....	63
System logging.....	63
sysklogd / klogd.....	63
modular syslog.....	65
next generation syslog.....	65
Nsyslogd.....	65
Log monitoring.....	65
Psionic Logcheck.....	66
colorlogs .....	66
WOTS.....	66
swatch.....	66
Kernel logging.....	66
auditd.....	67
Shell logging.....	67
bash.....	67
Attack detection.....	68
Baselines.....	68
File system monitoring.....	69
Tripwire.....	69
AIDE.....	69
ViperDB .....	69

Pikt.....	70
Backups.....	70
Network monitoring / attack detection.....	70
DTK.....	71
Psionic TriSentry - PortSentry, HostSentry and LogSentry.....	71
scanlogd.....	71
Firewalls.....	71
TCP-WRAPPERS.....	72
Intrusion Detection Papers.....	72
Dealing with attacks.....	72
Packet sniffers.....	73
Snort.....	73
tcpdump.....	73
Ethereal.....	73
SPY.....	74
Other sniffers.....	74
Packet sniffer detection.....	74
AntiSniff.....	74
Intrusion testing - scanning / intrusion tools.....	75
Host scanners.....	75
Network scanners.....	75
Nmap.....	76
Firewalk.....	76
ICMP related scanning.....	76
spidermap.....	76
Application level Scanners.....	76
Nessus.....	77
Saint.....	77
Ftpcheck / Relaycheck.....	77
SARA.....	77
BASS.....	78
Exploits.....	78
Firewalling .....	78
Firewall software for Linux.....	78
Firewall concepts.....	79
IPTables.....	80

IPChains.....	80
IPFWADM.....	80
Firewall piercing.....	80
Firewalling with IPTables.....	81
A very basic example .....	81
Some more tricks with IPTables.....	82
Invalid packets.....	82
Listing Tables.....	83
Dropping every second or third packet.....	83
Network security.....	83
Physical protocols.....	84
TCP-IP security.....	84
IPSec.....	85
IPv6.....	85
Attacking TCP-IP.....	85
HUNT Project.....	85
Basic config files and utilities.....	85
/etc/inetd.conf.....	86
/etc/services.....	86
TCP_WRAPPERS.....	86
What is running and who is it talking to?.....	88
ps .....	88
netstat .....	89
lsof.....	91
Encryption of network traffic (non VPN).....	91
SSL.....	91
Routing security.....	91
routed.....	92
gated.....	92
MRT.....	92
zebra.....	92
Network Based Authentication.....	93
Overview.....	93
NIS / NIS+.....	93
Kerberos.....	94
Radius.....	94

xtradius.....	94
gnu-radius.....	95
perlradius.....	95
Cistron RADIUS server.....	95
FreeRADIUS.....	95
ICRADIUS.....	95
Certificate authority software for Linux.....	95
Overview.....	96
Certificate authority software.....	96
OpenCA.....	96
pyCA.....	96
Network services - DHCP.....	96
Overview.....	96
DHCP servers.....	98
ISC DHCPD.....	98
Moreton Bay DHCP Server.....	99
Network services - DNS.....	100
Overview.....	100
DNS servers.....	100
Bind.....	100
Dents.....	103
Email servers.....	103
Overview.....	103
Sendmail.....	104
Postfix.....	107
Sendmail Pro.....	109
QMAIL.....	109
Zmailer.....	109
DMail.....	109
nullmailer.....	109
POP servers.....	110
UW IMAPD (contains the default popd for most distros).....	110
Cyrus.....	110
IDS POP.....	110
Qpopper.....	111
IMAP.....	111

UW IMAPD (contains the default imapd for most distros).....	111
Cyrus.....	111
Courier-IMAP.....	112
Scanning email for viruses.....	112
Protector.....	112
AMaViS.....	112
Enhancing E-Mail Security With Procmail.....	113
SSL wrapping POP and IMAP servers.....	113
Non-commercial mailing list software.....	114
SmartList.....	115
Majordomo.....	115
Minordomo.....	115
Sympa.....	115
Listar.....	115
File / print servers.....	116
Overview.....	116
Network booting.....	116
tftp .....	116
UNIX file sharing.....	116
NFS.....	116
rsync.....	116
Printing under Linux.....	116
lpd.....	117
CUPS.....	117
LPRng.....	117
pdq.....	117
Windows file and print sharing.....	117
Samba.....	117
General file sharing.....	117
Coda.....	117
Drall .....	118
AFS.....	118
Source code sharing.....	118
CVS.....	118
Network services - FTP.....	118
Overview.....	118

FTP servers.....	120
vsftpd.....	120
ProFTPD.....	121
WU-FTPD.....	123
NcFTPD.....	125
FTP - SSL.....	125
FTP - SRP.....	125
sftp.....	125
Linux LDAP servers.....	125
Overview.....	125
LDAP servers.....	126
OpenLDAP.....	126
LDAP authentication.....	126
NSS LDAP Module.....	126
LDAP tools.....	126
web2ldap.....	126
kldap.....	126
GQ.....	126
LDAPExplorer.....	126
Perl/Java/C SDK' for LDAP.....	127
Network services - NNTP.....	127
Overview.....	127
NNTP server software.....	127
INN.....	127
Diablo.....	128
DNews.....	128
Cyclone.....	129
Typhoon.....	129
Proxy software.....	129
SQUID.....	130
Cut the crap.....	133
WWWOFFLE.....	133
Circuit level proxy software.....	133
SOCKS.....	133
Dante.....	133
Shell servers.....	133

Telnet.....	134
Telnet - SSL.....	136
SSLtelnet and MZtelnet.....	136
SSH - server and client software.....	136
OpenSSH.....	137
SSH - client software: .....	139
Fresh Free FiSSH.....	139
Tera Term.....	139
putty.....	139
mindterm.....	140
The Java Telnet Application.....	140
Secure CRT.....	140
Fsh.....	140
SSH Win32 ports.....	140
SRP.....	140
NSH.....	140
R services.....	141
SNA connectivity.....	141
Network services - SNMP.....	142
Overview.....	142
SNMP server software.....	143
Network services - NTP.....	143
NTP server software.....	144
XNTP.....	144
NTP client software.....	144
ntpdate.....	144
User information.....	145
Overview.....	145
Ident server software.....	145
Identd.....	146
Other Identd daemons.....	147
Finger server software.....	147
PFinger.....	148
Network services - HTTP / HTTPS.....	148
Overview.....	148
WWW server software.....	149

Apache.....	149
Secure configuration of Apache.....	150
thttpd.....	151
AOL Server.....	151
webfs.....	151
Flash Web Server.....	151
Secure WWW server software.....	151
Apache-SSL.....	152
Apache with mod_ssl.....	152
Red Hat Secure Server.....	153
Roxen.....	154
Zeus.....	154
Netscape Enterprise.....	154
IBM HTTP Server.....	154
Accessing your WWW server files.....	154
FTP.....	154
Samba access.....	155
FrontPage access.....	155
RearSite.....	155
WWW based email readers.....	156
Overview.....	156
Non-commercial .....	156
AtDot .....	156
acmemail .....	156
IMHO .....	156
IMP .....	156
MailMan.....	157
SquirrelMail.....	157
Commercial .....	157
DmailWeb .....	157
Webmail.....	157
X Window System.....	157
Overview.....	158
X server security configuration.....	158
xhost .....	158
mkxauth .....	159

SSH tunnel.....	159
Software.....	159
RPM .....	160
dpkg.....	161
tarballs / tgz .....	162
Checking file integrity.....	162
RPM integrity.....	163
dpkg integrity.....	163
PGP signed files.....	163
GnuPG signed files.....	163
MD5 signed files.....	163
Automating software updates.....	164
NSBD.....	164
Automating updates with RPM.....	164
AutoRPM.....	164
RpmWatch.....	164
Automating updates with dpkg.....	165
Automating updates with tarballs / tgz.....	165
Tracking software installation.....	165
instmon.....	165
Converting file formats.....	166
alien.....	166
Finding software.....	166
Secure programming.....	166
Secure Linux Programming FAQ.....	166
Secure UNIX Programming FAQ.....	167
Secure Internet Programming.....	167
Writing Safe Setuid Programs.....	167
userv.....	167
More.....	167
Testing software.....	168
fuzz.....	168
Compiler patches.....	168
Stackguard.....	169
Encryption.....	169
Encrypting data files and email.....	169

GnuPG (Gnu Privacy Guard).....	170
pgp4pine.....	170
Netscape Messenger.....	170
Sources of random data.....	170
Limiting and monitoring users.....	171
Limiting users.....	171
PAM.....	171
Bash.....	173
Quota.....	174
Monitoring users.....	175
sshsniff.....	176
dsniff.....	177
Firewalling.....	177
Viruses.....	178
Disinfection of viruses / worms / trojans.....	178
Virus Scanners for Linux.....	179
Sophos Anti-Virus.....	179
AntiVir.....	179
InterScan VirusWall.....	179
F-Secure Anti-Virus.....	179
AVP.....	180
Virus scanning of email.....	180
Virtual private networks.....	180
IP Security (IPSec).....	181
PPTP (Point to Point Tunneling Protocol).....	181
CIPE (Crypto IP Encapsulation).....	181
Zbedee.....	181
The Linux kernel.....	182
Compiling and installing a kernel:.....	182
Kernel versions.....	184
Kernel patches.....	184
Secure Linux kernel patch.....	184
International kernel patch.....	184
Linux Intrusion Detection System Patch (LIDS).....	185
RSBAC.....	185
LOMAC .....	185

auditd.....	185
Fork Bomb Defuser.....	185
Debugging the Linux kernel.....	185
KDB v0.6 (Built-in Kernel Debugger) .....	186
kGDB (Remote kernel debugger).....	186
Security techniques.....	186
Checklists.....	187
Appendices.....	188
Appendix A: Books and magazines.....	188
Appendix C: Other Linux security documentation.....	189
Appendix D: Online security documentation.....	190
Appendix E: General security sites.....	191
Appendix F: General Linux sites.....	192

---

Last updated on 1/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## License and Warranty

The OpenContent (OC) is defined as the set of webpages comprising the "Linux Administrator' s Security Guide" (primary location at <http://seifried.org/lasg/>). The OpenContent license is available from <http://www.opencontent.org/>.

### LICENSE

#### Terms and Conditions for Copying, Distributing, and Modifying

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

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

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

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

## **NO WARRANTY**

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

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

[Back](#)

Last updated on 27/7/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Preface**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

## **About this book**

I wrote this book originally because no Linux security documentation existed back in the late 1990' .sSince then several Linux security books have been published, but in general they all have shortcomings (for example one of them spends 40 pages on cops, a largely obsolete tool). So anyways I' ve decided it's time to update the book, which is one major advantage of doing this on the www. Currently it is April 2001, I' m hopig the major rewrite will be done by fall or winter of 2001 and then the LASG can go back into "maintenance" mode for a while. Writing the preface is boring so I'm gonna stop here and you can go read the book itself.

## Acknowledgements

I'd like to thank my family for being supportive and putting up with me.

## Contacts and mailing lists

There is an announcement mailing list at: <http://lists.seifried.org/mailman/listinfo/lasg-announce/>. Semi-regular postings are made to announce new updates, sections and any major changes.

There is a discussion list for the LASG available at: <http://lists.seifried.org/mailman/listinfo/lasg-discuss/>. Appropriate topics include feature requests, errors, expanding on content, linux security, etc.

You can contact the author directly at: [kurt@seifried.org](mailto:kurt@seifried.org).

## Typographical conventions

Since this is rapidly approaching "real book status" I decided it was time to format it nicely.

### Chapter Title - H1

Only one chapter title per chapter (individual files), linked from TOC.

### Section Title H2

Multiple section titles per chapter, linked from TOC

### SubSection Title H3

<b>Bold</b>	Hostnames, Usernames, IP' s, UID/GID's
-------------	--

<i>Italic</i>	file and directory names, program and command names, command-line options, email addresses and pathnames, site names and new t
Formatted	Examples of commands
Formatted <i>Italic</i>	variable options, keywords, text the user replaces with their value
Formatted <b>Bold</b>	commands and text to be literally typed by the user

## Bugs and errata

The beauty of this book is that it is available online, thus bugs and errata are done in real time. What you read is generally the most up to date version. If you find a problem please tell me, [kurt@seifried.org](mailto:kurt@seifried.org).

## Dates, times and other conventions

This book refers to dates and times in numerous places. Unfortunately there are no universally used conventions for simple things like writing dates, which is rather annoying. I have decided to use "international" standards as opposed to American/Canadian standards, while this may cause some confusion it makes sense to me.

### Dates

Dates will be written using the ISO standard of Day first, then the month, and finally the year. For example the date August 21, 2001 would be written:

21/8/2001

Long dates will be written the day first, then the month, then the date and finally the year. For example the date August 21, 2001 would be written:

Tuesday, August 21, 2001

### Times

Time references are in GMT unless otherwise noted. References to duration are in the form hours, minutes and seconds (HH:MM:SS). A time of one hour, twenty minutes and thirty seconds would be written:

1:20:30



```
bDgNRR0PfIizHHxbLY7288kjwEPwpVsYjY67VYy4XTjTNP18F1dDox0Ybn4zISy1
Kv884bEpQBGRjXyEpwpy1obEAXnIByl6ypUM2Zafq9AKUJScRtMIPWakXUGfnHy9
iUsiGSa6q6JewlXpMgs7AAICCAD1mLQv5THh1JfuQEN26KbdRXWtw5tJ2LiXri17
G1BGS4pz7CVgNIhmKxhm9xvTD7Yb0xi2RoA5yre04xG77OQ47k0IjawSHdfr+PBZ
8C7003QS17vKHthrpKayKENOUqWKOK3jGd2fx50EgKMnt5o+n1szEuhwvmxh1lOp
iV414EMc2QykM1W/weTgCmTvBVABfgm0OQoNswdkrKPyY16Li2IBI9ebqo6Vnz8
NWiz2Hzta0cKvuGak/mmNkLsZFXQ3oH/J6ubRb9LskqJ4o7SwUaCAHR1sjlq5LS/
JNVjwkG18Q+Jrr4X6NncRK1eCuHm8yD5dbvHPZi0VnltXHwsiQBMBBgRagAMBQI7
DXMYBQkQMOwAAAoJEK1jC06tVuV0vHwAmwTOfoVT5Rjqa1uoEvXy7qpRjnzUAKCw
4DM73//OxJSRLTwVO5IVtq/WIQ==
=azr0
-----END PGP PUBLIC KEY BLOCK-----
```

---

[Back](#)

Last updated on 31/8/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Introduction to computer security

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

## What is computer security?

Security is risk management. - unknown

"A computer is secure if you can depend on it and it's software to behave as you expect" - Practical UNIX and Internet Security

Security is: availability, consistency, access control, data confidentiality, authentication. - <http://www.sun.com/security/overview.html>

"The principal objective of computer security is to protect and assure the confidentiality, integrity, and availability of automated information systems and the data they contain." - <http://csrc.nist.gov/publications/secpubs/csaw.txt>

There are numerous definitions for "computer security", and most of them are correct. Essentially computer security means enforcement of usage policies, this must be done since people and software have flaws that can result in accidents, but also because someone may want to steal your information, use your resources inappropriately or simply deny you the use of your resources.

## Security Policy

A security policy is an expression of your organizations security goals. Security goals will vary greatly by organization, for example a software vendor is typically more concerned about the confidentiality and integrity of their source code anything else, while a pornographic website is probably more concerned about processing credit cards online. There is an immense amount of security technology available, from software and hardware to specific techniques and implementations. You cannot properly implement the technology without a solid idea of what your goals are. Do home users connecting to the office need to use VPN software? If your security policy states that all file and print transfers must either be encrypted or sent across a "trusted" network then the answer is probably yes. What authentication methods are acceptable for services that can be reached from the Internet? Do you require strong passwords, tokens, biometric authentication, or some combination? The data you collect from remote sites, is it more important that this data is collected, or is it more important that this data remain secret? Data that is remote weather telemetry is probably not as sensitive as credit card numbers.

Some security policies will need to be exceptionally broad and general, for example the security policy for JANET, the academic backbone network for England states:

### **Monitoring and Enforcement**

19. It follows from the policy of cascaded responsibility backed up by written site agreements, that there must be some method for UKERNA to enforce the possible disconnection envisaged by the AUP, and to provide full access and assistance to law enforcement agencies where necessary.

20. The JANET-CERT has therefore been given the responsibility (in conjunction with UKERNA) to

- Monitor use of the network, as far as is possible while respecting privacy, either in response to information about a specific threat, or generally because of a perceived situation
- Require a primary site, through its nominated contact, to rectify any omission in its duty of responsibility
- Where a site is unable or unwilling to co-operate, report the issue to UKERNA and initiate the procedure for achieving an emergency disconnection
- Obtain evidence and pass on information as necessary in order to assist an investigation by a law enforcement agency
- Provide support and co-ordination for investigations into breaches of security

On the other hand a security policy can be fine grained:

All internal email must be encrypted with PGP or GnuPG using a key of at least 1024 bits that is signed by an authorized signing entity.

Generally speaking the more detailed and technology oriented a security policy is the harder it will be to follow and keep up to date. The actual technical details of implementing a security policy should be separated from it. Keeping a separate set of best practices or an actually "implementation of security policy" document is a better idea than rolling it all into the security policy.

## **Acceptable Use Policy**

Another component of computer security is an AUP (Acceptable Use Policy). This is a document that states what a user of your resources may or may not do with them. Typically it is part of a contract and is signed at the time when the services are purchased. Many acceptable use policies generally forbid actions that are illegal, potentially annoying to other people (and hence likely to cause problems for the provider in the form of complaints) or are controversial (such as pornography). Other standard clauses include "this notice may change at any time without warning" and "we can terminate your service (or fire you) at any time without warning if you violate this policy. Generally speaking the majority of AUP's are reasonable and will not be a problem for normal users. These are a good complement to a security policy as they set out in concrete terms what a user is or is not allowed to do, and as they are often part of a contract it allows a provider to enforce their security policy when a user attempts to violate it or has violated it.

## **Privacy Policy**

Privacy policies are interesting in that they are supposed to prevent an organization (typically a company) from violating some aspects of a user's security (specifically the confidentiality of their information). Unfortunately the majority of privacy policies contain clauses like "this policy may change at any time without warning" or are simply discarded when a company decides to profit off of a user's information (such as name, address, credit card details, purchase history, etc.).

## **Security is a process**

You only need to make one mistake or leave one flaw available for an attacker to get in. This of course means that most sites will eventually be broken into. Witness the effects of Code Red, an Nimda, both of which were hugely successful exploiting well known and long solved vulnerabilities in the Microsoft IIS server. Regularly apply patches (after testing them). Regularly scan your network for open ports. Regularly scan your servers with intrusion testing software such as Nessus. Audit file permissions and make sure there are no unexpected setuid or setgid binaries.

## **Defense in depth**

All technical security measures will eventually fail or be vulnerable to an attacker. This is why you must have multiple layers of protection. Firewalls are not enough, you may accidentally forget a rule or leave yourself exposed while reinitializing a ruleset, use `tcp_wrappers` to restrict access to services as well where possible. Restrict which users can access these services, especially if only a few users need access. Encrypt traffic where possible so that attackers cannot easily snoop usernames and passwords or hijack sessions. Since security measures will fail you also need a strong audit and logging process so that you can later find out what went wrong and how bad it is.

## **Technical problems**

These are just a handful of thousands of specific technical problems facing security administrators.

### **Network Connectivity**

One of the biggest security challenges is the increase in network connectivity. If you have a machine that is not connected to any other machines an attacker will generally need to gain physical access. This of course greatly narrows down the number of attackers. However with everything connected to the Internet there are over 100 million people that can potentially get into your machine.

### **Insecure defaults**

This is one of the problems that has caused no end of security problems since day one. Vendors typically ship their operating systems with insecure defaults (i.e. `finger`, `telnet`, `ftp`, etc.) meaning that administrators must expend a lot of effort to close security problems before they can even start to pro-actively secure their systems and networks.

### **Legitimate access vs.s a break in**

Because you must grant legitimate users access to resources there is always the potential for attackers to gain access. Attacker can guess authentication credentials (i.e. weak passwords), steal them (password sniffing), exploit flaws in the server itself and so on.

## Minimizing access and privilege

When possible restrict access. If you do not need to run fingerd turn it off and remove it. An attacker cannot exploit fingerd if it isn' present. Keep users off of servers if possible, if they need shell accounts for some reason setup a separate system and partition it off from the rest of your network. Lock down workstations where possible, set BIOS passwords, secure the boot sequence, do not give them administrative access.

---

[Back](#)

Last updated on 1/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Installation

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

How you install Linux will have a big impact on how much you will be able to secure. If you install Linux from an untrusted source you could potentially end up with an installation that has backdoors or other security issues. If you install packages you do not need and forget about there is a greater chance that someone will be able to break in. As well there are security tasks that are best accomplished during installation, attempting to do them post install can be troublesome.

## Verifying packages and files

Some sources of install packages and files are safer then others. Buying Linux on CD from a large vendor such as Red Hat at a large store is relatively safe and the chances of acquiring a trojan'ed CD set is very unlikely. Installing Linux via ftp from a remote ftp server provides an attacker many avenues to replace packages with modified versions. The good news is that most vendors have addressed this problem.

Using public/private key cryptography most vendors have a keypair that they use to sign packages, you can verify these signatures with the public

key. These public keys can be downloaded online and most vendors include them on their CD's. A list of vendor key names, ID's, fingerprints and the keys themselves are available [here](#).

You first need to download the vendors PGP or GnuPG key, for example Red Hat's signing key has the identity of "security@redhat.com" and the fingerprint is:

```
CA20 8686 2BD6 9DFC 65F6 ECC4 2191 80CD DB42 A60E
```

Generally speaking you can download the vendor keys from their websites, and many vendors ship the key on their CD's. To verify a directory full of RPM's this simple script will automate the task:

```
[user@server RPMS]# for a in *.rpm
> do
> rpm -K $a >> ~/sign-log
> done
```

The file sign-log should consist of lines like:

```
ElectricFence-2.2.2-5.i386.rpm: md5 gpg OK
ImageMagick-5.2.2-5.i386.rpm: md5 gpg OK
ImageMagick-devel-5.2.2-5.i386.rpm: md5 gpg OK
Inti-0.5preview-1.i386.rpm: md5 gpg OK
```

## Installation Media

You cannot secure your installation very well if an attacker manages to compromise the system before you even install it. While it is rare for an attacker to try and compromise installation media it is certainly possible. The first step is to verify the packages and installation files on your installation media, this can be done by checking GnuPG signatures on the RPM's/etc, and the other files, although finding the signatures of the other files or MD5 sums is relatively tricky it is possible. The next step is to secure the system providing the files and the path between this system and the machine you are installing the software on. This is very easy for example with a CD-ROM, and much more tricky if doing an FTP or NFS install over the Internet from a system you do not control. It is strongly suggested you use a server that you control to host installation files.

### CD's

CD's are typically the easiest to install from, and relatively secure. If you purchase CD's from the vendor it is unlikely that they have been somehow compromised, all you do is simply boot from them and install the software. You can leave the machine offline for the installation typically, and if you burn the updates onto a CD you can easily update the system, again without needing to bring it online. It is strongly recommended to keep a set of

vendor CD's around so that you can cleanly install critical systems as needed.

## **Harddrive**

Installing from a local harddrive is also a relatively secure way of installing Linux. Simply partition the drive first (if you are installing Linux to it) and then copy the files onto a prepared partition, boot the system and away you go. Installing from the harddrive provides many of the benefits of installing from CD without the need to burn CD's. As well you can easily copy all the updates to the drive and then install them on the first reboot and bring the system completely up to date. Additionally with the increased availability of external harddrives utilizing parallel, FireWire and USB ports it is relatively easy to plug a harddrive into a system without needing to take it apart (assuming of course the installation floppy disk supports it).

## **Network (FTP / NFS)**

These two methods are very convenient, all you need is a single floppy disk (typically) and a server with all the files. Simply mirror the directory structure you need (typically something like /pub/redhat/7.1/en/i386) and then make it accessible via ftp (anonymous or username/password required) or via NFS (to the IP or subnet you use for installations). The files only need to be readable, there is absolutely no need for writing to them and in fact you are probably better off removing the write bit for all to prevent "accidents". These network methods can also be combined with the automated installations such as KickStart, or you can do your own custom installation of sorts (by removing packages you do not want for example). If you choose to do network installs it is critical that the NFS/FTP server remains secure, to this end I highly recommend using a protected subnet, not attached to your main network if possible.

## **Automating installs**

Automating installs can relieve tedium and prevent security problems. If you have to do 100 Linux installs chances are you will make error if you have to do them each manually. However if you can create an automated installation that is secured then it is much less likely that you will have problems.

## **Red Hat Kickstart**

Red hat provides a facility for automating installs, which can be very useful. Simply put you create a text file with various specifications for the

install, and point the Red Hat installer at it, then sit back and let it go. This is very useful if rolling out multiple machines, or giving users a method of recovery (assuming their data files are safe). You can get more information at: <http://www.redhat.com/mirrors/LDP/HOWTO/KickStart-HOWTO.html>. The configuration file can also be placed on a tftp server, in a directory named for the IP address of the client, thus using a dhcp/tftp server you can completely automate installs for machines if you know the MAC address of the client machine.

## Filesystem layout and structuring

There are a number of common attacks/exploits in Linux (and UNIX) that can be reduced in risk and significance with a proper directory and partition structure. One of the most common denial of service attacks is to fill up the disk space with junk data, this can also happen unintentionally with software that experiences a problem. This is typically defended against by assigning root a set percentage of the disk space as reserve (usually 5-10%), so on a 10 gig disk users would not be able to use the last 500 megs, this would be reserved for root. This doesn't do you any good however if something running as root, that generates log files, goes nuts, or is attacked and made to generate massive log files. The next big attack that takes advantage of disk setup would be /tmp races, and core dumps, programs that create links or files improperly without checking to make sure they exist first, especially programs that run as root. An attacker can then link /tmp/foo to /etc/passwd and potentially add a user account, wipe the password database out, and so on.

Mounting options can be used to mount a partition read only, not allow execution of programs, and other useful things. You may encounter difficulties when using these options however, for example if you mount /usr as read only you will have significantly more work when upgrading system components, especially on critical servers (such as e-commerce machines) that need to be up and running, but also require critical updates. More useful options are "nosuid" (no setuid or setgid files), "noexec" (no executables) and "nodev" (no device files).

So with this in mind we have several guidelines:

- Put filesystems that users can write to on separate partitions
- Put filesystems with critical system components/configuration on separate partitions
- Consider mounting some partitions with system binaries as read-only (this will make upgrades more difficult however), mounting /bin/, /sbin/ and /etc/ separately however will make booting the system tricky (depending on your configuration), test this before using it in a production environment.

Some notes on the various flags

noexec, if you mount /tmp noexec for example you can copy a binary in, but it will not run, however if you execute it using ld-linux.so it will work fine:

```
[seifried@stench /tmp]$ ./date
bash: ./date: Permission denied
[seifried@stench /tmp]$ /lib/ld-linux.so.2 ./date
Thu Aug 24 21:59:08 MDT 2000
[seifried@stench /tmp]$
```

Dir	no dev	noexec	nosuid	read-only	separate	comments
/	yes	yes	yes	yes	good idea	Ideally you should mount / totally restricted and then have directories like /boot/ separate, this also forces you to configure the directories properly since any "dangerous" directory (for example /dev/) will be "broken" (i.e. noexec would severely break /dev/). This is only recommended if you are going all out.
/boot/	yes	yes	yes	ok	good idea	Critical directory with kernel images, if an attacker replaces your kernel or deletes it you have a lot of problems.
/bin/	yes	no	no	ok	tricky	Directory with important system binaries, do not mount noexec or nosuid, your system will not work correctly. Mounting read-only will prevent trojans, and make updating software significantly more difficult.
/dev/	no	yes	yes	no	yes	Mounting /dev/ with the nodev option will severely break your system, as will mounting it read only. /dev/ is usually less than a few megs, and the ability to write to device files can result in huge damage, and system compromise.
/etc/	yes	yes	yes	no	tricky	Critical directory with system configuration information, usually the first target for an attacker. There should be no binaries in here (although some Unix systems do keep binaries in /etc/, Linux is not one of them). Mounting it read only will not allow you to change passwords, or other important settings so is not recommended.
/home/	yes	good idea	yes	no	yes	/home/ is the primary area where users keep their files and work with them (assuming they can log in), if you provide services like IMAP this is where their mail folders will be. You should make it a separate partition since users have a tendency of eating up space rapidly, as well it will prevent them from making hard links to files and then using setuid programs that dump core for example and wiping out system files. Mounting it noexec is probably a good idea, however depending on the type of work users do it may seriously hamper them, mounting it nosuid is a good idea and shouldn't really affect users.
/lib/	yes	no	yes	ok	good idea	Most programs will rely on libraries in this directory, if they are damaged or compromised you will have a hard time cleaning up. There are executable files in here (.so's, etc.), so setting it noexec would not be advised, but setting it nosuid is probably wise.
/mnt/	yes	good idea	good idea	ok	no need	/mnt/ is typically used to mount removable devices, such as /mnt/floppy/ or /mnt/cdrom, as such it rarely contains anything other than a few directories, making it separate is not a real issue since anything in it will be mounted as well.

/opt/	yes	no	no	no	good idea	Typically this directory is used for optional packages, vendor software and the like, oftentimes this stuff needs setuid bits to work properly (a good reason to separate it since a lot of vendors have terrible software security).
/proc/						/proc/ is a virtual filesystem
/root/	yes	no	no	no	good idea	root's private playground usually, many people use it instead of /usr/local/ when testing things/etc
/sbin/	yes	no	no	ok	tricky	Directory with other important system binaries, do not mount noexec or nosuid or you will break your system. Mounting read-only will prevent trojans, and make updating software significantly more difficult.
/tmp/	yes	yes	yes	no	yes	Temporary directory for use by users and system, mount read only will break things, make it separate because many exploits consist of making hard links in tmp to files, and then having a program misbehave and destroy/modify the target file maliciously. Binaries, especially setuid binaries should not be allowed in /tmp/ since any user can modify them usually.
/usr/	yes	no	no	ok	good idea	This directory is where the majority of software will be installed, along with source code and other stuff typically, mounting it separately is a good idea since it tends to contain relatively important software (especially in /usr/bin and /usr/sbin). Mounting it read only will prevent an attacker from inserting trojan software, but will make upgrades significantly harder. I wouldn't bother mounting it read only unless you also mount /bin/ and /sbin/ read only.
/var/	yes	yes	yes	no	yes	/var/ is used for a lot of things, least of which includes system logging. This partition should be separate since attackers can attempt to fill it up by flooding the log files, and other user data is stored here, such as mail (/var/spool/mail usually). Software that stores data here includes: Mail servers (Sendmail, Postfix, etc.), INN (Usenet news), Proxy software like Squid (WWW/FTP proxy), and so on. There should be no binaries at all here, just log files and data. Setting it noexec may break programs, Red Hat 7.0 places various binaries used for anonymous ftp with WuFTP and arpwatc binaries in /var/ for example. You can place those files on another partition and symlink the directories to within /var/.

## Hardening your installation

So you've got a fresh install of Linux (Red Hat, Debian, whatever, please, please, DO NOT install really old versions and try to upgrade them, it's a nightmare), but chances are there is a lot of extra software installed, and packages you might want to upgrade or things you had better upgrade if you don't want the system compromised in the first 15 seconds of uptime (in the case of BIND/Sendmail/etc.). Keeping a local copy of the updates directory for your distributions is a good idea (there is a list of errata for distributions at the end of this document), and making it available via

NFS/ftp or burning it to CD is generally the quickest way to make it available. As well there are other items you might want to upgrade, for instance imapd or bind. You will also want to remove any software you are not using, and/or replace it with more secure versions (such as replacing RSH with SSH).

## **Bastille Linux**

If you are running Mandrake / Red Hat or a similar Linux you can use the Bastille Linux hardening script available at: <http://www.bastille-linux.org/>. It will disable various servers, install login banners and generally automates many of the tasks a security administrator will have to do in any event.

## **Harden SuSE**

Harden SuSE is a script specifically for SuSE Linux and available from: <http://www.suse.de/~marc/>. Usage can be automated.

## **Summary**

Linux, like most UNIX systems uses a directory structure based off of /, which directories like /home, /tmp, /usr, and so on. These directories can be placed on the same partition, or separate partitions, separating them properly can have security (and performance) benefits. There are also a number of mounting options that can be used to prevent common problems and attacks.

---

[Back](#)

Last updated on 31/8/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Physical and console security**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

While the majority of random and remote attacks come in over the network physical and console security are important factors. In a perfect world every machine would be physically secure with access to the console (i.e. keyboard, reset switch and monitor) tightly controlled. Unfortunately this is not a perfect world it is rare to find a physically secure machine outside of a server room.

## Physical security

Remember that an attacker may not want to break into your desktop machine or network, they may be looking for a quick way to make \$200, and stealing a desktop computer is one way to do that. All systems should be securely fastened to something with a cable system, or locked in an equipment cage if possible. Case locks should be used when possible to slow attackers down (thefts of ram and CPU' are becoming increasingly popular). Some systems, like Apple G4's, when cable locked cannot be opened, if you need machines for public areas features like this are ideal. For secure rooms make sure that the walls go beyond the false ceiling, and below the raised floor, large vents should also be covered with bars if possible. Monitoring the room with CCTV and possibly requiring a remote person to "buzz" you in (in addition to a lock, keypad or other access control device) is recommended.

With enough time and physical access an attacker will be able to gain access to the system, several methods of attack include:

- Rebooting the system from other media such as floppy disk, CD-ROM, external SCSI drives and so on
- Removing the case, and removing the BIOS battery to get around any BIOS restrictions
- Using a default BIOS password to gain access to the BIOS
- Rebooting the system and passing boot arguments to LILO
- Installing physical monitoring devices such as KeyGhost
- Stealing the system's disk(s)
- Unplug the server, or turn the power bar off (a very effective DoS), if done several times this can lead to filesystem corruption.

These are just a few of many possible attacks. The primary goal is to stop them where possible, and failing that slow them down so that hopefully someone will notice the attacker tearing apart a system in someone's office. The installation of monitoring devices is becoming especially worrisome, as they are now available for purchase online for less than \$100. An attacker can easily log all keystrokes for a long time period before the attacker comes back to retrieve them. Periodic physical inspections (in teams of two) of user machines for items like KeyGhost, modems are so on are a good idea.

Gaining access to office buildings is often trivial. While working for the government there was no access control to the building itself from 8 A.M. until 5 P.M. (after 5 P.M. a security guard forced you to sign in). Worse yet the building had a back entrance that was not monitored. If you were to enter at 4:30 P.M., hide in a bathroom for an hour or two (crouched on top of a toilet reading the latest Linux Journal) you could easily spend several hours fiddling with desktop systems and leave at your convenience without being seen by anyone. Cleaning staff also never questioned me when I stayed late (and conversely I never questioned them), you should train staff to approach people they do not recognize and ask them politely if they need assistance or are lost. While access to buildings cannot often be controlled effectively (to many entrances / different tenants / etc.) you can

control access to your floor, a locked door with a CCTV monitoring it after hours is often a good deterrent.

"Practical Unix and Internet Security" from O'Reilly covers physical problems as well and is definitely worth buying.

## Console security

With physical access to most machines you can simply reboot the system and ask it nicely to launch into single user mode, or tell it to use `/bin/sh` for `init`. You can enter the BIOS and tell the machine to boot from a floppy, doing a quick end run around most security. Alternatively you can simply enter the bios, disable the IDE controllers, put a password on the BIOS, rendering the machine largely unusable.

## BIOS / Open Firmware security

Assuming the attacker does not steal the entire machine the first thing that they will usually try is to reboot the system and either boot from a floppy disk (or other removable media). If they can do this then any standard file protection is useless, the attacker declares themselves to be root, mounts the filesystem and gains complete access to it.

To secure a x86 BIOS you typically enter it by hitting "delete" or a function key during the boot process, the actual name and location of the BIOS password varies significantly, look for "security" or "maintenance". There are usually different levels of password protections, on some motherboards you can disable the keyboard until a password is typed in (the BIOS intercepts and blocks input until it sees the password entered on the keyboard), on others it only prevents accessing the BIOS. Generally speaking you want to block access to the BIOS, and lock the boot sequence to the first internal storage device (i.e. the first IDE disk or SCSI).

Even if you do everything right there are still some ways for an attacker to subvert the boot process. Many older BIOS's have universal passwords, generally speaking this practice has declined with modern systems, but you may wish to inquire with the vendor. Another potential problem to be aware of is that many add-on IDE and SCSI controller cards have their own BIOS, from which you can check the status of attached devices, choose a boot device, and in some cases format attached media. Many high-end network cards also allow you to control the boot sequence, letting you boot from the network instead of a local disk. This is why physical security is critical for servers. Other techniques include disabling the floppy drive so that attackers or insiders cannot copy information to floppy and take it outside. You may also wish to disable the serial ports in users machines so that they cannot attach modems, most modern computers use PS/2 keyboard and mice, so there is very little reason for a serial port in any case (plus they eat up IRQ' s Same goes for the parallel port, allowing users to print in a fashion that bypasses your network, or giving them the chance to attach an external CD-ROM burner or harddrive can decrease security greatly. As you can see this is an extension of the policy of least privilege and can decrease risks considerably, as well as making network maintenance easier (less IRQ conflicts, etc.). There are of course programs to get the BIOS password from a computer, one is available <http://www.cgsecurity.org/>, it is available for DOS and Linux.

If you decide to secure the BIOS's on s systems you should audit them once in a while if possible, simply reboot the machine and try to boot off of a

floppy disk or get into the BIOS. If you can then you know someone has changed settings on the system, and while there may be a simple explanation (a careless technician for example) it may also be your first warning that an attack has occurred. There are several programs for Linux that allow an attacker with root access to gain the BIOS password, while this is a rather moot point it does bear mentioning (if an attacker has gained root access they can already do pretty much anything).

To secure a Sparc or UltraSparc boot prom send a break during boot-up, hit stop-a, and you should be presented with the ok> prompt. Setting your password is a simple matter of using the password command and typing the password in twice. You will then want to set the security-mode, using "setenv" from the default of none to command at the very least, and full if you are security conscious (warning, you will need the password to boot the machine).

```
ok
ok password
ok New password (only first 8 chars are used):*****
ok Retype new password: *****
ok
ok setenv security-mode full
ok
```

You can also set "security-mode" to "command" which will require the password to access the open firmware but is less strict than "full". Do not lose this password, especially if you set the security-mode to full, as you may need to replace the PROM to recover the system. You can wipe the password by setting "security-mode" to "none".

Unfortunately if you are using Apple hardware you cannot secure the boot process in any meaningful manner. While booting if the user holds down the command-option-P-R keys it will wipe any settings that exist, there is no way to avoid this. About the only security related option you can set is whether the machine automatically reboots or not, this is useful for servers to prevent a remote attacker from changing the kernel for example (which require a system reboot). Hold down the command-option-O-F keys to access the OpenFirmware and from there you need to:

```
go> setenv auto-boot? False
```

However because a local attacker can easily flush the settings there is no inherent security. If you need to use Apple systems as servers then it is highly advisable to lock them in a cabinet of some sort. As workstations in a public area your best solution is to automate the reloading of the OS to speed recover time.

## **LILO security**

LILO is the Linux boot loader, it handles all the nitty gritty tasks of getting the kernel into memory and bootstrapping them machine into something

that resembles a useful computing device. LILO handles many things, from telling the kernel to look for multiple network cards to how much memory there is (assuming Linux won't detect how much your system has). There are several options you can pass to LILO to bypass most any account security or console security.

Booting Linux into single user mode with (assuming the label is "linux"):

```
linux single
```

This will dump you into a root level command prompt on many systems. From there you can simply add a new users with UID 0, format the disks or copy Trojan'ed binaries off a floppy disk. One way many vendors deal with this is by using sulogin, single user login, which prompts for root's password before letting you on. This will prevent people from accessing single user mode and getting directly to a root prompt however there is a another attack. You can specify which program will be used to init the system, instead of the default init you can use a command shell:

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

Which will present you with a root prompt. The best way to defeat this and the single mode problem are to secure LILO and prevent the passing of boot arguments without a password. Many people argue that this somehow inhibits normal use of the system, however there should be no need normally for the user booting the system to pass LILO arguments (if an argument is needed at boot time it should be put into lilo.conf permanently). Generally speaking the only time you will need to pass LILO arguments is when something on the system breaks, at which point it's probably wise to send out a technician who knows the password to fix the system.

To secure LILO you must set a password, and use the restricted keyword. You can also set security on a per image basis, or on LILO as a whole. If you set security on a per image basis you will be able to tell LILO which image you wish to boot without needing a password, but you will not be able to pass the kernel arguments such as "single". If you set security on LILO as a whole then you will only be able to boot to the default image, specifying a different image will require the password.

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

password=thisisapassword
restricted

image=/boot/vmlinuz-2.2.18
    label=linux
    read-only
    root=/dev/hda1
```

```
image=/boot/vmlinuz-2.2.17
    label=linux-old
    read-only
    root=/dev/hda1
```

This can be cumbersome, however it does prevent attackers from booting a different image.

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

```
image=/boot/vmlinuz-2.2.18
    label=linux
    read-only
    root=/dev/hda1
    password=thisisapassword
    restricted
```

```
image=/boot/vmlinuz-2.2.17
    label=linux-old
    read-only
    root=/dev/hda1
    password=thisisapassword
    restricted
```

The above example will prevent attackers from messing with the "linux" and "linux-old" image, however if they need to boot the old kernel (because a driver is broken, or SMP support doesn't work quite right) then they will not need the password to do so.

Depending on the level of security you want you can either restrict and password protect all of LILO, or just the individual images. In any event if you are deploying workstations or servers you should use one to prevent users from breaking into systems in less than 10 seconds. While things like `sulogin` will prevent a user from getting a root prompt if they enter single user mode the attacker can always use `"init=/bin/sh"`.

One minor security measure you can take to secure the `lilo.conf` file is to set it immutable, using the `"chattr"` command. To set the file immutable simply run the following command as root:

```
chattr +i /sbin/lilo.conf
```

and this will prevent any changes (accidental or otherwise) to the `lilo.conf` file. If you wish to modify the `lilo.conf` file you will need to unset the immutable flag run the following command as root:

```
chattr -i /sbin/lilo.conf
```

only the root user has access to the immutable flag.

## Grub security

Grub is the default boot loader in Debian, Mandrake and possibly Red Hat 7.2. More on this to come.

## Rebooting the system

If possible you should make rebooting the system more difficult, by default almost all Linux distributions have ctrl-alt-del enabled so that it reboots the machines. However, unlike Windows, this is not necessary. In Linux you can easily control the behavior of ctrl-alt-del, simply by editing the /etc/inittab file:

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

You may wish to disable it entirely (simply comment it out with a #) or modify the command issued. Minimally you can create a bash script such as:

```
#!/bin/bash
#
# This script emails a message to root and then shuts
# down the system after a 5 second delay
#
/bin/date | /bin/mail -s "ctrl-alt-del used" root
/bin/sleep 5s
/sbin/shutdown -t3 -r now
```

Now every time someone hits ctrl-alt-del you will have an email message in root's account logging it. You can also send that email to another system (i.e. to root@example.org). If you do this you may wish to introduce a small delay between the mail command and the shutdown command to ensure the email gets out before the mail system is turned off. Of course an attacker can always hit the reset switch, power button, unplug the system, or trip the breakers for an entire floor of computers. If the system is in a locked cabinet however this is quite a bit more conspicuous than simply using a three finger salute (ctrl-alt-del) to reboot the system.

# Summary

An attacker with physical access to the console, or worse yet the hardware itself has many potential avenues of attack they can pursue. Worse yet if their goal is to simply deny use of service, or damage the software and hardware doing so is trivial. Flipping the power off and on during the boot process will often result in drive corruption, or they can simply pour a glass of water into the power supply. Locking servers up in secure rooms is critical, a competitor can easily afford to pay a janitor several thousand dollars to turn a server off, and pour pop into the power supply, resulting in a dead server when staff turns it back on. Workstations are typically more vulnerable as they are in accessible areas, but hopefully they are interchangeable with another workstation, with all data being stored on servers.

---

[Back](#)

Last updated on 5/1/2002

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Administrative tools

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

There are a variety of tools to make administration of systems easier, from local tools like sudo which grant limited superuser privileges to www based systems that allow for remote management from a cybercafe while on vacation. For information on how to login remotely (i.e. interactive shell prompts) please see the [shell server](#) section.

## Local tools

While it is possible to administer a Linux system from the command line using no "additional" tools it can be bothersome. If you wish to split up administrative tasks the "sub administrators" will often require root access to restart daemons, modify configuration files and so forth. Simply giving them all root access, or sharing the root password is often the first step to serious problem (this is one of the major reasons many large sites get broken into).

## YaST

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

## sudo

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

```
#Give the user 'seifried' full access
seifried ALL=(ALL) ALL
```

```
#Create a group of users, a group of hosts, and allow them to shutdown the server as root
Host_Alias WORKSTATIONS=localhost, station1, station2
User_Alias SHUTDOWNUSERS=bob, mary, jane
Cmnd_Alias REBOOT=halt, reboot, sync
Runas_Alias REBOOTUSER=admin
SHUTDOWNUSERS WORKSTATIONS=(REBOOTUSER) REBOOT
```

## Super

Super is one of the very few tools that can actually be used to give certain users (and groups) varied levels of access to system administration. In addition to this you can specify times and allow access to scripts, giving setuid access to even ordinary commands could have unexpected

consequences (any editor, any file manipulation tools like chown, chmod, even tools like lp could compromise parts of the system). Debian ships with super, and there are rpm's available in the contrib directory. This is a very powerful tool (it puts sudo to shame in some ways), but requires a significant amount of effort to implement properly (like any powerful tool), and I think it is worth the effort. Some example config files are usually in the /usr/doc/super-xxxx/ directory. Super is available <ftp://ftp.ucolick.org/pub/users/will/>.

## WWW based tools

WWW based administration tools provide an attractive solution since virtually every modern computer and Internet access point is web capable (sometimes that is all they are capable of).

### Webmin

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

### Linuxconf

Linuxconf is a general purpose Linux administration tool that is usable from the command line, from within X, or via it's built in www server. From within X it provides an overall view of everything that can be configured (PPP, users, disks, etc.). To use it via a www browser you must first run Linuxconf on the machine and add the host(s) or network(s) you want to allow to connect (Conf > Misc > Linuxconf network access), save changes and quit. Then when you connect to the machine (by default Linuxconf runs on port 98) you must enter a username and password. By default Linuxconf only accepts root as the account, and Linuxconf doesn' t support any encryption (it runs standalone on port 901), so I would have to recommend very strongly against using this feature across networks unless you have IPsec or some other form of IP level security. Linuxconf ships with several distributions and is available <http://www.solucorp.qc.ca/linuxconf/>. Linuxconf also doesn' t seem to ship with any man pages/etc, the help is contained internally which is slightly irritating.

## Other network based tools

On the other hand web based administration tools tend to be limited, and are typically not designed for hetrogenous installations (i.e. Linux, HP-UX, AIX and so forth). "Industrial" strength tools may be called for, like the following ones.

### Pikt

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

### VNC

Virtual Network Computer (VNC) is similar to X or PCAnywhere. You can display a graphical desktop, and control it remotely, with NT or Linux as the server and/or client. VNC across 10 megabit Ethernet is quite good, however it does tend to use a lot of computer power relative to other methods of remote administration. You can get VNC <http://www.uk.research.att.com/vnc/>. Security VNC isn' t so grāt, but there are several sites with information on securing VNC, using SSL, SSH and other methods. There is also a page on securing VNC with SSH port forwarding at: <http://www.zip.com.au/~cs/answers/vnc-thru-firewall-via-ssh.txt>.

### cfengine

cfengine is a set of tools for automating administration tasks and is network aware. You can get cfengine <http://www.cfengine.org/>.

---

[Back](#)

Last updated on 5/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

Backups

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

I don't know how many times I can tell people, but it never ceases to amaze me how often people are surprised by the fact that if they do not backup their data it will be gone, if the drive suffers a head crash on them or they hit 'delete' without thinking. Always backup your system, even if it's just the config files, you'll save yourself time and money in the long run. This is even on the SANS top 20 list.

To backup your data under Linux there are many solutions, all with various pros and cons. There are also several industrial strength backup programs, the better ones support network backups which are a definite plus in a large non-homogenous environment.

One of the other critical things to remember with backups is that whoever has access to them (backup admin, cleaning staff) will have access to all your files unless you encrypt the backups. Physically securing backups is critical, damaging backups physically so they cannot be recovered is extremely easy, with magnetic media simply place a strong magnet near them, for CD's simply scratching the surface or cracking the CD will prevent usage. You should also keep a relatively recent set of backups offsite in case the building burns down or is inaccessible for some other reason (such as a chemical spill).

## Non-commercial backup programs for Linux

There are numerous non commercial backup programs for Linux ranging from simple tools suitable for saving a few files to professional multi-system network backups.

### Tar and Gzip or Bzip2

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

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

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

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

to compress it as much as possible (granted harddrive space is cheaper then a politicians promise but compressing it makes it easier to move around). You might want to use bzip2, which is quite a bit better then gzip at compressing text, but it is quite a bit slower. I typically then make a copy of the archive on a remote server, either by ftping it or emailing it as an attachment if it' s not too **lg** (e.g. the backup of a typical firewall is around 100k or so of config files).

## **rsync**

rsync is an ideal way to move data between servers. It is very efficient for maintaining large directory trees in synch (not real time mind you), and is relatively easy to configure and secure. rsync does not encrypt the data however so you should use something like SSH or IPsec if the data is sensitive (SSH is easiest, simply use "-e ssh"). rsync is [covered here](#).

## **Amanda**

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

## **Commercial backup programs for Linux**

### **BRU**

BRU (Backup and Restore Utility), has been in the Linux world since as long as Linux Journal (they have had ads in there since the beginning as far as I can tell). This program affords a relatively complete set of tools in a nice unified format, with command line and a graphical front end (easy to automate in other words). It supports full, incremental and differential backups, as well as catalogs, and can write to a file or tape drive, basically a solid, simple, easy to use backup program. BRU is available at <http://www.tolisgroup.com/products3.html>.

## **Quickstart**

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

<http://www.tolisgroup.com/products3.html>.

## **Backup Professional**

[http://www.unitrends.com/br\\_bp.html](http://www.unitrends.com/br_bp.html)

## **CTAR**

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

## **CTAR:NET**

[http://www.unitrends.com/br\\_ct.html](http://www.unitrends.com/br_ct.html)

## **PC ParaChute**

[http://www.unitrends.com/ps\\_cr.html](http://www.unitrends.com/ps_cr.html)

## **Legato Networker**

Legato Networker is another enterprise class backup program, now completely supported on Linux as both client and server. You can get it from:

<http://www.legato.com/>.

## Backup media

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

Name of Media	Pro' s	Con' s
Hard Drive	It's fast. It' s cheap. It' s huge (160 gigs). It' s pretty reliable. (\$2-\$3 USD per gig)	It might not be big enough, and they do fail, usually at the worst possible time. Harder to take offsite as well. RAID is a viable option though.
CDROM	Not susceptible to EMP, and everyone in the developed world has a CDROM drive. Media is also pretty sturdy and cheap (\$0.20 USD per 650 Megs or so)	CDROM's do have a finite shelf life of 5-15 years, and not all recordables are equal. Keep away from sunlight, and make sure you have a CDROM drive that will read them.
Tape	It's reliable, you can buy BIG tapes, tape carousels and tape robots, but they' æ not very cheap.	Magnetic media, finite life span and some tapes can be easily damaged (you get what you pay for), also make sure the tapes can be read on other tape drives (in case the server burns down....).
Floppies	I'm not kidding, there are rumors some people still use these to backup data.	It' æ floppy. They go bad and are very small. Great for config files though.
Zip Disks	I have yet to damage one, nor have my cats. They hold 100 megs which is good enough for most single user machines.	Not everyone has a zip drive, and they are magnetic media. The IDE and SCSI models are passably fast, but the parallel port models are abysmally slow. Watch out for the click of death.
Jazz Drives	1 or 2 gig removable hard drives, my SCSI one averages 5 meg/sec writes.	They die. I'm on my third drive. The platters also have a habit of going south if used heavily. And they aren't cheap. They are junk.
LS120	120 Megs, and cheap, gaining in popularity (hah, I actually believed that). These things are dead as far as I can tell.	Slow. I' m no kidding. 120 megs over a floppy controller to something that is advertised as "up to 3-4 times faster then a floppy drive".
Printer	Very long shelf life. requires a standard Mark 1 human being as a reading device. Handy for showing consultants and as reference material. Cannot be easily altered.	You want to retype a 4000 entry password file? OCR is another option as well.

Last updated on 7/5/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

Filesystem security

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

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

So the attacker somehow manages to circumvent the physical security and console security you put in place and now has a root prompt. What can you do to protect sensitive files, and prevent the attacker from accessing or copying them? Keeping data on a server can help, even with local root access an attacker will still have to steal credentials to access the fileserver (assuming you use root\_squash under NFS or have similarly secured your file sharing configuration). Of course there is still sensitive local information, and the solution to this is disk encryption. There are several disk encryption packages for Linux, most will let you create an encrypted volume and access it. The usual method is to create an encrypted file and mount it using a loopback mount so it appears as a disk to the system. There is then usually a kernel interface, or userland program that allows you to access it. These systems are good and will protect sensitive information, however for an additional level of security you should encrypt everything, data partitions, system partitions, and swap partitions. An attacker can usually find interesting bits of data in the swap partition, and if they can access system files they can replace your favorite text editor with one that creates unencrypted backup copies of the file in /tmp for later retrieval.

## Basic file commands

The basic utilities to interact with files are: "ls", "chown", "chmod" and "find". Others include "ln" (for creating links), "stat" (tells you about a file) and many more. As for creating and maintaining the file systems themselves, we have "fdisk" (good old fdisk), "mkfs" (MaKe FileSystem, which formats partitions), and "fsck" (FileSystem ChecK, which will usually fix problems). So, what is it we are trying to prevent hostile people (usually users, and or network daemons fed bad info) from doing? A Linux system can be easily compromised if access to certain files is gained, for example the ability to read a non-shadowed password file results in the ability to run the encrypted passwords against crack, easily finding weak password. This is a common goal of attackers coming in over the network (poorly written CGI scripts seem to be a favorite). Alternatively, if an attacker can write to the password file, he or she can seriously disrupt the system, or (arguably worse) get whatever level of access they want. These conditions are commonly caused by "tmp races", where a setuid program (one running with root privileges) writes temporary files, typically in /tmp, however far to

many do not check for the existence of a file, thus allowing an attacker to make a hard link in /tmp pointing to the password file, and when the setuid program is run, kaboom, /etc/passwd is wiped out or possibly appended to. There are many more attacks similar to this, so how can we prevent them?

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

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

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

I thought I should also mention "find". It finds files (essentially it will list files), and can also filter based on permissions/ownership (also on size, date, and several other criterion). A couple of quick examples for hunting down setuid/guid programs:

to find all setuid programs:

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

to find all setgid programs:

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

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

```
chown user:group object
```

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

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

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

## Secure file deletion

One thing many of us forget is that when you delete a file, it is not actually gone. Even if you overwrite it, reformat the drive, or otherwise attempt to destroy it, chances are it can be recovered, and typically data recovery services only cost a few thousand dollars, so it might well be worth an attackers time and money to have it done. The trick is to scramble the data by repeatedly flipping the magnetic bits (a.k.a. the 1's and 0's) so that when finished no traces of the original data remain (i.e. magnetic bits still charged the same way they originally were). Two programs (both called wipe) have been written to do just this. An ideal solution for long term security is to encrypt data where possible, if keys are protected properly then an attacker will have a very difficult time recovering any data. This combined with good deletion techniques makes recovering data very difficult, of course there are still possibilities for data leakage (i.e. temporary files on unencrypted swap partitions).

### wipe

Wipe is available <http://wipe.sourceforge.net/>.

### fwipe

fwip is available <http://www.nb.net/~lbudney/linux/software/fwipe.html>.

## Access control lists (ACL's)

One major missing component in Linux is a filesystem with Access Control Lists (ACL's) instead of the standard User, Group, Other with it's dozen or so permissions. ACL's enable you to control access to the filesystem in a much more fine grained fashion, for example on a file you may want to grant the user "bob" full access, "mary" read, the groups sales "change", the accounting group "read", and nothing for everyone else . Under existing Linux permissions you could not do this. Hence the need for ACL's. The ACL stuff is currently under kernel patches.

## NSA SELinux

NSA SELinux is now based on the Linux Security Module (LSM) kernel patches and provides (among other things) access control lists.

## RSBAC

[Information is available here.](#)

## Critical system configuration files

Like any operating system Linux has a number of configuration files. Some of these are extremely critical, for example the password file. This section covers several of the more critical files, explains their layout, how they interact with the system, and so on. For example it is counter-intuitive that while the password file is probably the most critical file on the system, it must be readable by everyone, and there are several good reasons for this.

### **/etc/ directory**

Typically the /etc/ directory contains the majority of the system and application configuration files and many critical startup scripts. Some applications when installed to the /usr/local/ directory will place their files in /usr/local/etc/ (or other locations), this directory should largely be treated in the same way as /etc/. If an attacker is able to change files chances are they will be able to elevate their privileges and ultimately gain root access since many startup scripts are run with root privileges.

## /etc/passwd

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

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

Passwords are stored utilizing a one way hash (the default hash used is crypt, newer distributions support MD5 which is significantly stronger). Passwords cannot be recovered from the encrypted result, however you can attempt to find a password by using brute force to hash strings of text and compare them, once you find a match you know you have the password. This in itself is generally not a problem, the problem occurs when users choose easily guessed passwords. The most recent survey results showed that %25 of passwords could be broken in under an hour, and what is even worse is that %4 of users choose their own name as the password. Blank fields in the password field are left empty, so you would see “:”, this is something that is critical for the first four fields (name, password, uid and gid).

## /etc/shadow

The shadow file holds the username and password pairs, as well as account information such as expiry date, and any other special fields. This file should be protected at all costs and only the root user should have read permission to it.

## /etc/groups

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

```
groupname:encrypted_password:GID:member1,member2,member3
```

A group may contain no members (i.e. it is unused), a single member or multiple members, and the password is optional (and typically not used).

## /etc/gshadow

Similar to the password shadow file, this file contains the groups, password and members. Again, this file should be protected at all costs and only the root user should have read permission to it.

/etc/login.defs

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

/etc/shells

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

/etc/securetty

This file contains a list of tty's that root can log in from. Console tty's are usually /dev/tty1 through /dev/tty6. Serial ports (if you want to log in as root over a modem say) are /dev/ttyS0 and up typically. If you want to allow root to login via the network (a very bad idea, use sudo) then add /dev/typ1 and up (if 30 users login and root tries to login root will be coming from /dev/typ31). Generally you should only allow root to login from /dev/tty1, and it is advisable to disable the root account altogether, before doing this however please install sudo or program that allows root access to commands.

## File encryption

Do you have files on your computer that you wouldn't want your spouse to read, or perhaps your main competitor. Chances are if you use your computer for work or general usage the answer is yes. Also what happens if you want to send a file to someone, or let them download it from you, but you only have access to a public site (like a free web hosting company). The answer is to encrypt the file, and fire it off.

## PGP

Pretty Good Privacy is available as a command line driven program for most UNIX platforms, and there are a variety of front end GUI programs for it. I would not recommend using PGP on a UNIX platform since a completely OpenSource, and compatible replacement is now available, in the form of GnuPG.

## GnuPG

GnuPG is a GPL licensed (a.k.a. completely free in every respect), written in Germany (a very pro-crypto and pro-privacy country). Since it is available in full source code chances are it has been ported to your UNIX platform (and if not try compiling it, it might work). You can download [GnuPG](#) as a compressed tarball of source code, and there are links to a number of source and binary packages for various UNIX platforms. Once installed GnuPG behaves very similarly to PGP. The first thing you'll probably want to do is generate a new keypair, simply use the command "gpg --gen-key", it will create a ".gnupg" directory in which to store your keys, option files and so on and exit, you then run it again and it will lead you through the key creation process. Choosing the defaults during key generation is a pretty safe bet, although you may want to use a 2048 bit keysize (realistically if someone manages to crack 1024 bit keys, chances are they can get at your 2048 bit key, however if they are only trying to brute force it a longer key is a good way to reduce the chances of that). For personal keys the expiry is typically set to "0" (that is to say they do not expire), however if these keys are for corporate use, or for really sensitive information it is a good idea to expire keys and rotate them (every month, year, decade, whatever your security policy dictates). The most important thing when generating a key (in my opinion) is the passphrase. This is a string of characters which should consist of letters (upper and lower case) numbers and punctuation marks, the longer the better (I'd say the bare minimum is 10 characters). This controls access to the private key, which is used to sign items (and if compromised means an attacker could easily impersonate you), and to decrypt data (meaning an attacker could access all your data). Keep your private key secure! If an attacker gains access to this key they only have to brute force the passphrase, which is typically a lot weaker than a random 1024 bit (or longer) key. Worse yet they may steal your passphrase, with a keyboard sniffer or similar attack, resulting in a compromise of your key. If the attacker does not have access to your private key they will be forced to guess it, which takes a brutally long time (on average however, there is a chance they may guess the key correctly on their first try).

Signing files is useful if you want to distribute a file to someone, and be able to prove that you sent it, and it was not tampered with. Internally GnuPG takes a hash sum (such as MD5 or SHA1) of the file (basically it reduces the file to a shorter, unique string of data) which it then encrypts with your private key, generating a signature. This signature can then be decrypted with your public key, resulting in possession of the hash sum of the file, simply take the hash sum of the file in question, and if they match, then obviously the file is what it claims to be. This signature file can be a binary file, or converted into text (for example signing email, or distributing file signatures via email). To sign a file with gpg simply use :

```
$ gpg -b file
```

which will create a detached signature of the file.

To verify the signature use "gpg --verify file.sig file". If all is well you should see something like:

```
$ gpg --verify file.sig file
gpg: Signature made Sat 15 Jan 2000 05:23:31 AM MST using DSA key ID 47D0D9A8
gpg: Good signature from "Kurt Seifried <seifried@seifried.org>"
```

If someone has fiddled with the file or signature you will see something like:

```
$ gpg --verify file.sig file
gpg: Signature made Sat 15 Jan 2000 05:23:31 AM MST using DSA key ID 47D0D9A8
gpg: BAD signature from "Kurt Seifried <seifried@seifried.org>"
```

Encrypting files is also relatively simple, a person uses your public key to run the data through a one way algorithm which results in a seemingly random mishmash of data, you can then use your private key to recover what the original data was, thus decrypting it. To encrypt a file to someone you first need their public key, you can download it from their homepage (if they have it online of course), or you can go to a public key server, of which there are many:

<http://pgp.ai.mit.edu/> - PGP key server

<http://www.keyserver.net/> - OpenPGP key server

Once you have their key it is simply a matter of signing and encrypting the file (just encrypting the file is rare as there is no proof of who the data is from, unless you use some other method, like physically handing them a floppy disk with the encrypted file). The following is an example of me signing a file and encrypting it with my public key:

```
$ gpg -s -e file
```

```
You need a passphrase to unlock the secret key for
user: "Kurt Seifried <seifried@seifried.org>"
1024-bit DSA key, ID 47D0D9A8, created 2000-01-15
```

```
You did not specify a user ID. (you may use "-r")
```

```
Enter the user ID: seifried@seifried.org
```

The user ID can either be the key ID (such as: 47D0D9A8), the email address associated with the key (kurt@seifried.org) or the name (not recommended as these are not unique, there are many John Smith' s). You will end up with a "file.gpg" that is binary, if you wish to send the file via email it is advisable to use the "-a" ("--armor") option which will result in "file.asc" and is ASCII text, so you can read it straight into an email, or print it out, mail it, and let them OCR and decrypt it at their end. To decrypt a file sent to you simply:

```
$ gpg --decrypt file.asc
```

```
You need a passphrase to unlock the secret key for
user: "Kurt Seifried <seifried@seifried.org>"
1024-bit ELG-E key, ID 47D0D9A8, created 2000-01-15 (main key ID 39B0D9A8)
```

and it will display the file (hopefully a text file) to your screen, followed by the veracity of the signature (if you have the persons public key):

```
gpg: Signature made Sat 15 Jan 2000 06:06:19 AM MST using DSA key ID 47D0D9A8
gpg: Good signature from "Kurt Seifried <seifried@seifried.org>"
```

if you want to save the decrypted file simply use "--output filename" and it will dump the content to "filename". You can also use shell commands such as "|" or ">" to further mangle the output (this is useful if you have automated systems such as a reporting mechanism which sends encrypted emails to a central repository).

# Filesystem encryption

Filesystem encryption is an important security consideration if the data stored on the machine is of sufficient value or sensitivity. No matter how well you secure a system and tightly control file permissions if someone steals the machine or boots it into single user mode or from a floppy disk they can trivially bypass most protection methods, except for file encryption. Unfortunately most encrypted file systems will "leak" information, temporary files, swap partitions and so on can all end up with clear text copies of the files you want to protect with encryption. The best solution to deal with this problem is of course to encrypt everything, unfortunately this is very difficult and results in a lot of over head. Another solution (although less optimal) is to periodically wipe the clear space on your harddrive (i.e. the space where deleted files remain), as well as cleaning out temporary files and swap partitions, of course finding all of these is never simple.

## TCFS

TCFS is a kernel level data encryption utility, similar to CFS. It however has several advantages over CFS; as it is implemented at the kernel level it is significantly faster. It is tightly integrated with NFS meaning you can server data securely on a local machine, or across the network. It decrypts data on the client machine, so when used over a network the password/etc is never passed over the network. The only catch is that it has not yet been ported to the latest 2.2 kernels (2.2.16 and 2.2.17 only). You can get TCFS <http://tcfs.dia.unisa.it/>.

## BestCrypt

BestCrypt is a disk encryption program available for Windows and Linux. The nice thing is you can create an encrypted container (a file that is then mounted as a filesystem), and use it in Windows or in Linux (as long as it resides on a partition accessible to both, so putting it on your Windows partition is fine since Linux reads almost all Windows partition types). BestCrypt consists of some kernel modules (so your kernel will need to support loadable kernel modules obviously, and it helps if you are using tools like depmod, modprobe and the kernel module loader), and a userspace utility called "bctool". This program is however officially in "beta testing" for Linux, and probably should not be used for critical data (if it is, make sure you have backups). After testing BestCrypt for Linux I am satisfied that even though the software is officially beta, it is probably stable enough for most users, however your mileage may vary, all sales final, and don' blame me for any lost data. The only real problem with BestCrypt is a severe lack of documentation, while there is a man page that explains basic options, there is not a single example of how to create and mount a container (I suspect the release will have documentation, their Windows version documentation is quite good, a half meg helpfile). You need to download the software first, available as a source tarball, and source rpm (very easy to install on an RPM based system). Simply download either one, I would recommend the source rpm if you can. You can get it <http://www.jetico.com/>.

```
# rpm -Uvh BestCrypt-0.3b-1.src.rpm
BestCrypt #####
# cd /usr/src/redhat/SPECS
```

```
# rpm -ba bcrypt.spec
```

followed by a lot of text while it unpacks, compiles and assembles the source RPM and binary RPM. You should then have a:

```
/usr/src/redhat/RPMS/i386/BestCrypt-0.3b-1.i386.rpm  
/usr/src/redhat/SRPMS/BestCrypt-0.3b-1.src.rpm
```

Simply install the binary RPM with a:

```
# rpm -Uvh /usr/src/redhat/RPMS/i386/BestCrypt-0.3b-1.i386.rpm  
BestCrypt #####
```

If you do not have an RPM based system, or the source RPM doesn' work for you, compiling the source code directly from it's tarball should be possible. Simply download the file, unpack it to an appropriate place (such as /usr/local/src) and issue the commands:

```
# make  
# make install
```

And you should be up and running. The first step is to create a container (a file that is encrypted and mounted as a partition):

```
# bctool new -a blowfish -s 10M file  
Enter password:  
Verify password:
```

You can of course use the "gost" or "des" algorithms, I would not recommend them as gost is less tested then the "twofish" and "blowfish" algorithms that BestCrypt supports, and single des is to easy to brute force. The next step is to format the container, you' ll probably want to use MS-DOS if sharing with Windows (i.e. a dual boot Linux and Windows machine), or if just Linux then ext2 is a good bet. You can also specify the size, if you make it so small this can be a problem, but because it is a file and not a true partition you can easily create a new, larger file, move all the data to it and use it instead of the older smaller one.

```
# bctool format -t ext2 file  
Enter password:  
mke2fs 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09  
Filesystem label=  
OS type: Linux  
Block size=1024 (log=0)  
Fragment size=1024 (log=0)  
2560 inodes, 10238 blocks  
511 blocks (4.99%) reserved for the super user  
First data block=1  
2 block groups  
8192 blocks per group, 8192 fragments per group  
1280 inodes per group  
Superblock backups stored on blocks:  
8193
```

```
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Once the file is formatted you should be able to mount it:

```
# bctool mount file /root/encrypt/
Enter password:
# df
Filesystem      1k-blocks      Used Available Use% Mounted on
/dev/hda1        3122956        70596   2893720    2% /
/dev/hda2        2917360        24224   2744940    1% /crypto
/root/file       9909           13      9385      0% /root/encrypt
```

As you can see it is mounted as a part of the filesystem, just like a floppy disk would be for example. Remember to control access to the directory hosting the encrypted files carefully, no matter how good the encryption, if you have it set world readable you won't have gained any security. Also remember that as a user, root owns the / and can take ownership of any file or directory and see what's in it. Alternatively if an attacker gains root access they can log your keystrokes (or terminal traffic) and gain your password (and access to your files). As always your security is only as good as the weakest link.

## PPDD

PPDD is similar to BestCrypt, but instead of creating a file, encrypting that and mounting it, it actually uses a partition which is encrypted and mounted using the PPDD driver, because of this it can do a few additional things BestCrypt can't. If you only want to encrypt a few directories then I advise compiling PPDD as a kernel module, but if you want to encrypt the entire file system (including what you boot from) you will need to compile PPDD directly into the kernel (although as of 1.0 it's not to hard). Unless you have a GPL only policy I would recommend using BestCrypt if you are new to this (it is easier to install and use, and you can buy support). PPDD does have one enormous advantage over BestCrypt however, you can encrypt all of the system, including the boot drive and swap partition, making it ideal for situations such as laptops with sensitive data and minimizing the risk (to zero if need be) of accidentally leaving sensitive data in an unencrypted location (such as the swap file, /tmp, and so on) so if you need a higher security level I would recommend PPDD over BestCrypt (simply because you can encrypt everything). Another advantage of PPDD is that it uses two passwords instead of just one for each encrypted filesystem, so you can give one administrator one password, and another administrator the other password, meaning no single person can gain access to the data. Unfortunately as of the writing of this chapter PPDD is not available for kernel 2.2.13 or 2.2.14, so you will have to run the older 2.2.12 kernel (which is the stock kernel on many distributions in any case). You can get PPDD <http://linux01.gwdg.de/~alatham/>.

Download PPDD, and unpack it in a suitable location, such as /usr/local/src/, there are several files you should read, most notable the README file, and once done install I would recommend reading the PPDDHow.txt file. Installation is rather simply with:

```
#make check_linux
```

```
#make trial_patch
#make apply_patch
#make devices
```

This will first test the kernel source to make sure it's the right version and so on, then it will test the patches, then apply the patches proper, and then create the devices needed (similar to what BestCrypt does). At this point you need to recompile your kernel, first make sure you go into the configuration (via `make config` or `make menuconfig` or `make xconfig`), and enable the PPDD driver (in the Block devices section). Then save the config file and recompile the kernel as you normally would. Once that is done you will have to install the new kernel (copy it to `/boot` typically, edit `lilo.conf` and rerun `lilo`). Once you have rebooted you will want to build the tools for PPDD and install them with:

```
#make
#make install
```

At this point you should be ready to use it, however I would recommend running the tests with:

```
#make test
```

They take a while to run, but it will save frustration later on if something is broken. Using PPDD is relatively simple, there are a number of utilities for creating, managing, encrypting file systems, and so on. You will also want to set the permissions and ownership on the `/dev/xxxx` that contains your encrypted data so that only root has access to it, PPDD will complain otherwise

```
#chown root:root /dev/hda3
#chmod ugo-a /dev/hda3
#ppddinit /dev/ppdd0 /dev/hda3
#ppddsetup -s /dev/ppdd0 /dev/hda3
#mke2fs -b 1024 /dev/ppdd0
#mount /dev/ppdd0 /crypt
```

At this point you should have a directory called `/crypt` which is `/dev/hda3` (although on `df` and the like it will show up as `/dev/ppddx`). I will cover how to encrypt your entire filesystem with PPDD, at a later date however (it is extensively documented though).

## Hiding data

One issue many people forget that is the very act of encrypting data can draw attention. For example if a corporate administrator scanned workstations for files ending in `.pgp`, and you were the only one with files such as that.... Unfortunately most of these programs leave traces and are detectable.

## StegHide

StegHide hides data in files such as sound and picture files where not all of the bits in a byte are used. Since the data is encrypted it will appear random, and proving that the data is actually there is difficult. The only downside is to store a one megabyte file you need a sound/picture file of several megabytes, which can be cumbersome (but hard drives and high speed access are becoming cheap so it's a moot point). You can get StegHide <http://www.stego.com/>.

## StegFS

Steganographic File System actually hides data on your harddrive, making it difficult to prove that it even exists. This can be very useful as the attacker first has to find the data, let alone break the strong encryption used to protect it. You can get StegFS <http://ban.joh.cam.ac.uk/~adm36/StegFS/>.

## OutGuess

OutGuess hides data in image files, meaning you can send files in a way that won't attract too much attention (and can't really be proved either). You can get it <http://www.outguess.org/>.

## RubberHose

RubberHose is available <http://www.rubberhose.org/>.

---

[Back](#)

Last updated on 3/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

# Authentication

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Authentication is typically one of the two main lines of defense that systems and networks rely upon, so ensuring that your authentication subsystems are implemented correctly is important. The majority of Linux systems rely on usernames and passwords, while support for tokens, smartcards and other authentication systems are available they are still relatively rare. On top of this sits PAM, as far as I know all major vendors use PAM by default, so understanding how PAM works and using it correctly is very important.

## PAM

"Pluggable Authentication Modules for Linux is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users." Straight from the PAM documentation, I don't think I could have said it any better. But what does this actually mean? For example; take the program "login", when a user connects to a tty (via a serial port or over the network) a program answers the call (getty for serial lines, telnet or SSH for network connections) and starts up a login program, login then typically requests a username, followed by a password, which it checks against the /etc/passwd file. This is all fine and dandy until you have a spiffy new digital card authentication system and want to use it. Well you will have to recompile login (and any other apps that will do authentication via the new method) so they support the new system. As you can imagine this is quite laborious and prone to errors.

PAM introduces a layer of middleware between the application and the actual authentication mechanism. Once a program is PAM'fied, any authentication methods PAM supports will be usable by the program. In addition to this PAM can handle account, and session data which is something normal authentication mechanisms don't do very well. For example using PAM you can easily disallow login access by normal users between 6pm and 6am, and when they do login you can have them authenticate via a retinal scanner. By default Red Hat systems are PAM aware, and newer versions of Debian are as well (see below for a table of PAM'fied systems). Thus on a system with PAM support all I have to do to implement shadow passwords is convert the password and group files; and possibly add one or two lines to some PAM config files (if they weren't already added). Essentially, PAM gives you a great deal of flexibility when handling user authentication, and will support other features in the future such as digital signatures with the only requirement being a PAM module or two to handle it. This kind of flexibility will be required if Linux is to be an enterprise-class operating system. Distributions that do not ship as "PAM-aware" can be made so but it requires a lot of effort (you must recompile all your programs with PAM support, install PAM, etc), it is probably easier to switch straight to a PAM'fied distribution if this will be a requirement. PAM usually comes with complete documentation, and if you are looking for a good overview you go <http://www.sun.com/software/solaris/pam/>.

Other benefits of a PAM aware system is that you can now make use of an NT domain to do your user authentication, meaning you can tie Linux

workstations into an existing Microsoft based network without having to say buy NIS / NIS+ for NT and go through the hassle of installing that. As far as I know all modern Linux distributions have PAM support and default to it.

## **PAM Cryptocard Module**

A PAM cryptocard module is available <http://projects.jdimedia.nl/index.phtml?ID=crypto&L=&BROW=1&W=1260&H=886>. Cryptocards are excellent for securing interactive logins since they do not require any special equipment on the client end, thus you can log in from a cybercafe for example with no fear of your password being stolen (since it changes each time you log in). Unfortunately Cryptocards tend to be expensive and require some user training, I would advise them primarily for installations with a higher need of security then "normal" or for infrastructure related servers and equipment (i.e. Authentication servers).

## **Pam Smart Card Module**

Smartcards can be used to sign and encrypt email as well as providing login services. The primary problem with smartcards however is that the client station needs a compatible card reader, the chances of finding these on a system outside of your office are slim indeed. A module to provide PAM support for smartcards is available <http://www.linuxnet.com/apps.html>.

## **Pam module for SMB**

SMB (Server Message Block) is incredibly popular protocol for the simple reason Microsoft has chosen to use it as their primary protocol for Windows 9x and NT (it is also supported in 2000). Many sites have existing NT infrastructures, adding Linux servers that require their own authentications infrastructure can be quite troublesome. Fortunately you can authenticate on Linux machines against SMB servers, packages are available [http://rpmfind.net/linux/rpm2html/search.php?query=pam\\_smb](http://rpmfind.net/linux/rpm2html/search.php?query=pam_smb) and the primary site is [http://www.csn.ul.ie/~airlied/pam\\_smb/](http://www.csn.ul.ie/~airlied/pam_smb/). You can also install SAMBA on the machine and use this to authenticate but for workstations the PAM module is much more appropriate.

## **Authentication services**

Authentication services such as NIS and Kerberos are covered in the network servers section of the LASG [here](#). Generally speaking they are easy to implement client side on modern Linux distributions, during install you are often given the choice of Kerberos, LDAP or NIS+ passwords and their related settings. Setting up the servers however is another matter.

# Passwords

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

## Use a better hash

Using a hash such as MD5 or blowfish significantly increases the amount of computing power needed to execute a brute force attack, but there are two large problems with switching from the traditional crypt hash. The first is compatibility, if you use NIS or NIS+ with systems such as Solaris using a different hash then crypt will break authentication, obviously a problem. The other problem is that no matter how strong a hash you use poor passwords (such as the username or "dog") will still be easily discovered. If possible you should use a better hash, but if this is not possible then there is another solution.

## Use shadow passwords

User account data is stored in `/etc/passwd` traditionally, but the actual password hashes and related data (password expiry, etc.) is stored in `/etc/shadow`, a file only readable by root. Programs that need to check a password can either run as root or use a `setuid` or `setgid` wrapper program (like PAM provides) to check the password, the only way to get access to `/etc/shadow` requires root privileges. There have been problems in past with `setuid` programs that read `/etc/shadow` leaking information, however these are relatively rare (and you are no worse off then storing passwords in a world readable location).

Several OS's take the first solution, Linux has implemented the second for quite a while now. Because most vendors rely on PAM for authentication services, implementing a new authentication scheme is relatively simple, all you need to do it add a PAM module that understands the new authentication scheme and edit the PAM config file for whichever program (say `login`) uses it. Now for an attacker to look at the hashed passwords they must go to quite a bit more effort then simply copying the `/etc/passwd` file.

## Cracking passwords

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

Most modern Linux distributions use MD5 hashed passwords at a minimum (notable exceptions are SuSE and Debian which default to crypt for backwards compatibility with NIS and the like). In any event password crackers will usually catch poor passwords or dictionary based passwords quickly. As well on modern systems passwords are protected in shadow password files, if an attacker has access to this file chances are they have sufficient privilege to do other things to compromise the system.

## VCU

VCU (Velocity Cracking Utilities) is a windows based programs to aid in cracking passwords, “VCU attempts to make the cracking of passwords a simple task for computer users of any experience level.” You can download it [http://packetstormsecurity.org/groups/wiltered\\_fire/NEW/vcu/](http://packetstormsecurity.org/groups/wiltered_fire/NEW/vcu/)

## Password storage

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

Many of these programs have been found to contain flaws, I advise using them with caution.

## Strip

Strip is a palm pilot program for storing passwords securely and can also be used to generate passwords. It is GNU licensed and available <http://www.zetetic.net/products.html>. The generation function is flawed and should not be used.

---

### [Back](#)

Last updated on 4/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Log files and other forms of monitoring

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

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

## General log security

Generally speaking you do not want to allow users to see the log files of a server, and you especially don't want them to be able to modify or delete them. Generally speaking most log files are owned by the root user and group, and have no permissions assigned for other, so in most cases the only user able to modify the logs will be the root user (and if someone cracks the root account all bets are off). There are a few extra security precautions you can take however, the simplest being to use the "chattr" (CHange ATTTRibutes command) to set the log files to append only. This way in the event of a problem like a /tmp race that allows people to overwrite files on the system they cannot significantly damage the log files. To set a file to append only use:

```
chattr +a filename
```

only the superuser has access to this function of chattr. If you set all your log files to append only you must remember that log rotation programs will fail as they will not be able to zero the log file. Add a line to the script to unset the append only attribute:

```
chattr -a filename
```

and add a line after the log rotation script to reset the append only flag. If you keep log files on the system you may also wish to set them immutable so they cannot be tampered with as easily, to set the file immutable simply:

```
chattr +i filename
```

and this will prevent any changes (due to /tmp races, etc.) to the file unless the attacker has root access (in which case you're already in a world of hurt).

```
chattr -i filename
```

only the root user has access to the immutable flag.

## System logging

One feature of Linux (and most unices) is the syslog and klog facilities which allow software to generate log messages that are then passed to alog daemon and handled (written to a local file, a remote server, given to a program, and so on).

### sysklogd / klogd

In a nutshell klogd handles kernel messages, depending on your setup this can range from almost none to a great deal if for example you turn on

process accounting. It then passes most messages to syslogd for actual handling (that is it places the data in a physical file). The man pages for sysklogd, klogd and syslog.conf are pretty good with clear examples. One exceedingly powerful and often overlooked ability of syslog is to log messages to a remote host running syslog. Since you can define multiple locations for syslog messages (i.e. send all kern messages to the /var/log/messages file, and to console, and to a remote host or multiple remote hosts) this allows you to centralize logging to a single host and easily check log files for security violations and other strangeness. There are several problems with syslogd and klogd however, the primary ones being the ease of which once an attacker has gained root access to deleting/modifying log files, there is no authentication built into the standard logging facilities.

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

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

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

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

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

And you can selectively log to a separate log host:

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

Which would result in all kernel messages being logged to /var/log/kernel-log, this is useful on headless servers since by default kernel messages go to /dev/console (i.e. someone logged in at the machines). In the second case all emergency messages would be logged to the host "syslog-host", and all the mail log files would be sent to the "mail-log-host" server, allowing you to easily maintain centralized log files of various services. The default syslog that ships with most Linux distributions is horribly insecure, log files are easily tampered with (or outright destroyed), and logging across the network is completely insecure as well as dangerous for the servers involved. I do not advise using syslog if you actually have a need for reliable logging (i.e. the ability to later view log files in the event of a break-in).

The default file permissions on the log files are usually read / write for root, and nothing for anyone else. In addition to this you can (and should) set

the files append only (remember to take logrotate into account though, it needs to zero the files). This will prevent any deletion / modifications to the log files unless root unsets the append only attribute first.

## **modular syslog**

changed

[http://www.core-sdi.com/download/download1\\_modular.html](http://www.core-sdi.com/download/download1_modular.html)

The major problem with syslog however is that tampering with log files is trivial (setting the log files append only with “chattr +a” helps, but if an attacker gains root, they can unset the attribute). There is however a secure version of syslogd, available at [http://www.core-sdi.com/download/download1\\_modular.html](http://www.core-sdi.com/download/download1_modular.html) (these guys generally make good tools and have a good reputation, in any case it is open source software for those of you who are truly paranoid). This allows you to cryptographically sign logs to ensure they haven’t been tampered with. Ultimately, however, an attacker can still delete the log files so it is a good idea to send them to another host, especially in the case of a firewall to prevent the hard drive being filled up.

## **next generation syslog**

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

## **Nsyslogd**

Nsyslogd supports tcp, and SSL for logging to remote systems. It runs on a variety of UNIX platforms and you can download it from: <http://coombs.anu.edu.au/~avalon/nsyslog.html>.

## **Log monitoring**

Log files are not much good unless you actually check them once in a while, this is an almost impossible task for most of us however due to the sheer volume of log files. There are a variety of tools to automate these tasks however.

## **Psionic Logcheck**

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

<http://www.psionic.com/abacus/logcheck/>.

## **colorlogs**

colorlogs will color code log files allowing you to easily spot suspicious activity. Based on a config file it looks for keywords and colors the lines (red, cyan, etc.), it takes input from STDIN so you can use it to review log files quickly (by using "c at", "tail" or other utilities to feed the log file through the program). You can get it at: <http://www.resentment.org/projects/colorlogs/>.

## **WOTS**

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

## **swatch**

swatch is very similar to WOTS, and the log files configuration is very similar. You can download swatch from:

<ftp://ftp.stanford.edu/general/security-tools/swatch/>.

## **Kernel logging**

The lowest level of logging possible is at the kernel level. Generally speaking users cannot disabled or avoid this type of logging, and also are usually not even aware it exists (a definite advantage).

## **auditd**

[Information is available here.](#)

## **Shell logging**

A variety of command shells have built in logging capabilities.

### **bash**

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

#### **HISTFILE**

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

#### **HISTFILESIZE**

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

#### **HISTSIZE**

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

The variables are typically set in `/etc/profile`, which configures bash globally for all users, however, the values can be over-ridden by users with the `~username/.bash_profile` file, and/or by manually using the `export` command to set variables such as `export EDITOR=emacs`. This is one of the reasons that user directories should not be world readable; the `.bash_history` file can contain a lot of valuable information to a hostile party. You can also set the file itself non world readable, set your `.bash_profile` not to log, set the file non writeable (thus denying bash the ability to write and log to it) or link it to `/dev/null` (this is almost always a sure sign of suspicious user activity, or a paranoid user). For the root account I would highly recommend setting the `HISTFILESIZE` and `HISTSIZE` to a low value such as 10. On the other hand if you want to log users shell history and otherwise tighten up security I would recommend setting the configuration files in the user's home directory to immutable using the `chattr` command, and set the log files (such as `.bash_history`) to append only. Doing this however opens up some legal issues, so make sure your users are aware they are being logged and have agreed to it, otherwise you could get into trouble. Don' t forget to set `/home/username/.bash_history` append only (`chattr +A`).

---

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Attack detection

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

No matter how good your security is you it can be compromised. That is why it is so important to have various forms of attack detection present, so that when an incident happens you are alerted to it as soon as possible (and not when you start getting complaints from other sites).

## Baselines

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

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

There are also a variety of tools that do not quite fit under the headings here, but are aimed at attack detection. One is the Linux Intrusion Detection System (LIDS) project, [more information is listed here](#).

## File system monitoring

So you' ve secured your machines, and done all the things that needed to be done. So how do you make sure it' s actually doing what it is supposed to do, or prove to someone that it is as secure as you say it is? Well you conduct an audit. This can be as simple as reviewing the installed software, configuration files and other settings, or as complex as putting together or hiring a tiger team (or ethical hackers, or whatever buzzword(s) you prefer) to actively try and penetrate your security. If they can' t then you did your job well (or they suck), and if they do get in, you know what needs to be fixed (this is also a good method to show the CIO that security is not a one shot affair, it is a constant battle). One thing almost all attackers do is modify system files, once you detect a break in, how do you know which files are ok and which are not? Short of a complete reinstall the only way to be sure (and even then it' s not always 100%) is to use software to create signatures of files that cannot be forged so you can compare them later on.

## Tripwire

Tripwire is no longer a open source tool. I have absolutely NO problems with commercial software. However, when you expect me to rely on a program to provide security, when I (nor anyone else really) can not view the source (it is available under some special license agreement, probably an NDA) I must decline. Tripwire costs approximately \$70 for Linux, and is only available as an RPM package aimed at Red Hat Linux (tripwire is \$500 for other operating systems). I feel this is rather on the high side for a piece of software that can easily be replaced with alternatives such as L5 or Gog&Magog. Tripwire is available <http://www.tripwiresecurity.com/>. There is also the "classic" version available <http://www.tripwire.org/>.

## AIDE

AIDE is a tripwire replacement that attempts to be better then tripwire. It is GPL licensed which makes it somewhat more desirable then tripwire from a trust point of view. It supports several hashing algorithms, and you can download <http://www.cs.tut.fi/~rammer/aide.html>.

## ViperDB

ViperDB checks setuid/setgid programs and folders and can notify you (via syslog) of any changes or reset their permissions and ownership to what they should be. ViperDB creates a series of databases (flat text files actually) in the directory root, i.e.: /etc/.ViperDB might contain:

```
/etc/login.defs,1180,-,root,rw-,root,r--,r--,Apr,15,18:03
/etc/minicom.users,1048,-,root,rw-,root,r--,r--,Mar,21,19:11
/etc/CORBA,1024,d,root,rwx,root,r-x,r-x,Jun,14,16:51
```

```
/etc/X11,1024,d,root,rwx,root,r-x,r-x,Jun,14,23:05  
/etc/cron.d,1024,d,root,rwx,root,r-x,r-x,Apr,14,17:09
```

Unfortunately ViperDB doesn't seem to handle sub directories, so you will have to add them to the viperdb.ini file with something like:

```
find /etc/ -type d >> /usr/local/etc/viperdb.ini
```

viperdb.pl has 3 options, -init (creates a set of databases), -check (checks files against databases, sends any messages to syslog, and then recreates the databases) and -checkstrict (checks files against databases, resets permissions if necessary, sends any messages to syslog, and then recreates the databases). What this means is if you use -check, you will get a warning that say /etc/passwd is now world writeable, and since it recreates the databases the next time you run viperdb you will NOT get a warning. I would advise running viperdb is -checkstrict mode only, and make sure you run viperdb with the -init option after manipulating any file / folder permissions in protected directories. ViperDB is available <http://www.resentment.org/projects/viperdb/>.

## Pikt

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

## Backups

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

## Network monitoring / attack detection

If the last section has you worried you should be. There are however many defenses, active and passive against those types of attacks. The best ways to combat network scans are keep software up to date, only run what is needed, and heavily restrict the rest through the use of firewalls and other mechanisms.

Luckily in Linux these tools are free and easily available, again I will only cover open source tools, since the idea of a proprietary firewall/etc is rather worrying. The first line of defense should be a robust firewall, followed by packet filters on all Internet accessible machines, liberal use of TCP-WRAPPERS, logging and more importantly automated software to examine the logs for you (it is unfeasible for an administrator to read log files nowadays).

## **DTK**

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

## **Psionic TriSentry - PortSentry, HostSentry and LogSentry**

Psionic TriSentry consists of three components, PortSentry, HostSentry and LogSentry. PortSentry detects and logs port scans, including stealthy scans (basically anything nmap can do it should be able to detect). Psionic PortSentry can be configured to block the offending machine (in my opinion a bad idea as it could be used for a denial of service attack on legitimate hosts), making completion of a port scan difficult. As this tool is in beta I would recommend against using it, however with some age it should mature into a solid and useful tool. HostSentry spots local anomolous behaviour in user accounts, and reports situations that fall outside of normal parameters (i.e. Bob from accounting logging in at 2 AM on a Sunday). The last component is LogSentry, essentially a log file monitoring applications which will alert you if it spots strange problems. TriSentry is available <http://www.psionic.com/products/>.

## **scanlogd**

scanlogd monitors network packets and if a threshold is exceeded it logs the packets. You can get it at: <http://www.openwall.com/scanlogd/>.

## **Firewalls**

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

## TCP-WRAPPERS

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

## Intrusion Detection Papers

FAQ: Network Intrusion Detection Systems, an excellent FAQ that covers all the major (and many minor) issues with IDS systems. Available <http://www.robertgraham.com/pubs/network-intrusion-detection.html>.

## Dealing with attacks

So you' ve done your homework, you installed tripwire, DTK, and so on. Now what do you do when your pager starts going off at 3am and tells you that someone just made changes on the primary NIS server? Dealing with an attack depends on several factors, is the attack in progress? Did you discover your company plan being sent out by the mail server to a Hotmail address? Did you get called in to find a cluster of dead servers? What are your priorities? Restoring service? Ensuring confidential data is safe? Prosecuting the attacker(s)? Several things to keep in mind:

- Response from the admin will depend heavily on the environment they are in. The attacker may have compromised the administrative accounts, so sending email may not work.
- Most sites usually don' t want to report attacks (successful or not) due to the potential embarrassment and related public relations problems.
- Most quick attacks, denial of service attacks and the like are spoofed. Tracking down the real attacker is very difficult and resource intensive.
- Even if all goes well there is a chance law enforcement will seize your equipment as evidence, and hold it, not something to be taken lightly.
- Do you know how the attacker got in (i.e. NFR recorded it), if so you might just want to plug the holes and go on.
- Try not to ignore attacks, but at the same time there are many people running garbage attacks in an effort to waste administrators time and energy (and possibly distract them from more subtle attacks).

Also before you deal with an attack, you should consult your company policy. If you don't have one consult your manager, the legal department, etc. It' s also a good idea to have a game plan to deal with attacks (i.e., the mail server is first priority, checking file servers is number two, who do you notify, etc) this will prevent a lot of problems when it happens (be prepared). The O' Reilly book ["Practical Unix and Internet Security"](#) covers this topic in great detail so I' m not going to rehash it. Go buy the book. There is also "Incident response" from O' Reilly, you should probably read it, it' s quite good.

An excellent whitepaper on this is also available, see [Appendix D](#), "How to Handle and Identify Network Probes".

## Packet sniffers

Packet sniffing is the practice of capturing network data not destined for your machine, typically for the purpose of viewing confidential/sensitive traffic such as telnet sessions or people reading their email. Unfortunately there is no real reliable way to detect a packet sniffer since it is mostly a passive activity, however by utilizing network switches and fiber optic backbones (which are very difficult to tap) you can minimize the threat. There is also a tool called AntiSniff, that probes network devices and sees if their response indicates an interface in promiscuous mode. These tools are also invaluable if your network is under attack and you want to see what is going on. There is an excellent FAQ on sniffing <http://www.robertgraham.com/pubs/sniffing-faq.html>.

## Snort

Snort is the king of packet sniffers now. It can also be used to detect various attacks. It can watch for activity such as Queso TCP-IP fingerprinting scans, Nmap scans, and the like. Snort is available <http://www.snort.org/>.

## tcpdump

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

## Ethereal

A nice looking network protocol analyzer (a.k.a., a souped up sniffer) with an interface very similar to NT's network monitor. It allows easy viewing of data payloads for most network protocols (tftp, http, Netbios, etc). It is based on GTK, thus meaning you will probably have to be running gnome to use it. I haven' tested it yet (but intend to). It is available <http://www.ethereal.com/>. Ethereal can also be used to open trace files from other software.

## **SPY**

SPY is an advanced multi protocol sniffer that runs on various platforms. You can get it <http://www.gromeck.de/Spy/>.

## **Other sniffers**

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

Numerous other packet sniffers are available <http://freshmeat.net/search/?site=Freshmeat&q=packet+sniff&section=projects>.

## **Packet sniffer detection**

In theory most operating systems leave tell tale signs when packet sniffing (that is to say their network interfaces respond in certain, non standard ways to network traffic). If the attacker is not to savvy, or is using a compromised machine then chances are you can detect them. On the other hand if they are using a specially built cable, or induction ring there is no chance of detecting them unless you trace every physical piece of network cable and check what is plugged into it.

## **AntiSniff**

As mentioned before AntiSniff is a tool that probes network devices to try and see if they are running in promiscuous mode, as opposed to normal modes of operation. It is supposedly effective, and will work against most sniffers. You can get it <http://www.securitysoftwaretech.com/antisniff/>. AntiSniff has not been maintained in over a year.

---

[Back](#)

Last updated on 7/5/2002

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

# Intrusion testing - scanning / intrusion tools

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Over the last few years the number of security tools for Windows and UNIX has risen dramatically, even more surprising is the fact that most of them are freely available on the Internet. I will only cover the free tools since they tend to be the best (i.e. nmap, Nessus, etc.) and any serious cracker will have these tools at their disposal, why shouldn't you? There are several main categories of tools; ones that scan hosts from within that host, ones that scan other hosts and report back variously what OS they are running, services that are available and so on (i.e. TCP level scanners), and at the top of the food chain are the intrusion tools (i.e. application level scanners) that can actually execute exploits, and report back whether they worked or not, and lastly I include the exploits category, while not strictly an intrusion tool per se they do exist and you should be aware of them. These tools and techniques can also be used to conduct a self audit and ensure that the systems react as you think they should (i.e. you should be able to run the denial of service attacks that Nessus is capable of with no ill effects on your servers).

## Host scanners

Host scanners are software packages you run locally on the system to probe for problems. Most of them are "dead" now because vendors have become much more security conscious and started to move away from some of the insanely insecure defaults that used to be common. As well with the increase of networking (i.e. virtually every system is now attached to the Internet full-time or at least part-time) the focus has shifted from host security (i.e. people with accounts) to network security (which means anyone of 100+ million people can potentially access it). For information on finding and removing setuid bits and tightening file permissions please see the [filesystem section](#).

## Network scanners

Network scanners typically operate at the network level (imagine that), using protocols like TCP-IP, UDP, ICMP to elicit a response that will (among other things) tell them if a server is listening on the port, if it is firewalled, what the OS in use is and so forth. If you can find open ports and services then chances are an attacker can too. The popularity of these network scanners is apparent if you run a firewall, often within an hour (or much less) you will be scanned, often aggressively. These tools are also quite useful for finding out how a network is secured, firewall and other restriction software such as tcp\_wrappers tend to respond differently.

## **Nmap**

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

## **Firewalk**

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

## **ICMP related scanning**

There is an excellent paper entitled "ICMP Usage In Scanning" by Ofir Arkin available <http://www.sys-security.com/html/papers.html>. It covers topics from detecting ACL' s using ICMP to specific hardware and operating system issues.

## **spidermap**

spidermap is a set of perl scripts to help automate scans and make them more selective. You can get it <http://www.digitaloffense.net/spidermap/>.

## **Application level Scanners**

Application level scanners are one evolutionary step up from network scanners (although they often incorporate network scanning). These software packages will actually identify vulnerabilities, and in some cases allow you to actively try and exploit them. If your machines are susceptible to these attacks, you need to start fixing things, as any attacker can get these programs and use them.

## **Nessus**

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

## **Saint**

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

## **Ftpcheck / Relaycheck**

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

## **SARA**

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

## BASS

BASS is the ‘Bulk Auditing Security Scanner’ allows you to scan the internet for a variety of well known exploits. It was basically a proof of concept that the Internet is not secure. You can get it from: <http://www.securityfocus.com/data/tools/network/bass-1.0.7.tar.gz>

## Exploits

I won't cover exploits specifically, since there are hundreds if not thousands of them floating around for Linux. Probably the best site to visit for exploits is Packetstorm (which is about the only major public exploit archive available now), available <http://www.packetstormsecurity.net/>.

---

### Back

Last updated on 7/5/2002

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Firewalling

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Firewalling is the practice of filtering network traffic, typically at the point where your network connects to another (e.g. the Internet, a customers LAN, etc.) network, that may be untrusted (in the case of the Internet) or perhaps even trusted (another floor of your building). Like firewalls in a large building, a network firewall can prevent and even block the spread of an attack if one segment is compromised successfully, like their namesake firewalls can stop your network from being further compromised. There is a good FAQ on Internet firewalls at: <http://www.interhack.net/pubs/fwfaq/>. A list of port numbers and what they are is available [here at seifried.org](http://www.seifried.org).

## Firewall software for Linux

Linux has gone through a series of firewalls. The most recent is IPTables (sometimes referred to as NETFILTER), preceding that was ipchains, and

preceding this was ipfwadm, as well there are a number of other firewalls such as IPF and SINUS firewall (some of which only work on 2.0 and 2.2 series kernels so are largely obsolete). I would recommend using a 2.4 kernel with IPTables if possible since it offers stateful packet inspection, which makes firewalling protocols such as DNS and FTP properly a lot easier. Linux also supports IPMASQ (IP Masquerading) as part of its firewall capabilities, an advanced form of NAT (Network Address Translation). IPMASQ allows you to hook up a network of computers to the Internet but proxy their connections at the IP level. Thus all traffic appears to be coming and going to one machine (the Linux IPMASQ box) which affords a high degree of protection to the internal network. As an added bonus the clients on the internal network require NO proxy configuration; as long as the Linux IPMASQ server is configured correctly, and the clients use it as their default gateway, things will work quite well.

Both ipchains and ipfwadm provide the following basic capabilities:

- blocking/passing of data based on source/destination IP address and port
- masquerading of connections based on source/destination IP address and port

In addition to which ipchains supports:

- port forwarding
- creation of chains, for more intricate rules and conditions, speedier rule parsing and easier management
- quality of service (QOS) routing, useful on low speed connections or otherwise saturated connections
- specification of IP/port/interface as well as inverse specification (using the !, i.e. "everything but")

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

When deciding policy you should ideally choose a policy that will default to denying everything unless specifically allowed through (that is if there is a failure it will hopefully be minimized via default policies) or a policy that allows everything and blocks certain services/hosts. I typically use a policy of default denial as it can accommodate mistakes and changes more safely than a policy that defaults to allowing data through.

Case in point, you have a server secured via firewalling, currently running Apache, you install WU-FTPD on it for internal use (so people can upload files) at 3 am, you forget to change the firewall rules. If you have chosen a policy of default allowal, anyone on the Internet can access the ftp server, and silly you, you installed an old version which allowed someone to compromise the machine. If on the other hand you go with a policy of default denial, they would not have access to the ftp server, and neither would your users, but you would find out quite quickly. Annoyed users are much easier to appease than fixing a network that has been compromised.

## Firewall concepts

Some of the problems with firewalling and what you can do ensure they don't affect you too much.

## [IPTables](#)

Firewalling for Linux 2.4 and 2.5. Provides stateful filtering unlike IPChains and IPFWADM. Combined with Linux advanced routing you can do many interesting and several downright bizarre things.

## [IPChains](#)

Firewalling for Linux 2.2 and present in 2.4 for backwards compatibility as well. Does not provide stateful filtering, but provides chains, making it easier to manage than ipfwadm.

## [IPFWADM](#)

Firewalling for Linux 2.0, no longer present. Not recommended.

## **Firewall piercing**

Sometimes you will be stuck behind a firewall that is not properly set up, or otherwise stopping you from accessing data you need to. Other times users will simply want to climb over any walls you put on the network, and the best way to defend is to understand the attack. There is actually a mini-HOWTO on this; <http://www.linuxdoc.org/HOWTO/mini/Firewall-Piercing.html>. In addition to this is an add-on for the IP Masquerading code that allows certain types of VPN' \$through, you can get it at: [ftp://ftp.rubyriver.com/pub/jhardin/masquerade/ip\\_masq\\_vpn.html](ftp://ftp.rubyriver.com/pub/jhardin/masquerade/ip_masq_vpn.html).

---

[Back](#)

Last updated on 27/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

# Firewalling with IPTables

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

IPTables (sometimes referred to as NETFILTER) is the current generation of packet firewalling for Linux. It is stateful, and easily extended, providing very powerful firewall capabilities to Linux. IPTables includes numerous new features, from advanced logging to the ability to do string matching. Backwards compatibility for ipfwadm and ipchains rules is provided with modules, allowing you to easily upgrade to IPTables with minimal effort. Unfortunately the majority of IPTables documents are not incredibly great, and the more advanced features can be difficult to use correctly.

Although the HOWTO's are far from perfect I can't really do justice to recreating the IPTables HOWTO's so instead I will just point you to them:

<http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc.html>.

This is the most basic document, and will get you up and running.

<http://netfilter.samba.org/documentation/HOWTO//NAT-HOWTO.html>

This document cover NAT (Network Address Translation, sometimes referred to as IPMASQ in Linux).

<http://netfilter.samba.org/documentation/HOWTO//netfilter-extensions-HOWTO.html>

This document covers extensions available in IPTables.

<http://netfilter.samba.org/documentation/index.html>

This lists all the documents available in various languages.

## A very basic example

For those of you that just want to get on with it here is a simple iptables firewall script I use that is suitable for machines with one interface:

```
#
# First set some default policies
#

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

```

#
# Then block the reserved network 10.* on the external interface eth0
#

-A INPUT -s 10.0.0.0/255.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j DROP

#
# Then we allow SSH, SMTP and DNS
#
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -p tcp -m tcp --dport 22:22 -j ACCEPT
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -p tcp -m tcp --dport 25:25 -j ACCEPT
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -p udp -m udp --dport 53:53 -j ACCEPT
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -p tcp -m tcp --dport 53:53 -j ACCEPT
#
# Now we block all incoming traffic to ports between 1 and 1024. For your system
#
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -p tcp -m tcp --dport 1:1024 -j REJECT
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -p udp -m udp --dport 1:1024 -j REJECT

```

## Some more tricks with IPTables

### Invalid packets

One of the nicest things about IPTables is that it is stateful, and there are several options for state: NEW, ESTABLISHED, RELATED and INVALID. INVALID is especially interesting as it will:

"A packet which could not be identified for some reason: this includes running out of memory and ICMP errors which don't correspond to any known connection. Generally these packets should be dropped."

Putting this rule first in your list may be a wise decision since it will prevent mangled packets from traversing your chains and additionally it may help with survivability of your server if someone attacks it.

```
-A INPUT -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -m state --state INVALID -j DROP
```

## Listing Tables

Typically most of us will use:

```
iptables -L -v -n
```

However this will not show all the tables (most importantly it misses the pretrouting and postrouting tables, used for NAT among other things).

To list current NAT tables:

```
iptables -t nat -L -v -n
```

## Dropping every second or third packet

This is guaranteed to drive network administrators a bit nutty, and amuse attackers to no end. Using the "nth" rule (not supported by default in most kernels) you can drop every "nth" packet:

```
iptables -A INPUT -p tcp --dport 80 -m nth --every 2 -j DROP
```

This rule would drop every second packet coming in to port 80 (the webserver usually).

---

[Back](#)

Last updated on 18/3/2002

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Network security

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Network security is a pretty broad topic, so I've broken it down into a couple of sections. In this area I cover the bottom 4 or so layers (transport,

network, datalink, physical) of the 7 layer OSI protocol stack, the top 3 (application, presentation, session) are in the network server section and so forth (roughly speaking). I also cover some of the basic network configuration files, since an explanation is always useful.

## Physical protocols

Physical protocols such as PPP and Ethernet provide the foundation for TCP-IP.

## **TCP-IP security**

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

Hostnames don't always point at the right IP addresses and IP addresses don't always reverse lookup to the right hostname. Do not use hostname-based authentication if possible. Because DNS cache poisoning is relatively easy, relying on an IP address for authentication reduces the problem to one of spoofing, which is somewhat more secure but by no means truly secure. There are no mechanisms in wide spread use to verify who sent data and who is receiving it except by use of session or IP level encryption (IPSec/IPv6 and other VPN technologies are starting to gain momentum however).

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

If you must communicate securely with people, consider using VPN technology. The only available technology that has wide acceptance and is slated to become a the standard (in IPv6) is IPSec, it is an open standard supported by many vendors and most major vendors have actual working implementations native to their OS (although some are crippled to comply with US export laws). Please see Appendix B or the Encrypting Services and Data section for more details.

## **IPSec**

[IPSec is covered in it's own section.](#) I think it is the future of VPN technology (it's the most commonly supported standard as of today, and an integral part of IPv6).

## **IPv6**

IPv6 provides no security per se, but it does have built in hooks for future security enhancements, IPSec support and so on. If used on a network it would of course make life more difficult for an attacker as IPv6 use is not yet widespread. If you want to learn more visit:

<http://www.bieringer.de/linux/IPv6/>. Linux currently supports IPv6 pretty much completely.

## **Attacking TCP-IP**

TCP-IP was not built with security in mind and consequently suffers a wide variety of problems that can be exploited by attackers. From hijacked connections to spoofing and sending mangled packets, the limits are endless.

## **HUNT Project**

The HUNT Project is a set of tools for manipulating TCP-IP connections (typically on an Ethernet LAN), that is it can reset connections, spy on them and do otherwise “naughty” things. It also includes a variety of ARP based attacks and other mischievous sources of fun, You can get HUNT at:

<http://lin.fsid.cvut.cz/~kra/index.html>

## **Basic config files and utilities**

## **/etc/inetd.conf**

inetd.conf is responsible for starting services, typically ones that do not need to run continuously, or are session based (such as telnet, or ftpd). This is because the overhead of running a service constantly (like telnet) would be higher than the occasional start up cost (or firing in.telnetd up) when a user wants to use it. For some services (like DNS) that service many quick connections, the overhead of starting the service every few seconds would be higher than constantly running it. Also with services such as DNS and email time is critical, a few seconds delay starting an ftp session won't hurt much. The man page for inetd.conf covers the basics ("man inetd.conf"). The service itself is called inetd and is run at boot time, so you can easily stop/start/reload it by manipulating the inetd process. Whenever you make changes to inetd.conf you must restart inetd to make the changes effective, `killall -1 inetd` will restart it properly. Lines in inetd.conf can be commented out with a # as usual (this is a very simple and effective way of disabling services like rexec). It is advisable to disable as many services in inetd.conf as possible, typically the only ones in use will be ftp, pop and imap. Telnet and r services should be replaced with SSH and services like systat/netstat and finger give away far too much information. Access to programs started by inetd can be easily controlled by the use of TCP\_WRAPPERS.

## **/etc/services**

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

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

for example:

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

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

For a large services file please see <http://seifried.org/security/ports/>

## **TCP\_WRAPPERS**

Using TCP\_WRAPPERS makes securing your servers against outside intrusion is a lot simpler and painless than you would expect. TCP\_WRAPPERS is controlled from two files:

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

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

```
in hosts.deny:
ALL: 0.0.0.0/0.0.0.0
```

which means all services, all locations, so any service not explicitly allowed is then blocked (remember the default is to allow). You might also want to just default deny access to say telnet and leave ftp wide open to the world. To do this you would have:

in hosts.allow:

```
in.telnetd: 10.0.0.0/255.255.255.0 # allow access from my internal network of 10.0.0.*
in.ftpd: 0.0.0.0/0.0.0.0 # allow access from anywhere in the world
```

in hosts.deny:

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

or if you wish to be really safe:

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

This may affect services such as ssh and nfs, so be careful!

You may wish to simply list all the services you are using separately:

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

If you leave a service on that you shouldn't have in inetd.conf, and DO NOT have a default deny policy, you could be up the creek. It is safer (and a bit more work, but in the long run less work than rebuilding the server) to have default deny rules for firewalling and TCP\_WRAPPERS, thus is you leave something on by accident, by default there will be no access to it. If you install something that users need access and you forget to put allow rules in, they will quickly complain that they can't get access and you will be able to rectify the problem quickly. Erring on the side of caution and accidentally denying something is a lot safer than leaving it open.

The man pages for TCP\_WRAPPERS are very good and available by:

```
man hosts.allow
```

```
man hosts_allow
```

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

```
man hosts.deny
man hosts_deny
```

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

- 1) strip off all \ (line continuations), making all the lines complete (also note the max length of a line is about 2k, better to use multiple lines in some cases).
- 2) strip out lines starting with #'s, i.e. all commented out lines. Thus:

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

this means the "in.telnetd: 1.1.1.1" line would be ignored as well.

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

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

### ps

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

```
USER PID %CPU %MEM SIZE RSS TTY STAT START TIME COMMAND
bin 320 0.0 0.6 760 380 ? S Feb 12 0:00 portmap
daemon 377 0.0 0.6 784 404 ? S Feb 12 0:00 /usr/sbin/atd
named 2865 0.0 2.1 2120 1368 ? S 01:14 0:01 /usr/sbin/named -u named -g named -t /home/named
nobody 346 0.0 18.6 12728 11796 ? S Feb 12 3:12 squid
nobody 379 0.0 0.8 1012 544 ? S Feb 12 0:00 (dnsserver)
```

```

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

```

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

## netstat

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

```

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

```

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

## **lsof**

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

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

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

## **Encryption of network traffic (non VPN)**

### **SSL**

There are a number of sources for information on SSL. Generally where SSL is applicable it is in the individual resource (i.e. WWW). For a good FAQ go here: <http://www2.psy.uq.edu.au/~ftp/Crypto/>. OpenSSL is an OpenSource implementation of the SSL libraries that is available from: <http://www.openssl.org/>.

## **Routing security**

There are a variety of routing software packages available for Linux. Most of them support the newer routing protocols which have a much higher degree of security than the older protocols such as RIP.

## routed

routed is one of the standard routing packages available for Linux. It supports RIP (about the oldest routing protocol still in service), and that's it. RIP is very simple, routers simply broadcast their routing tables to neighboring routers, resulting (in theory) in a complete routing table that contains entries for every destination on the Internet. This method is fundamentally insecure, and very inefficient outside of small secure networks (in which case it probably is not needed). Securing it is really not possible, you can firewall ports 520 and 521 which RIP uses to transfer data, however this can result in routes you want not getting through, and attackers can still spoof routes. Running this service is a very bad idea.

## gated

gated is a more advanced piece of routing software than routed. It supports RIP versions 1 and 2, DCN HELLO, OSPF version 2, EGP version 2, and BGP versions 2 through 4. Currently the most popular routing protocol seems to be BGP (Border Gateway Protocol), with OSPF gaining popularity (OSPF has built in security, is very efficient, and quite a bit more complicated).

## MRT

MRT (Multi-threaded Routing Toolkit) is a routing daemon and test toolkit that can handle IPv4 and IPv6. You can get it at: <http://www.mrtd.net/>.

## zebra

zebra is much more featured than gated, and sports a nice Cisco style command line interface. It runs as a daemon, and is multi threaded for performance, each protocol (RIP, OSPF, etc.) has it's own configuration, and you can run multiple protocols simultaneously (although this could lead to confusion/problems). There is a master configuration port, and a port for each protocol:

zebrasrv	2600/tcp	# zebra service
zebra	2601/tcp	# zebra vty
ripd	2602/tcp	# RIPd vty
ripngd	2603/tcp	# RIPngd vty
ospfd	2604/tcp	# OSPFd vty
bgpd	2605/tcp	# BGPd vty
ospf6d	2606/tcp	# OSPF6d vty

I would advise firewalling these ports. Access is controlled by a login password, and access to command functions requires another password (using the same syntax as Cisco, "enable"). You can download zebra from: <http://www.zebra.org/>.

---

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

Network Servers

## Network Based Authentication

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

### Overview

There are a variety of methods to share authentication data between systems. This is an increasingly common task as networks become more distributed, access points more numerous, and users scatter away from the traditional corporate LAN.

### NIS / NIS+

NIS and NIS+ (formally known as “yellow pages”) stands for Network Information Service. Essentially NIS and NIS+ provide a means to distribute password files, group files, and other configuration files across many machines, providing account and password synchronization (among other services). NIS+ is essentially NIS with several enhancements (mostly security related), otherwise they are very similar.

To use NIS you set up a master NIS server that will contain the records and allow them to be changed (add users, etc), this server can distribute the records to slave NIS machines that contain a read only copy of the records (but they can be promoted to master and set read/write if something bad happens). Clients of the NIS network essentially request portions of the information and copy it directly into their configuration files (such as /etc/passwd), thus making them accessible locally. Using NIS you can provide several thousand workstations and servers with identical sets of usernames, user information, passwords and the like, significantly reducing administration nightmares.

However this is part of the problem: in sharing this information you make it accessible to attackers. NIS+ attempts to resolve this issue however, but NIS+ is an utter nightmare to set up. NIS+ uses secure RPC, which can make use of single DES encryption (which I personally feel is too weak to even bother with). I would not recommend relying on the single DES encryption of secure RPC to secure your NIS data.

An alternative strategy would be to use some sort of VPN (like FreeS/WAN, which seems to solve a lot of problems) and encrypt the data before it gets onto the network. There is an NIS / NIS+ HOWTO at: <http://www.linuxdoc.org/HOWTO/NIS-HOWTO/> , and O'Reilly has an excellent book on the subject, "[Managing NFS and NIS](#)". NIS / NIS+ runs over RPC which uses port 111, both tcp and udp. This should definitely be blocked at your network border, but will not totally protect NIS / NIS+. Because NIS and NIS+ are RPC based services they tend to use higher port numbers (i.e. above 1024) in a somewhat random fashion, making firewalling of it rather difficult. The best solution is to place your NIS server(s) on an internal network that is blocked completely from talking to the Internet, inbound and outbound. There is also an excellent paper on securing NIS available from: <http://www.eng.auburn.edu/users/doug/nis.html>.

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

or

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

## Kerberos

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

## Radius

Radius is a commonly used protocol to authenticate dial-in users, and other types of network access. Many radius servers are available, check freshmeat.

### xtradius

<http://www.xtradius.com/>

## **gnu-radius**

<ftp://ftp.gnu.org/gnu/radius/>

## **perlradius**

<http://members.iinet.net.au/~michael/radius.html>

## **Cistron RADIUS server**

<http://www.radius.cistron.nl/>

## **FreeRADIUS**

<http://www.freeradius.org/>

## **ICRADIUS**

<http://radius.innercite.com/>

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Certificate authority software for Linux**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## Overview

Certificates are becoming increasingly popular, and if you have a high enough demand running your own certificate server can save a lot of time and money. There are other projects where it would be nice to have a certificate authority, but using a commercial one may be too expensive, in these cases the OpenSource alternatives can be quite good.

## Certificate authority software

### OpenCA

A project to provide a comprehensive set of tools for an "out of the box" certificate authority capable of handling X.509 certificates. It is available from: <http://www.openca.org/>.

### pyCA

pyCA is a collection of software scripts written in python to setup a certificate authority. You can download it from: <http://sites.inka.de/ms/python/pyca/>.

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Network services - DHCP

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## Overview

DHCPD is something all network admins should use. It allows you to serve information to clients regarding their network settings/etc, typically

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

I also highly recommend you run DHCPD version 2.x (3.x is in testing), it's got a lot of new features, and is easier to setup and work with. The absolute latest version(s) of this tend to be a bit neurotic however, be warned it is beta software. Definitely firewall DHCPD off from the Internet. DHCP traffic should only be on local segments, possibly forwarded to a DHCP server on another segment, but the only DHCP traffic you would see coming over the Internet would be an attack/DOS (they might reserve all your IP' ,sthus leaving your real clients high and dry). If you are forwarding DHCP traffic over the Internet, DON' TThis is a really bad idea for a variety of reasons (primarily performance / reliability, but security as well).

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

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

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

or

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

# DHCP servers

## ISC DHCPD

### Chroot'ing DHCPD

DHCPD consists of 2 main executables:

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

DHCPD requires 2 libraries:

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

A config file:

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

And a few other misc. files:

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

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

```
# Install bind so we have the appropriate files
#
rpm -i dhcpd-2.0b1p10-1.i386.rpm
#
# Create the directory structure
#
cd /chroot/dhcpd/ # or wherever
mkdir ./etc
mkdir ./usr/sbin
mkdir ./usr
mkdir ./var/dhcpd
mkdir ./var
mkdir ./lib
#
# Start populating the files
#
```

```
cp /usr/sbin/dhcpd ./usr/sbin/dhcpd
cp /etc/dhcpd.conf ./etc/dhcpd.conf
cp /etc/rc.d/init.d/dhcpd ./etc/dhcpd.init
cp /etc/rc.d/init.d/functions ./etc/functions
#
# Now to get the latest libraries, change as appropriate
#
cp /lib/ld-linux.ld-linux.so.2 ./lib/
cp /lib/libc.so.6 ./lib/
#
# And create the necessary symbolic links so that they behave
# Remember that dhcpd thinks /chroot/dhcpd/ is /, so use relative links
```

Then modify or create your startup script.

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

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

And a ps -xau should show something like:

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

## Moreton Bay DHCP Server

<http://www.moretonbay.com/dhcpd/>

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

# Network services - DNS

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## Overview

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

## DNS servers

### Bind

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

-u

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

-g

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

-t

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

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

Another aspect of bind is the information it contains about your network(s). When a person queries a DNS server for information, they typically send

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

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

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

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

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

```

};

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

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

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

```

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

```

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

```

or

```

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

```

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

```

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

```

```
ipfwadm -I -a deny -P udp -S 0.0.0.0/0 -D 0.0.0.0/0 53
```

or

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

## Chroot'ing DNS

<http://www.etherboy.com/dns/chrootdns.html>

## Dents

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

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Email servers

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

## Overview

Simple Mail Transfer Protocol (SMTP) is one of the more important services provided by the Internet. Almost all companies now have or rely upon email, and by extensions SMTP servers. There are many SMTP server packages available, the oldest and most tested is Sendmail (now commercially supported, etc.), and there are two new contenders, Postfix and Qmail, both of which were written from scratch with security in mind. Firewalling SMTP is straightforward, it runs on port 25, TCP:

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 25
```

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

```
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 25
```

```
ipchains -A input -p tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 25
```

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

```
ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 25
```

```
iptables -A INPUT -p tcp -m tcp -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 --dport 25
```

```
iptables -A INPUT -p tcp -m tcp -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 --dport 25
```

```
iptables -A INPUT -p tcp -m tcp -j REJECT -s 10.0.0.0/8 -d 0.0.0.0/0 --dport 25
```

## Sendmail

Sendmail has earned itself a very bad reputation for security, however I find it hard to blame software when I find systems running ancient versions of Sendmail. The root of the problem (if you'll pardon a bad pun) is that almost everyone runs Sendmail as root (and something like %70 of Internet email is handled by Sendmail machines, so there are a lot of them). As soon as an exploitable bug is found, finding a system to exploit it on is not all that hard. The last few releases of Sendmail have been quite good, no root hacks, etc, and with the new anti-spam features Sendmail is finally come of age. More information on Sendmail and source code is available from: <http://www.sendmail.org/>.

Chroot'ing Sendmail is a good option, but a lot of work, and since it largely runs as root, rather debatable as to the effectiveness of this (since root can break out of a chroot'ed jail). Newer releases of Sendmail from some vendors run less processes as root, and make it easier to chroot the daemon, however the last hole in Sendmail on Linux was locally exploitable and no amount of chroot or non root would have helped.

Keeping Sendmail up to date is relatively simple, I would recommend minimally version 8.12.x. More information on 8.12.x is available at: <http://www.sendmail.org/8.12.1.html>. Most distributions ship 8.11.x or 8.12.x, and some vendors also include SMTP auth support, although configuring it can be difficult (it is poorly documented in most cases). You can also get the source from but compiling Sendmail is not for the faint of heart or those that do not have a chunk of time to devote to it (although building a more up to date version via dpkg or rpm is possible without too much black magic). The latest version of sendmail is available from: <ftp://ftp.sendmail.org/>.

Sendmail only needs to be accessible from the outside world if you are actually using it to receive mail from other machines and deliver the mail locally. Indeed this is the default with Red Hat, Sendmail only listens on localhost, allowing you to receive mail sent locally, and send mail to other systems (however changing this default can be rather annoying).

If you only want to run Sendmail so that local mail delivery works (i.e. a stand alone workstation, test server or other) and so mail can easily be sent to other machines, simply firewall off Sendmail, or better, do not run it in daemon mode (where it listens for connections). Sendmail can be run in a queue flushing mode, where it simply 'wakes' up once in a while and processes local mail, either delivering it locally, or sending it off on its way across the 'net. To set Sendmail to run in queue mode:

edit your Sendmail startup script

and change the line that has:

```
sendmail -bd -qlh
```

to:

```
sendmail -qlh
```

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

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

`/etc/sendmail.cf`

Primary config file, also tells where other config files are.

`/etc/mail/`

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

`access`

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

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

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

`aliases`

`aliases` file, allows you to control delivery of mail local to the system, useful for backing up incoming user's email to a separate spool. Most list serve software uses this file to get mail sent to lists delivered to the programs that actually process them. Remember to run the command “`hewaliases`” after editing this file, and to then restart sendmail.

`domaintable`

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

`majordomo`

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

`sendmail.cw`

file containing names of hosts for which we receive email, useful if you host more than one domain.

sendmail.hf

location of help file (telnet to port 25 and type in "HELP")

virtusertable

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

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

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

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

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

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

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

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

## Dynamic Relay Authorization Control

Dynamic Relay Authorization Control (DRAC) ties into your POP/IMAP server to temporarily grant SMTP relay access to hosts that successfully authenticate and pick up mail (the assumption being these hosts will send mail, and not abuse this privilege. You can get it from:

<http://mail.cc.umanitoba.ca/drac/index.html>.

# Postfix

Postfix is a mail transfer agent (MTA) aimed at security, speed, ease of configuration, generally things Sendmail fails miserably at. I would highly recommend replacing Sendmail with Postfix. The only portion of Postfix that runs as root is a master control program, aptly called “master”, it calls several other programs to process mail to the queue (“pickup”), a program to manage the queue, wait for incoming connections, deferred mail delivers and so on (“qmgr”), a program to actually send and receive the mail (“smtpd”) and so on. Each part of Postfix is very well thought out, and usually does one or two tasks, very well. For example instead of the sendmail model where queued mail simply gets dumped into /var/spool/mqueue, in Postfix there is a world accessible directory called “maildrop” which is checked by “pickup”, which feeds the data to “cleanup” which moves the mail (if it’s properly formatted and so on) to a secure queue directory for actual processing.

The primary configuration files are held in /etc/postfix, and there are several primary configuration files you must have:

master.cf

Controls the behavior of the various “helper” programs, are they chroot’ed, maximum number of processes they may run and so forth. It’s probably best to leave the defaults on most mail servers unless you need to do some tuning for high loads or securing the server (i.e. chroot’ing it).

main.cf

This file is as close to sendmail.cf as you will get (for purpose, as for layout it’s quite different). It is well commented and sets all the major variables, and the locations and format of various files containing information such as virtual user mappings and related information.

Here is a list of variables and file locations you will typically have to set, the /etc/postfix/main.cf file is usually heavily commented. Please note the following examples of main.cf entries are not a complete main.cf.

```
# what is the machines hostname?  
myhostname = mail.example.org
```

```
# what is the domain name?  
mydomain = example.org
```

```
# what do I label mail as “from”?  
myorigin = $mydomain
```

```
# which interfaces do I run on? All of them usually.  
inet_interfaces = all
```

```
# a file containing a list of host names and fully qualified domains names I  
# receive mail for, usually they are listed like:  
# mydestination = localhost, $myhostname, etc  
# but I much prefer to keep them listed in a file.
```

```
mydestination = /etc/postfix/mydestination

# map of incoming usernames. "man 5 virtual"
virtual_maps = hash:/etc/postfix/virtual

# alias mappings (like /etc/aliases in sendmail), "man 5 aliases"
alias_maps = hash:/etc/postfix/aliases

# alias database, you might have different settings. "man 5 aliases"
alias_database = hash:/etc/postfix/aliases

# where to deliver email, Mailbox format or Maildir (traditional /var/spool/mail).
home_mailbox = Maildir/

# where to keep mail, usually /var/spool/mail/ but you can easily change it
mail_spool_directory = /var/spool/mail

# what command do we use to deliver email? /usr/bin/procmail is the default but if
# you want to use scanmail which is the AMaViS anti-virus tie in softwaresimply put:
mailbox_command = /usr/sbin/scanmails

# who do I relay email for, again you can list them, or keep them in a file (one
# per line).
relay_domains = /etc/postfix/relaydomains

# list of local networks (by default we relay mail for these hosts).
mynetworks = 10.0.0.0/24, 127.0.0.0/8

# what do we display to people connecting to port 25? By default it displays the
# version number which I do not.
smtpd_banner = $myhostname ESMTP $mail_name
```

Generally speaking any files that simply list one item per line (like `/etc/postfix/mydestination` or `/etc/postfix/relaydomains`) are usually just stored as a flat text file. Files that contain mappings (i.e. aliases, where you have entries like `'root: someuser'`) should be turned into hashed database files for speed (you can specify the type of file as `hash`, `dbm`, etc.).

Like most IBM products, Postfix has a very funky license, but appears to be mostly open source and free. Postfix is available at: <http://www.postfix.org/>. You can get binary postfix packages from a number of sources including most Linux vendors.

## Sendmail Pro

Sendmail Pro is a commercial version of Sendmail with support, and is available at: <http://www.sendmail.com/>. I haven't been able to get a demo or find anyone using it so I'm not 100% sure as to how close it is to the "original" Sendmail, although the company has told me it uses the same code base.

## QMAIL

Qmail (like postfix) was created as a direct response to perceived flaws in Sendmail. Qmail is not under an OpenSource approved license, and there is no binary distribution clause meaning you must install it from source code. OpenBSD has removed Qmail from ports due to numerous conflicts with the license, and I do not recommend using Qmail as vendor support is non-existent. Very little code in Qmail runs as root, and it is very modular compared to Sendmail (which is a pretty monolithic piece of code). You can download it from: <http://www.qmail.org/>. An excellent book is available for free at: <http://www.lifewithqmail.org/>.

## Zmailer

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

## DMail

DMail is a commercial mail server, and is not open source. You can download a trial version from: [http://netwinside.com/dmail\\_first.htm](http://netwinside.com/dmail_first.htm).

## nullmailer

nullmailer sends mail to smart hosts (relays) so that the local machine doesn't have to run any mail server software. It's at <http://em.ca/~bruceg/nullmailer/>.

## POP servers

POP (post Office Protocol) is a relatively simple protocol that allows you to retrieve email from a server and delete it. The basic commands are USER, PASS (used to login), LIST (to list emails and sizes), RETR (to retrieve and email) and DELE (to delete an email).

## UW IMAPD (contains the default popd for most distros)

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

<http://www.washington.edu/imap/>.

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

or

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

## Cyrus

Cyrus is an imap (it also supports pop and kpop) server aimed at ‘closed’ environments. That is to say that the users will not have any access to the mail server other than by imap or pop protocols. This allows Cyrus to store the mail in a much more secure manner and allows for easier management of larger installations. Cyrus is not GNU licensed but is relatively “free”, and available from: <http://asg.web.cmu.edu/cyrus/imapd/>.

## IDS POP

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

## Qpopper

Qpopper is freeware produced by Qualcomm (the makers of Eudora). I would not recommend it (the source code is available at: <ftp://ftp.qualcomm.com/eudora/servers/unix/popper/>). You can get it from: <http://eudora.qualcomm.com/freeware/qpop.html>.

## IMAP

IMAP is a much more advanced mail protocol, allowing you to retrieve email from the server, and manage it on the server (you can create folders to store messages on the server). This protocol is much more useful than POP since multiple email boxes are a bit more graceful, multiple people using one email box is workable, and for travelling users, since you download the headers first (subject, etc) and can more selectively retrieve email.

## UW IMAPD (contains the default imapd for most distros)

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

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

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

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

or

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

## Cyrus

Cyrus is an imap (it also supports pop and kpop) server aimed at 'closed' environments. That is to say that the users will not have any access to the

mail server other than by imap or pop protocols. This allows Cyrus to store the mail in a much more secure manner and allows for easier management of larger installations, and I highly recommend it. Cyrus is not GNU licensed but is relatively “free”, and available from:

<http://asg.web.cmu.edu/cyrus/imapd/>. There is also a set of add on tools for Cyrus available from: <ftp://ftp.hr.vc-graz.ac.at/cyrus-tools/>.

## **Courier-IMAP**

Courier-IMAP is a lightweight IMAP server specifically for use with Maildir style mailboxes (not /var/spool/mail). You can get it from:

<http://www.inter7.com/courierimap/>.

## **Scanning email for viruses**

While Linux is not terribly susceptible to viruses, Windows clients are.

### **Protector**

<http://protector.sourceforge.net/> blocks all attachments except those listed (i.e.a white list).

## **AMaViS**

AMaViS uses third party scanning software (such as McAfee) to scan incoming email for viruses. You can get AMaViS at: <http://www.amavis.org/>. Make sure you get the latest version, previous ones have a root compromise.

### **Sendmail**

Using AMaViS with Sendmail is relatively simple, it has a program called “scanmail” that acts as a replacement for procmail (typically the program that handles local delivery of email). When an email comes in instead of using procmail to deliver it, Sendmail calls scanmail which decompresses and decodes any attachments/etc. and then uses a virus scanner (of your choice) to scan the attachments. If no virus is found mail delivery goes ahead as usual. If a virus is found however, an email is sent to the sender informing them that they have sent a virus, and an email is sent to the intended recipient informing them about the person that sent them a virus. The instructions for this are at: <http://www.amavis.org/>

### **Postfix**

Since Postfix can make use of procmail to do local mail delivery it should work in theory without any trouble. In practice it takes a few minor tweaks to work correctly. To enable it replace the line in main.cf:

```
mailbox_command = /usr/bin/procmail
```

with the line:

```
mailbox_command = /usr/sbin/scanmail
```

and restart postfix. For the local warning to work (a warning is sent to the intended recipient of the message) the hostname of the machine (sundog, mailserver01, etc.) must be listed in the “mydestination” in main.cf, otherwise the warning does not get delivered. You should (and most sites generally do) redirect root’s email to a user account using the aliases file, otherwise warnings will not be delivered to root properly. By default as well mail to “virusalert” is directed to root, you should also redirect this mail to a normal user account.

## Enhancing E-Mail Security With Procmail

procmail (the default local delivery agent typically) has a wide variety of features that can be used to help "sanitize" email. More information on this is available at: <ftp://ftp.rubyriver.com/pub/jhardin/antispam/procmail-security.html>.

## SSL wrapping POP and IMAP servers

```
simap stream tcp nowait root /usr/sbin/stunnel imapd -l imapd
```

```
RANDFILE = stunnel.rnd  
[ req ]  
default_bits = 1024  
encrypt_key = no  
distinguished_name = req_dn  
x509_extensions = cert_type  
[ req_dn ]  
countryName = Country Name (2 letter code)  
organizationName = Organization Name (eg, company)  
0.commonName = Common Name (FQDN of your server)  
[ cert_type ]  
nsCertType = server
```

```
openssl req -new -x509 -days 365 -config /etc/stunnel.cnf -out /etc/stunnel.pem -keyout stunnel.pem  
openssl x509 -subject -dates -fingerprint -noout -in stunnel.pem
```

Outlook

Note on Outlook express, you cannot view certificates, and as long as it is for the right site and the date is correct it will be accepted (it can be signed by anyone).

Netscape

Netscape walks you through the traditional certificate dialog and self signed certs will generate a warning.

## Non-commercial mailing list software

mailman

[http://sourceforge.net/project/showfiles.php?group\\_id=103](http://sourceforge.net/project/showfiles.php?group_id=103)

for links to download all the patches and the source tarball. If you decide to install the patches, please do read the release notes first:

[http://sourceforge.net/project/shownotes.php?release\\_id=63042](http://sourceforge.net/project/shownotes.php?release_id=63042)

Currently the SourceForge and [www.list.org](http://www.list.org) sites are up-to-date, and I expect the gnu.org site to be updated soon.

See also:

<http://www.gnu.org/software/mailman>

<http://www.list.org>

<http://mailman.sf.net>

I've also included links on the FAQ page to the Mailman FAQ wizard. Thanks everybody for contributing good entries! (I may do some reorg when I get a chance.) See the FAQ wizard at

<http://www.python.org/cgi-bin/faqw-mm.py>

## **SmartList**

<http://www.procmail.org/>

## **Majordomo**

<http://www.greatcircle.com/majordomo/>

## **Minordomo**

<http://www.nodomainname.net/software/minordomo/>

## **Sympa**

<http://listes.cru.fr/sympa/>

## **Listar**

<http://www.listar.org/>

<http://www.linux-magazin.de/ausgabe/2001/06/Amavis/amavis.html>,

<http://www.linux.ie/pipermail/ilug/2001-November/039565.html>

<http://www.computer-networking.de/~link/security/amavis-patch-old.php3#rlsmtppatch1a>

<http://www.linuxjournal.com/article.php?sid=4882>

---

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **File / print servers**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

### **Overview**

There are many ways to share resources over a LAN. Your main concern will be what the client side is running since most server operating systems (especially Linux) support almost all types of clients (UNIX, Windows, MAC, Novell, etc.). You will also need to take into account the type of files you are sharing, are they simply data files, work documents, source code, network boot files, or?

### **Network booting**

#### **tftp**

tftp is used by everything from X windows terminals to Cisco routers when booting to get their initial boot files and configuration data.

### **UNIX file sharing**

#### **NFS**

NFS is the most universal method of file sharing supported by UNIX in general. Almost every UNIX OS (Linux, \*BSD, Sun, etc.) supports NFS. There are also commercial NFS clients and servers for Windows. NFS is ideal for sharing out user home directories and other real time" filesystems.

#### **rsync**

rsync is the ideal method for synchronizing large amounts of data that isn' t time critical (i.e. for ftp site mirroring). It uses an extremely efficient algorithm to find files that are newer (or gone), and then retrieves them, it also has several nice security features.

### **Printing under Linux**

There are a variety of print daemons for Linux but they generally emulate lpd (the original).

## **lpd**

lpd is the age-old line printer daemons (when all you ever printed was text) which allows for the usage and sharing of printers.

## **CUPS**

Common UNIX Printing System (CUPS), is GPL licensed and version 1.0 just came out. CUPS is available from: <http://www.cups.org/>.

## **LPRng**

LPR next generation, an alternative to the stock LPR.

## **pdq**

## **Windows file and print sharing**

### **Samba**

SMB (server message block) is the current windows file sharing protocol. Samba does an incredible job of providing all the services required to properly share windows files (such as Primary and Backup Domain Controller services). You can also provide windows access to printers through Samba, and using smbclient access Windows printers.

## **General file sharing**

There are also a number of generic file sharing methods that support multiple types of clients and servers.

## **Coda**

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

## **Drall**

An https-based system for sharing files among machines securely. You can get it from: <http://www.edlund.org/hacks/drall/index.html>.

## **AFS**

A high end, commercial file sharing protocol suitable for large installations with high security and performance requirements.

## **Source code sharing**

### **CVS**

CVS is used to centrally maintain source code in a repository, and to allow people to make modifications, with an emphasis on the ability to roll back changes, get an old "snapshot" and so on. It is very popular for large software projects.

[Back](#)

Last updated on 31/8/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Network services - FTP**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## **Overview**

FTP used to be the most used protocol on the Internet by sheer data traffic until it was surpassed by HTTP a few years ago (yes, there was a WWW-free Internet once upon a time). FTP does one thing, and it does it well, transferring of files between systems. The protocol itself is insecure, passwords, data, etc is transferred in cleartext and can easily be sniffed, however most ftp usage is 'anonymous' , so this isn' t a **h**problem. One of the main problems typically encountered with ftp sites is improper permissions on directories that allow people to use the site to distribute their own

data (typically copyrighted material, etc). Again as with telnet you should use an account for ftping that is not used for administrative work since the password will be flying around the network in clear text.

Problems with ftp in general include:

- Clear text authentication, username and password.
- Clear text of all commands.
- Password guessing attacks
- Improper server setup and consequent abuse of servers
- Several nasty Denial of Service attacks still exist in various ftp servers
- Older version of WU-FTPD and derivatives have root hacks

Securing FTP isn't ~~td~~ bad, between firewalling and TCP\_WRAPPERS you can restrict access based on IP address / hostname quite well. In addition most ftp servers run chroot'ed by default for anyone anonymous access, or an account defined as guest. With some amount of work you can set all users that are ftping in to be chroot'ed to their home directory or wherever appropriate. You can also run ftp servers that encrypts the data (using such things as SSL/etc.) however this means your ftp clients must speak the encryption protocol, and this isn't always practical. Also make very sure you have no publicly accessible directories on your ftp server that are both readable and writeable, otherwise people will exploit it to distribute their own software (typically warez or porn).

An example of firewalling rules:

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

or

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

An example of the same using TCP\_WRAPPERS in /etc/hosts.allow:

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

And in /etc/hosts.deny:

```
in.ftpd: 0.0.0.0/0.0.0.0
```

There are several encrypted alternatives to ftp as mentioned before, SSLeay FTPD, and other third party utils. Since most ftp accounts are not used as admin accounts (cleartext passwords, you have been warned), and hopefully run chroot'ed, the security risk is minimized. Now that we have hopefully covered all the network based parts of ftp, lets go over securing the user accounts and environment.

# FTP servers

There are numerous ftp server software packages available for Linux. The popular ones (Wu-FTP and ProFTP) have had a severe number of problems, so make sure your version is up to date.

## vsftpd

Ok, the thing deserves a 1.x release version now, it seems to be doing useful work on various production sites.

Before the 1.0 release details, something potentially interesting: my next project. Before I start investigating the feasibility, I want to judge the demand. I'm considering a "vsshd", which would be a very cut down/minimal server-only implementation of the ssh2 protocol. The intended audience would be paranoid people who want no-frills secure remote access. I'm not saying the current sshd implementations are insecure; however, their design leaves something to be desired. In particular there seems to be rather too much use of "root" (witness the severity of the deattack.c flaw). I am tempted to investigate the possibility of writing a minimal sshd where all protocol parsing and in particular SSL code runs as non-root in a chroot() environment.

vsftpd-1.0.0

<ftp://ferret.lmh.ox.ac.uk/pub/linux/vsftpd-1.0.0.tar.gz>

Approximate changelog:

- Fix build on RedHat7.2
- Fix build on Mandrake systems

README:

vsftpd is an FTP server, or daemon. The "vs" stands for Very Secure. Obviously this is not a guarantee, but a reflection that I have written the entire codebase with security in mind, and carefully designed the program to be resilient to attack.

Recent evidence shows that vsftpd is also extremely fast and scalable. vsftpd has achieved ~4000 concurrent users on a single machine, in a production environment.

vsftpd is now a proven stable solution. Of particular note, RedHat used vsftpd to enable [ftp.redhat.com](http://ftp.redhat.com) to support 15,000 concurrent users across their server pool. This extreme load was generated by the release of RedHat 7.2 to the world.

Cheers  
Chris

## ProFTPD

ProFTPD is a GPL licensed ftp server that can run on a variety on UNIX platforms. It supports newer features such as virtual ftp, per directory configuration (using .ftpassess files similar to Apache's .htaccess files), support for expired accounts and more. It also supports really useful features such as limiting downloads and much tighter security controls then WU-FTP. I highly recommend it over any other freely available FTP server for UNIX.

ProFTPD's main configuration file is /etc/proftpd.conf, it has a rather Apache-esque configuration style which I like a lot. ProFTPD can be run from inetd (and make use of TCP\_WRAPPERS) or it can be run as a stand-alone server. It also supports per directory config files to limit access and so forth. ProFTPD supports virtual ftp as well (although unlike virtual www serving, extra IP addresses are required) and each site can be configured differently (different anonymous access, if any, and more things along those lines). The general proftpd.conf typically has a section covering global settings (inetd or standalone, maximum number of processes to run, who to run as, and so on), followed by a default config, followed by specific site (virtual sites) configuration. On a server doing virtual hosting it is probably a good idea to turn "DefaultServer" off, so any clients ftping in aimlessly are denied instead of being dumped into a default site.

Sample configuration for a ProFTPD server being run from inetd with no anonymous access:

```
ServerName "ProFTPD Default Installation"  
ServerType inetd  
DefaultServer on  
Port 21  
Umask 022  
MaxInstances 30
```

```
User nobody
Group nobody
<Directory /*>
AllowOverwrite on
</Directory>
```

Let's say, like me, that you are paranoid and want to control access to the ftp server by IP addresses, hostnames and domain names (although I would recommend only relying on IP's). You could accomplish via firewall rules, but that tends to slow the machine down (especially if you are adding lots of rules as would be prone to happen). You could use TCP\_WRAPPERS, but you wouldn't be able to selectively limit access to virtual sites, anonymous sites, just the server itself. Or you could do it in the proftpd.conf file using the "<Limit LOGIN>" directive.

The following example will limit access to 10.1.\*.\* and 1.2.3.4, all other machines will be denied access.

```
<Limit LOGIN>
Order Allow,Deny
Allow from 10.1., 1.2.3.4
Deny from all
</Limit>
```

If you place this within a "<VirtualHost>" or "<Anonymous>" directives it applies only to that virtual site or anonymous setup, if placed in a "<Global>" directive it will apply to all the "<VirtualHost>" and "<Anonymous>" sections, and if placed in the server config (i.e. with the "ServerName" and related items) it will behave like TCP\_WRAPPERS would, anyone not from 10.1.\*.\* or 1.2.3.4 immediately gets bumped when they try to connect to port 21, as opposed to simply being denied login if it's in a "<Global>", "<VirtualHost>" or "<Anonymous>" section.

If you want to add anonymous access simply append:

```
<Anonymous ~ftp>
User ftp
Group ftp
RequireValidShell off
UserAlias anonymous ftp
MaxClients 10
DisplayLogin welcome.msg
DisplayFirstChdir .message
<Directory *>
<Limit WRITE>
DenyAll
</Limit>
</Directory>
</Anonymous>
```

This would assign the "ftp" users home directory (assuming a normal setup "~ftp" would probably be /home/ftp) as the root anonymous directory, the ProFTPD would run as the user "ftp" and group "ftp" when people log in anonymously (as opposed to logging in as a normal user), and anonymous logins would be limited to 10. As well the file /home/ftp/welcome.msg would be displayed when anonymous users ftp in, and any directory with a .

message file containing text would be displayed when they changed into it. The “<Directory \*>” covers /home/ftp/\*, and then denies write access for all, meaning no-one can upload any files. If you wanted to add an incoming directory simply add the following after the “<Directory \*>” directives:

```
<Directory incoming>
<Limit WRITE>
AllowAll
</Limit>
<Limit READ>
DenyAll
</Limit>
</Directory>
```

This would allow people to write files to /home/ftp/incoming/, but not read (i.e. download) them. As you can see ProFTPD is very flexible, this results in ProFTPD requiring more horsepower than WU-FTP, but it is definitely worth it for the added control. You can get ProFTPD and the documentation from: <http://www.proftpd.net/>.

### **proftpd-ldap**

proftpd-ldap allows you to do password look ups using an LDAP directory, you can download it from: <http://horde.net/~jwm/software/proftpd-ldap/>.

## **WU-FTP**

WuFTP used to have numerous security issues however 2 years ago a security audit was done by SuSE and many of the problems were removed. If you use WuFTP make sure it is a descendant of the audited version (it should be unless the vendor is using an ancient codebase).

One of the main security mechanisms in WU-FTP is the use of chroot. For example; by default all people logging in as anonymous have /home/ftp/ set as their “root” directory. They cannot get out of this and say look at the contents of /home/ or /etc/. The same can be applied to groups of users and / or individuals, for example you could set all users to be chroot'ed to /home/ when they ftp in, or in extreme cases of user privacy (say on a www server hosting multiple domains) set each user chroot'ed to within their own home directory. This is accomplished through the use of /etc/ftpaccess and /etc/passwd (man ftpaccess has all the info). I will give a few examples of what needs to be done to accomplish this since it can be quite confusing at first. ftpd also checks /etc/ftpusers and if the user attempting to login is listed in that file (like root should be) it will not let the user login via ftp.

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

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

First off billybob needs a properly setup user account in /etc/passwd:

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

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

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

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

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

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

and with the line added to /home/billybob/etc/passwd:

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

and with a line for billybob's group added to /home/billybob/etc/group:

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

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

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

## **NcFTPd**

NcFTPd is a high volume ftp server, however it is only free for personal or .edu usage. You can get it from: <http://www.ncftpd.com/ncftpd/>.

## **FTP - SSL**

Also a drop in replacement for your favorite ftpd (probably WU-FTPd), also available as a set of patches for WU-FTPd. This is highly appropriate as most servers have many users that require ftp access. The tarball is available at: <ftp://ftp.uni-mainz.de/pub/internet/security/ssl/>.

## **FTP - SRP**

SRP can also be used to encrypt the username/password login portion of your ftp session, or the entire session. You can get SRP at <http://srp.stanford.edu/srp/> and it is covered in the LASG [here](#).

## **sftp**

sftp runs over ssh which makes for relatively ftp sessions. You can get it from: <http://www.xbill.org/sftp/>.

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Linux LDAP servers**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## **Overview**

Lightweight directory access protocol seems to be the future of storing user information (passwords, home directories, phone numbers, etc.). Many products (ADS, NDS, etc.) support LDAP interfaces, making it important for Linux to support LDAP as it will be required to tie it into future enterprise networks.

## **LDAP servers**

### **OpenLDAP**

OpenLDAP is a completely opensource (note it is not GPL) package that provides an LDAP server, replication server and utilities. You can get it from: <http://www.openldap.org/>.

### **LDAP authentication**

#### **NSS LDAP Module**

The NSS LDAP Module allows you to do user authentication via LDAP. You can get it from: [http://www.padl.com/nss\\_ldap.html](http://www.padl.com/nss_ldap.html).

### **LDAP tools**

#### **web2ldap**

web2ldap is a Python program that runs as a cgi and provides a www interface to an LDAP directory. You can get it: <http://www.web2ldap.de/>.

#### **kldap**

A KDE based LDAP browsing tool with the ability to edit objects (basically an LDAP admin tool). You can get it at: <http://www.mountpoint.ch/oliver/kldap/>.

#### **GQ**

A GTK based LDAP client that can modify settings/rtc. Available from: <http://biot.com/gq/>

#### **LDAPExplorer**

A www based admin tool for LDAP. Available from: <http://igloo.its.unimelb.edu.au/LDAPExplorer/>.

# Perl/Java/C SDK' for LDAP

A variety of Software Development Kits for LDAP, available from: <http://www.mozilla.org/directory/>

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Network services - NNTP

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## Overview

NNTP (network news transfer protocol) is useful for sharing large amounts of information among many servers. It is also useful for holding discussions and forums on topics like cryptography.

## NNTP server software

### INN

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

One of the main security threats with INN is resource starvation on the server. If someone decides to flood your server with bogus articles or there is a sudden surge of activity you might be in trouble if capacity is lacking. INN has had several bad security holes in past, but with today's environment the programmers seem to have chased down and eliminated all of them (none have surfaced recently). It is highly recommended (for more than security reasons alone) that you place the news spool on a separate disk system, let alone partition. You might also wish to use ulimit to restrict the

amount of memory available so that it cannot bring the server to it's knees.

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

example of `nnrp.access`:

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

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

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

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

or

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

## Diablo

Diablo is free software aimed at backbone news transport, that is to say accepting articles from other NNTP servers and feeding them on to other servers, it is not aimed at use by end users for reading or posting. You can get Diablo at: <http://www.openusenet.org/diablo/>

## DNews

A commercial NNTP server for various platforms. Available from: <http://netwinside.com/dnews.htm>.

# Cyclone

Cyclone is a commercial NNTP server aimed at backbone news transport, that is to say accepting articles from other NNTP servers and feeding them on to other servers, it is not aimed at use by end users for reading or posting. You can get Cyclone at:

<http://discussion.openwave.com/cyclone/cyclone.html>

# Typhoon

Typhoon is a commercial NNTP server aimed at end user news access, that is to say allowing users to post and read articles. You can get Typhoon at:

<http://discussion.openwave.com/breeze/typhoon.html>

Breeze

<http://discussion.openwave.com/breeze/breeze.html>

twister

<http://discussion.openwave.com/twister/twister.html>

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Proxy software

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

Overview

There are a variety of proxy software packages for Linux. Some are application level (such as SQUID) and others are at the session level (such as SOCKS).

Application proxy server software

# SQUID

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

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

Most network administrators will want to configure Squid so that an internal network can access www sites on the Internet. In this example 10.0.0.0/255.255.255.0 is the internal network, 5.6.7.8 is the external IP address of the Squid server, and 1.2.3.4 is a www server we want to see.

Squid should be configured so that it only listens for requests on its internal interface, if it were listening on all interfaces I could go to 5.6.7.8 port 3128 and request http://10.0.0.2/, or any internal machine for that matter and view www content on your internal network. You want something like this in your squid.conf file:

```
tcp_incoming_address 10.0.0.1
tcp_outgoing_address 5.6.7.8
udp_incoming_address 10.0.0.1
udp_outgoing_address 5.6.7.8
```

This will prevent anyone from using Squid to probe your internal network.

On the opposite side of the coin we have people that use Squid to make internal www servers accessible to the Internet in a controlled manner. For example you may want to have an IIS 4.0 www server you want to put on the Internet, but are afraid to connect it directly. Using Squid you can grant access to it in a very controlled manner. In this example 1.2.3.4 is a random machine on the Internet, 5.6.7.8 is the external IP address of the Squid server, 10.0.0.1 is its internal IP address, and 10.0.0.2 is a www server on the internal network running IIS 4.0.

To set Squid up to run as an accelerator simply set the "http\_port" to 80 in squid.conf:

```
http_port 3128
```

And then set the IP addresses differently:

```
tcp_incoming_address 5.6.7.8
tcp_outgoing_address 10.0.0.2
udp_incoming_address 5.6.7.8
udp_outgoing_address 10.0.0.2
```

And finally you have to define the machine you are accelerating for:

```
httpd_accel_host 10.0.0.2
httpd_accel_port 80
```

This is covered extensively in the Squid FAQ at: <http://www.squid-cache.org/Doc/FAQ/FAQ.html> (section 20).

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

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

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

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

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

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

One important security issue most people overlook with Squid is the log files it keeps. By default Squid may or may not log each request it handles (depends on the config file), from 'http://www.nsa.gov/' to 'http://www.example.org/cgi-bin/access&member=john&password=bob'. You definitely want to disable the access logs unless you want to keep a close eye on what people view on the Internet (legally this is questionable, check with your lawyers). The directive is "cache\_access\_log" and to disable it set it to "/dev/null", this logs ALL accesses, and ICP queries (inter-cache communications). The next big one is the "cache\_store\_log", which is actually semi useful for generating statistics on how effective your www cache is, it doesn't log who made the request, simply what the status of objects in the cache is, so in this case you would see the pictures on a pornographic

site being repeatedly served, to disable it set it to “none”. The “cache\_log” should probably be left on, it contains basic debugging info such as when the server was started and when it was stopped, to disable it set it to “/dev/null”. Another, not very well documented log file is the “cache\_swap\_log” file, which keeps a record of what is going on with the cache, and will also show you the URL’s people are visiting (but not who/etc), setting this to “/dev/null” doesn’t work (in fact Squid pukes out severely) and setting it to “none” simply changes the filename from “log” to “none”. The only way to stop it is to link the file to “/dev/null” (by default the root of the www cache files /log), and also to link the “log-last-clean” to “/dev/null” (although in my quick tests it doesn’t appear to store anything you can’t be sure otherwise). So to summarize:

in squid.conf:

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

and link:

```
/var/spool/squid/log to /dev/null
/var/spool/squid/log-last-clean to /dev/null
```

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

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

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

or in ipchains:

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

## **squidGuard**

squidGuard allows you to put in access control lists, filter lists, and redirect requests, easily and efficiently. It is ideal for controlling access to the WWW, and for more specific tasks such as blocking pornographic content (a valid concern for many people). It cannot make decisions based upon content however, it simply looks at the URL’s being processed, so it cannot be used to block active content and so on. squidGuard is available from: <http://www.squidguard.org>

## **LDAP auth module for SQUID**

This allows you to authenticate users via an LDAP server, however passwords/etc are transmitted in the clear, so use some form of VPN to secure it. You can get it from: [http://www.stroeder.com/proxy\\_auth\\_ldap/](http://www.stroeder.com/proxy_auth_ldap/)

## **Cut the crap**

Cut the crap (CTC) is aimed at blocking banner ads and reducing bandwidth usage while surfing. You can get it from: <http://www.softlab.ece.ntua.gr/~ckotso/CTC/>.

## **WWWOFFLE**

WWWOFFLE is a rather nice looking proxy for UNIX systems that handles HTTP and FTP. You can get it at: <http://www.gedanken.demon.co.uk/wwwoffle/>.

## **Circuit level proxy software**

### **SOCKS**

SOCKS is a circuit level proxy, typically loaded on firewalls because it has good access controls. Applications must be SOCKS' fied, most popular web browsers, ftp clients and so on have support by default. You can get it from: <http://www.socks.nec.com/>.

### **Dante**

Dante is a free implementaiton of the popular SOCKS server. It is available from: <http://www.inet.no/dante/>.

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Shell servers**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

Getting access to your server remotely is critical for most administrators. Most of us cannot sit down at the console and login, and in any case remote access is much less effort in the long run. Doing this securely is the trick of course.

## Telnet

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

Problems with telnet include:

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

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

An example of firewalling rules:

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

or in ipchains:

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

An example of the same using TCP\_WRAPPERS, in /etc/hosts.allow:

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

And in /etc/hosts.deny:

```
in.telnetd: ALL
```

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

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

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

in /etc/shells list:

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

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

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

and voila. The user telnets to the server, is prompted for their username and password, and is then prompted to change their password. If they do so successfully passwd then exits and they are disconnected. If they are unsuccessful passwd exits and they are disconnected. The following is a transcript of such a setup when a user telnets in:

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

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

Telnet also displays a banner by default when someone connects. This banner typically contains systems information like the name, OS, release and

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

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

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

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

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

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

## Telnet - SSL

## SSLtelnet and MZtelnet

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

## SSH - server and client software

SSH is a secure protocol and set of tools to replace some common (insecure) ones. It was designed from the beginning to offer a maximum of security and allows remote access to servers in a secure manner. SSH can be used to secure any network based traffic, by setting it up as a 'pipe' i(e.

binding it to a certain port at both ends). This is quite kludgy but good for such things as using X across the Internet. In addition to this the server components runs on most UNIX systems, and NT, and the client components runs on pretty much anything. Unfortunately SSH is no longer free; however, there is a project to create a free implementation of the SSH protocol. There aren't any problems with SSH per se like there are with telnet, all session traffic is encrypted and the key exchange is done relatively securely (alternatively you can preload keys at either end to prevent them from being transmitted and becoming vulnerable to man in the middle attacks).

## OpenSSH

Use openssh, available from <http://www.ssh.com/>. OpenSSH ships with almost all Linux distributions and typically installs by default. SSH typically runs as a daemon, and can easily be locked down by using the sshd\_config file.

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

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

or

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

Or via TCP\_WRAPPERS, hosts.allow:

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

hosts.deny:

```
sshd: 0.0.0.0/0.0.0.0
```

In addition to this, ssh has a wonderful configuration file, /etc/sshd/sshd\_config by default in most installations. You can easily restrict who is allowed to login, which hosts, and what type of authentication they are allowed to use. The default configuration file is relatively safe but following is a more secure one with explanations. Please note all this info can be obtained by a "man sshd" which is one of the few well written man pages out there. The following is a typical sshd\_config file:

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

Other useful sshd\_config directives in 1.2.x include:

AllowGroups - explicitly allow groups (/etc/group) to login using ssh  
DenyGroups - explicitly disallows groups (/etc/groups) from logging in  
AllowUsers - explicitly allow users to login in using ssh  
DenyUsers - explicitly blocks users from logging in  
IdleTimeout time - time in minutes/hours/days/etc, forces a logout by SIGHUP'ing the process.

These directives appear to have been removed from current OpenSSH versions:

AllowHosts - allow certain hosts, the rest will be denied  
DenyHosts - blocks certain hosts, the rest will be allowed

Make sure your sshd is compiled with tcp\_wrappers support and use hosts.allow and hosts.deny instead.

## **SSH - client software:**

### **Fresh Free FiSSH**

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

### **Tera Term**

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

### **putty**

putty is a Windows SSH client, pretty good, and completely free, and also small (184k currently). You can download it from: <http://www.chiark.greenend.org.uk/~sgtatham/putty.html>.

## **mindterm**

mindterm is a free java ssh client, you can get it at: <http://www.appgate.org/products/mindterm/>

## **The Java Telnet Application**

The Java Telnet Application supports ssh, and is free, you can get it at: <http://www.mud.de/se/jta/>.

## **Secure CRT**

A commercial Telnet / SSH client from Vandyke software. You can download / purchase it at: <http://www.vandyke.com/>.

## **Fsh**

Fsh stands for “Fast remote command execution” and is similar in concept to rsh/rcp. It avoids the expense of constantly creating encrypted sessions by bringing up an encrypted tunnel using SSH or LSH, and running all the commands over it. You can get it from:

<http://www.lysator.liu.se/fsh/>.

## **SSH Win32 ports**

Ports of SSH to Win32 available at: <http://guardian.htu.tuwien.ac.at/therapy/ssh/>.

## **SRP**

SRP is a relative newcomer, however it has several advantages over some of the older programs. SRP is free and does not use encryption per se to secure the data, so exporting it outside of the US isn't as much of a problem (there is a version that encrypts and is available within the US and Canada, and interoperates with the non encrypting version of SRP). SRP is available from <http://srp.stanford.edu/>. SRP currently has Telnet and FTP support (for windows as well) although SRP enabling other protocols is relatively straightforward. A windows client with SRP capabilities is available at: <http://www.kermit-project.org/k95.html>.

## **NSH**

NSH is a commercial product with all the bells and whistles (and I do mean all). It's got built in support for encryption, so it's relatively safe to use (I

cannot verify this completely however, as it isn't open source). Ease of use is high, you cd //computername and that 'logs' you into that computer, you can then easily copy/modify/etc. files, run ps and get the process listing for that computer, etc. NSH also has a Perl module available, making scripting of commands pretty simple, and is ideal for administering many like systems (such as workstations). In addition to this NSH is available on multiple platforms (Linux, BSD, Irix, etc.) with RPM's available for Red Hat systems. NSH is available from: <http://www.networkshell.com/>, and 30 day evaluation versions are easily downloaded.

## R services

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

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

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

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## SNA connectivity

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

Overview

SNA is a very common network protocol that hails back to the days of IBM and "heavy iron".

SNA software

<http://www.linux-sna.org/>

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Network services - SNMP

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

### Overview

SNMP (Simple Network Management Protocol) was designed to let heterogeneous systems and equipment talk to each other, report data and allow modifications to their settings over a TCP-IP network. For example an SNMP enabled device (such as a Cisco router) can be monitored/configured from an SNMP client, and you can easily write scripts to, say, alert you if denied packets/second rises above 20. Unfortunately SNMP has no security built into it. SNMPv1, originally proposed in RFC 1157 (May 1990) and section 8 (Security Considerations) reads thusly: "Security issues are not discussed in this memo." I think that about sums it up. In 1992/1993 SNMPv2 was released, and did contain security considerations however these security considerations were dropped later on when they were shown to be completely broken. Thus we end up today with SNMPv2 and no security.

Currently the only way to protect your SNMP devices consists of setting the community name to something hard to guess (but it is very easy to sniff the wire and find the name), and firewall/filter SNMP so that only the hosts that need to talk to each other can (which leaves you open to spoofing). Brute force community name attacks are easy to do and usually effective, and there are several tools specifically for monitoring SNMP transmissions and cracking open an SNMP community, it is a pretty dangerous world out there.

These risks are slightly mitigated by the usefulness of SNMP, if properly supported and implemented it can make network administration significantly easier. In almost every SNMP implementation the default community name is "public" (this goes for Linux, NT, etc), you must change this, to something obscure (your company name is a bad idea). Once a person has your community name they can conduct an "snmpwalk" and take over your network. SNMP runs over UDP on ports 161 and 162; block this at all entrances to your network (the backbone, the dialup pool, etc). If a segment of network does not have SNMP enabled devices or an SNMP console you should block SNMP to and from that network. This is your only real line of defense with SNMP.

Additionally the use of IPSec (or other VPN software) can greatly reduce the risk from sniffing. The RFC' s for SNMPv3 however go extensively into security (especially RFC 2274, Jan 1998) so there is hope for the future. If you are purchasing new SNMP aware/enabled products make sure they support SNMPv3, as you then have a chance at real security.

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

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

or

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

## SNMP server software

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Network services - NTP

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

### Overview

NTP (Network Time Protocol) is rather simple in it's mission, it keeps computers clocks in synchronization. So what? Try comparing log files from 3 separate servers if their clocks are out of synch by a few minutes. NTP simply works by a client connecting to a time server, working out the delay between them (on a local LAN it might be only 1-2ms, across the internet it might be several hundred ms), and then it asks for the time and sets it's own clock. Additionally servers can be 'clustered' to keep themselves synchronized, the chances of 3 or more servers losing track of what time it is (also called 'drift') is relatively low.

The time signal is typically generated by an atomic clock or GPS signal, measured by a computer, these are 'stratum 1' time servers, below them are

stratum 2 time servers that typically are publicly accessible, a company might maintain it's own stratum 3 time servers if they have sufficient need, and so on.

The data NTP exchanges is of course not terribly sensitive, it's a time signal, however if an attacker were able to tamper with it, all sorts of nastiness could result: log files might be rendered unusable, accounts might be expired early, cron jobs that backup your server might run in prime time causing delays, etc. Thus it is a good idea to run your own time server(s), and set the maximum adjustment they will make to only a few seconds (they shouldn't drift very much in any case). If you are really paranoid, or have a great number of clients you should consider buying a GPS time unit.

They come in all shapes and sizes, from a 1U rack mount job that plugs directly into your LAN to ISA and PCI cards that plug into a server and have an antenna. It is a good idea to firewall off your timeserver, as a denial of service attack on it would be detrimental to your network. In addition to this if possible you should use the encryption available in ntpd, based on DES it is generally sufficient to thwart most attackers. NTP runs on port 123 using udp (and when you connect to servers they will come from port 123 to your port 123), so firewalling it is relatively simple:

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

or

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

## NTP server software

### XNTP

XNTP is available from: <http://www.eecis.udel.edu/~ntp/>. There usually are no man pages with ntpd or xntpd (wonderful huh?) but documentation can be found in /usr/doc/ntp-xxxx/, or at: [http://www.eecis.udel.edu/~ntp/ntp\\_spool/html/index.htm](http://www.eecis.udel.edu/~ntp/ntp_spool/html/index.htm).

## NTP client software

### ntpdate

ntpdate ships with most distributions as part of xntp.

There is a version of ntp that uses the Linux kernel capability set system time and does not need to run as root (drops privileges once it binds to port 123). Need to find this.

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## User information

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

## Overview

There are a variety of services that can provide information about local users to other local users, and other machines. These can be useful if you want to find out which user connected to a machine, or see when they last logged in. Of course these are great services for attackers since they can glean a lot of information from them.

## Ident server software

The ident service is used to map users/processes to ports in use. For example most IRC servers attempt to find out who is connecting to them by doing an ident lookup, which basically consists of asking the ident server on the client computer what information it has about a port number, and the response can range from nothing (if no-one is using that particular port) to a username, groupname, process id, and other interesting information. The default setting in most distributions is that `identd` is on (it is polite to run it, irc servers and newer versions of sendmail check `identd` responses), and will only hand out the username. The primary use of `identd` is to allow remote systems some means of tracking down users that are connecting to their servers, irc, telnet, mail, or other, for authentication purposes (not a good idea since it is very easy to fake). The local university here in Edmonton requires you to run `identd` if you want to telnet into any of the main shell servers, primarily so they can track down compromised accounts quickly.

Running `identd` on your machine will help other administrators when tracking down problems, as they can not only get the IP address and time of a problem, but using `identd` can look up the user name. In this way it is a two edged sword, while it gives out information useful for tracking down malicious users (definitely people you want to boot off of your servers) it can also be used to gain information about users on your system, leading to their accounts being compromised. Running `identd` on servers only makes sense if they are hosting shell accounts/etc.

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

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

or

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

## Identd

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

`-p port`

`-a address`

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

`-u uid`

`-g gid`

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

`-o`

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

`-n`

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

`-N`

Allows users to make a `~/.noident` file, which will force `identd` to return "HIDDEN-USER" instead of information. This allows users the option of having a degree of privacy, but a malicious user will use this to evade identification.

`-F format`

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

In general I would advise running `identd` on servers with user shell accounts, and otherwise disabling it, primarily due to the number of denial of service attacks it is susceptible to. Running `identd` will make life a lot easier for other administrators when tracking down attacks originating from your site, which will ultimately make your life easier.

## Other Identd daemons

There are also other versions of identd available, some with security enhancements (I do not endorse these as I have yet to test them):

<http://www.tildeslash.org/nullidentd.html> - null identd

<http://www.ajk.tele.fi/~too/sw/> - fake identd

<http://p8ur.op.het.net/midentd/> - midentd

## Finger server software

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

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

If you still have the urge that you absolutely must run it use -u to deny finger @host requests that are only ever used to gather information for future attacks. Disable finger, really. Fingerd has also been the cause of a few recent and very bad denial of service attacks, especially if you run NIS with large maps, DO NOT, repeat NOT run fingerd. Finger runs on port 79, and cfingerd runs on port 2003, both use tcp.

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

or

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

# PFinger

PFinger is similar to Cfingerd in that it is a secure replacement for the stock fingerd. You can get PFinger from: <http://www.xelia.ch/unix/pfinger/>.

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Network services - HTTP / HTTPS

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

### Overview

WWW traffic is one of the largest components of Internet usage today. There are a variety of popular WWW servers for Linux, the most popular of course being Apache (with over %50 of the market). Most modern WWW servers also have the capability to use SSL to secure sessions (for e-commerce and so on). This section is very Apache-centric, but since this is the default www server for almost all Linux (and \*BSD) distributions it makes sense. I'm also writing for the 1.3.9 version of Apache which no longer uses access.conf or srm.conf, but instead has rolled everything into httpd.conf.

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

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

or in ipchains:

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

HTTPS runs on port 443, tcp, and if it is for internal use only (an Intranet, or www based control mechanism for a firewall server say) you should definitely firewall it.

```
ipfwadm -I -a accept -P tcp -S 10.0.0.0/8 -D 0.0.0.0/0 443
ipfwadm -I -a accept -P tcp -S some.trusted.host -D 0.0.0.0/0 443
ipfwadm -I -a deny -P tcp -S 0.0.0.0/0 -D 0.0.0.0/0 443
```

or in ipchains:

```
ipchains -A input -p all -j ACCEPT -s 10.0.0.0/8 -d 0.0.0.0/0 443
ipchains -A input -p all -j ACCEPT -s some.trusted.host -d 0.0.0.0/0 443
ipchains -A input -p all -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 443
```

## WWW server software

### Apache

What can I say about securing Apache? Not much actually. By default Apache runs as the user ' nobody' , giving very little access to the system, and by and large the Apache team has done an excellent job of avoiding buffer overflows/etc. In general most www servers simply retrieve data off of the system and send it out, most of the danger come not from Apache but from sloppy programs that are executed via Apache (CGI's, server side includes, etc).

If going with Apache, I would recommend using the 1.3 series unless you have some strange requirement for sticking to 1.2, the active development is now on 1.3, and includes many new features from security, usability, stability and performance viewpoints. Most servers based upon Apache (Red Hat Secure Server, Stronghold, etc.) are generally just as bug free but there are occasionally problems. You can download Apache from <http://httpd.apache.org/>. Almost all distributions ship with it, so you should have it as package or whatever.

#### Chroot'ing Apache

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

The simplest way to setup Apache chroot'ed is to simply install it and move/edit the necessary files. A good idea is to create a directory (such as /chroot/apache/), preferably on a separate filesystem from /, /usr, /etc (hard links, accidental filling of partitions, etc...), and then create a file structure under it for Apache. The following is an example, simply replace /chroot/apache/ with your choice. You must of course execute these steps as root for it to work. RPM supports this with the “-root /some/dir” directive, simply install Apache and the needed libs using rpm (thus gaining it' s support for dependencies/etc, making your life easier). If you are not on an RPM based system simply use ldd to find out which shared libraries are required, and move everything needed into your /chroot/apache/ directory.

```
[seifried@host seifried]$ ldd /usr/bin/httpd
libm.so.6 => /lib/libm.so.6 (0x40017000)
libc.so.6 => /lib/libc.so.6 (0x40060000)
```

```
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Apache logs requests and so forth internally, so you don't have to worry about setting up pseudo logging daemons like holelogd to pass information to syslog in order to get your log files behaving.

## Secure configuration of Apache

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

Example for httpd.conf:

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

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

### Controlling access

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

```
AccessFileName .htaccess
```

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

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

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

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

### apache-userdirldap

apache-userdirldap allows you to use an LDAP directory for looking up user home directories. In other words if you want to move all your users into an LDAP directory and do ALL authentication through it, you won't have to break Apache. You can get it at: <http://horde.net/~jwm/software/apache-userdirldap/>.

## **thttpd**

A lightweight http server suitable for web based interfaces and the like (Phoenix Adaptive firewall uses it for htier interface for example). You can get it at: <http://www.acme.com/software/thttpd/>

## **AOL Server**

I know, it sounds strange but it is true. AOL Server is a free www server, with source code available. Not only that but it supports SSL and several other advanced features. Definitely worth taking a look at. You can get it from: <http://aolserver.com/>.

There is more to securing your www server than installing Apache and configuring it properly. Most servers will need to allow access to their filesystems so that users can upload and modify files on the server. For this there are 4 widely used methods that I will cover in detail.

## **webfs**

webfs is a lightweight www server that implements basic functionality and is available from: <http://bytesex.org/webfs.html>

## **Flash Web Server**

A lightweight, fast www server, You can get it at: <http://www.cs.rice.edu/~vivek/flash/>.

## **Secure WWW server software**

There are several free alternatives for Apache with SSL, and several commercial ones. If you are located in the US, RSA is patented, so you either have to use DSA (which is hard to get site certificates for) or buy a commercial server based on Apache (like Stronghold). If you are located in Europe you may live in a country where IDEA is patented, so make sure you check first. There are also a variety of commercial packages. I have also written two article covering the various secure www servers:

Web server round up articles need to be put back online, please email [kurt@seifried.org](mailto:kurt@seifried.org) to remind me about this.

# Apache-SSL

This is the one I currently use (simple because I tried it before Apache with mod\_ssl, and it worked). You need to get Open-SSL, compile and install that, and then patch Apache with the Apache-SSL patch, compile Apache, and off you go. Open-SSL is available from: <http://www.openssl.org/>, simply get the latest tarball, unpack it, and run:

```
./config
make
make test
make install
```

It hasn't ever failed for me yet. You then need get the Apache-SSL stuff from <http://www.apache-ssl.org/>, unpack the Apache source somewhere, cd into the top level dir (/usr/local/src/apache\_1.3.9/) and then unpack the Apache-SSL stuff into it (it tells you to do this in the docs, sort of a catch 22). You then simply run:

```
./FixPatch
```

Which should work (if it doesn't read the README.SSL), then configure Apache as usual, and run make, followed by make install. Skip down to the "Creating a certificate" section.

## Apache with mod\_ssl

Apache with mod\_ssl is available from <http://www.modssl.org/>. I haven't tested it yet.

### Creating a certificate

This is the easy part, the next step is to create a key set, and then configure httpsd.conf to use it appropriately. Find where "openssl" is installed and make sure it is in your path, then cd to wherever you placed your apache config files (whatever you prefixed as the apache root followed by /conf/). If you need to create a test certificate, simply to use internally then you can:

```
openssl genrsa -des3 > httpsd.key
openssl req -new -x509 -key httpsd.key > httpsd.crt
```

Browsers will complain loudly about this certificate because it is created by the person who signs it, and they are untrusted. If you want to generate a certificate, and a certificate request to send to someone like Thawte or Verisign then you need to:

```
openssl genrsa -des3 > httpsd.key
openssl req -new -key httpsd.key > httpsd.csr
```

You can also get real certificates with limited life times (usually a week or two) from Verisign to use for testing in a more "real world" environment.

## Configuring Apache for SSL

There are several things you will need to add to your Apache config file to get Apache with SSL extensions actually doing anything useful with your certificates. You'll need to add some global configuration settings (note these apply to 1.3.9, and will barf on earlier versions of Apache):

```
# you will need to tell apache to listen to port 443, by default it only goes for 80
Listen 443
# if you use more then one secure site on an IP (BAD IDEA) you will need:
NameVirtualHost 10.1.1.1:443
#it's a good idea to disable SSL globally and enable it on a per host basis
SSLDisable
# SSL cache server, without this your server will die
SSLCacheServerPath /usr/bin/gcache
# port the cache server runs on
SSLCacheServerPort 12345
# timeout for the SSL cache, set shorter for testing, 300 is a good "real world" value
SSLSessionCacheTimeout 300
```

Now you can create a virtual host with SSL enabled:

```
<VirtualHost www.example.com:443>
DocumentRoot /www/secure/
ServerName www.example.com
ServerAdmin example@example.com
ErrorLog logs/https_error.log
TransferLog logs/https_access.log
# enable SSL for this virtual host
SSLEnable
# this forbids access except when SSL is in use. Very handy for defending
# against configuration errors that expose stuff that should be protected
SSLRequireSSL
SSLCertificateFile /usr/conf/httpsd.crt
# if the key is not combined with the certificate, use this
# directive to point at the key file. [OPTIONAL]
SSLCertificateKeyFile /usr/conf/httpsd.key
# If you want to require users to have a certificate you will need a bundle of
# root certificates so you can verify their personal certificates
#SSLCACertificateFile /etc/ssl/ca-cert-bundle.pem
SSLVerifyClient none
</VirtualHost>
```

## Red Hat Secure Server

Red Hat Secure Server is an Apache based product from (guess who) Red Hat software. Essentially it is stock Apache with RSA cryptographic

modules (which is what you are paying for essentially) and can also serve standard non cryptographic http requests. It can only be sold in the USA and Canada, and is the best option (in my opinion) as far as secure www servers that are legal to use in the US go (due to RSA patents). As far as security goes read the previous section on Apache / Apache-SSL, it all applies. Red Hat Secure Server costs \$100 US and you get a \$25 discount on your Thawte site certificate (so the site certificate only costs \$100 US). I personally like it a lot as it is based on software that runs over half the www sites in the world and as such getting support/updates/etc. is easy. You can buy Red Hat Secure Server from: <http://store.redhat.com/apps/commerce/>

## **Roxen**

Roxen is another commercial www server capable of https and is GPL licensed. You can freely download it if you are in the European Union or Australia, Canada, Japan, New Zealand, Norway, USA, or Switzerland. A version with “weak” (40 bit) crypto can be downloaded without any problems to any country. Roxen is an extremely solid product and is available from: <http://www.roxen.com/>.

## **Zeus**

Zeus is a high end www server that supports SSL. It is a commercial product, and you can get it (a 30 day demo is available) at: <http://www.zeus.com>

## **Netscape Enterprise**

Currently in beta testing (although it installed and runs fine) for Linux, available from: [http://www.iplanet.com/download\\_index/downloads\\_index\\_9\\_0.html](http://www.iplanet.com/download_index/downloads_index_9_0.html)

## **IBM HTTP Server**

IBM also makes an HTTP server for Linux (based on Apache) that you can download from here: <http://www-4.ibm.com/software/webservers/httpservers/download.html>.

## **Accessing your WWW server files**

At some point you will need to access the files on the www server to update them. Logging in and using a text editor like emacs is not usually a good long term decision if you value your time. Several popular HTML authoring packages can access your website via FTP or windows file sharing.

## **FTP**

This is the “classic” method of granting users access to ftp servers, typically concerns include users viewing each others data, viewing system data

they should not, and so forth. Chroot'ing the users ftp session will solve most of these problems, however the main problem with ftp, as for encrypting the username and password this is typically undoable due to the fact most people are running Windows FTP clients. I would recommend ProFTPD over WU-FTPD for an application such as this, ProFTPD has much better access controls.

## Samba access

Samba is quite useful for sharing out the www directories to Windows clients, you can then keep the usernames and passwords separate from the system (using smbpasswd rather than the system passwd) and encryption of logins is no problem. Simply make the shares non browseable, and use the "valid users" directive to restrict which users may view the share data. For example:

```
[www-example]
  path = /www/www.example.org/
  valid users = someuser
  read only = No
  browseable = No
```

will setup a pretty secure share for the directory "/www/www.example.org/" that only the user "example" can access.

## FrontPage access

FrontPage is one of the most popular HTML programs for Windows users (heck, I even use it). It can talk directly to WWW servers and download / upload files from a site (called a "Front Page Site") if the server supports FrontPage extensions. FrontPage extensions are available for various UNIX platforms, for free, from Ready To Run Software, at: [http://www.rtr.com/Ready-to-Run\\_Software/](http://www.rtr.com/Ready-to-Run_Software/). In the past, security wise, RTR's FrontPage extensions for UNIX have been a bit of a disaster. There are commercial alternatives however, one is Instant ASP, available from: <http://www.halcyonsoft.com/>. An excellent document on getting FrontPage working with Apache 1.3.X is available at: <http://www.itma.lu/howto/apache/>.

## RearSite

RearSite is a cgi program that provides users access to their directories via a normal web browser. You can get it from: <http://listes.cru.fr/rs/fd/>.

[Back](#)

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **WWW based email readers**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

### **Overview**

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

### **Non-commercial**

#### **AtDot**

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

#### **acmemail**

<http://www.nodomainname.net/software/atdot/>

#### **IMHO**

<http://www.lysator.liu.se/~stewa/IMHO/>

#### **IMP**

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

## **MailMan**

MailMan is a quasi free (for non comemrcial use) web based gateway for POP email. If you plan to use it commrcially you must buy it. You can get it at: <http://www.endymion.com/products/mailman/>

## **SquirrelMail**

<http://squirrelmail.sourceforge.net/>

## **Commercial**

## **DmailWeb**

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

## **Webmail**

<http://netwinsite.com/webmail/>

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **X Window System**

By Kurt Seifried, [kurt@seifried.org](mailto:kurt@seifried.org)

---

## Overview

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

X is very predictable in port usage, almost all implementations and installations of X use port 6000 for the first session and increment by one for other sessions, thus making it quite easy to scan for. If you are not going to be using X to display program running on remote systems I suggest strongly that you firewall port 6000.

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

or

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

## X server security configuration

There are a number of methods to secure access to your X server. --no-listen-tcp

### xhost

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

## mkxauth

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

## SSH tunnel

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

---

[Back](#)

Last updated on 2/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Software

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

# RPM

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

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

```
package_name-package_version-rpm_build_version-architecture.rpm
```

```
nfs-server-2.2beta29-5.i386.rpm
```

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

## Command Function

- q Queries Packages / Database for info
- i Install software
- U Upgrades or Installs the software
- e Extracts the software from the system (removes)
- v be more Verbose
- h Hash marks, a.k.a. done-o-dial

Command Example	Function
<code>rpm -ivh package.rpm</code>	Install 'package.rpm', be verbose, show hash marks
<code>rpm -Uvh package.rpm</code>	Upgrade 'package.rpm', be verbose, show hash marks
<code>rpm -qf /some/file</code>	Check which package owns a file
<code>rpm -qpi package.rpm</code>	Queries 'package.rpm', lists info
<code>rpm -qpl package.rpm</code>	Queries 'package.rpm', lists all files
<code>rpm -qa</code>	Queries RPM database lists all packages installed
<code>rpm -e package-name</code>	Removes 'package-name' from the system (as listed by rpm -qa)

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

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

## dpkg

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

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

```
packagename_packageversion-debversion.deb
```

```
ncftp2_2.4.3-2.deb
```

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

Command Function:

-I Queries Package

-i Install software

-l List installed software (equiv. to rpm -qa)

-r Removes the software from the system

Command Example	Function
<code>dpkg -i package.deb</code>	Install package.deb
<code>dpkg -I package.deb</code>	Lists info about package.deb (rpm -qpi)
<code>dpkg -c package.deb</code>	Lists all files in package.deb (rpm -qpl)
<code>dpkg -l</code>	Shows all installed packages
<code>dpkg -r package-name</code>	Removes 'package-name' from the system (as listed by dpkg -l)

Debian has 1500+ packages available with the system. You will learn to love dpkg (functionally it has everything necessary, I just miss a few of the

bells and whistles that rpm has, on the other hand dselect has some features I wish rpm had).

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

## tarballs / tgz

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

Command	Function
tar -tf filename.tar	Lists files in filename.tar
tar -xf filename.tar	Extracts files from filename.tar

## Checking file integrity

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

## **RPM integrity**

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

Be warned however there are some potential problems if you do not configure RPM correctly. Please see this security advisory:

<https://www.seifried.org/security/advisories/kssa-001.html>

## **dpkg integrity**

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

## **PGP signed files**

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

## **GnuPG signed files**

Also used is GnuPG, a completely open source version of PGP that uses no patented algorithms. You can get it from: <http://www.gnupg.org/>.

## **MD5 signed files**

Another way of signing a package is to create an MD5 checksum. The reason MD5 would be used at all (since anyone could create a valid MD5

signature of a trojaned software package) is that MD5 is pretty much universal and not controlled by export laws. The weakness is you must somehow distribute the MD5 signatures in advance securely, and this is usually done via email when a package is announced (vendors such as Sun do this for patches).

## **Automating software updates**

### **NSBD**

NSBD (not-so-bad-distribution) is a method to automatically distribute and update software securely over the network. You can get it from: <http://www.bell-labs.com/project/nsbd/>.

## **Automating updates with RPM**

### **AutoRPM**

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

### **RpmWatch**

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

## Automating updates with dpkg

Debian' software package management tools (dpkg and apt-get) support automated updates of packages and all their dependancies from a network ftp server. Simple create a script that is called by cron once a day (or more often if you are paranoid) that does:

```
#!/bin/bash
PATH=/usr/bin
apt-get update
apt-get upgrade
```

The only additional thing you will need to do is configure your download sites in /etc/apt/sources.list and general apt configuration in /etc/apt/apt.conf.

## Automating updates with tarballs / tgz

No tools found, please tell me if you know of any (although beyond mirroring, automatically unpacking and running “./configure ; make ; make install”, nothing really comes to mind, i.e. a ports collection similar to BSD).

## Tracking software installation

Usually when software installs from a source install as opposed to a package it has a tendency to go all over the place. Removing it can be an extremely troublesome task.

### **instmon**

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

## Converting file formats

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

### alien

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

## Finding software

One major problem with Linux is finding software that did not ship with your distribution. Searching the Internet is a pain. There are some resources however that can ease the pain:

- <http://www.rpmfind.net/>
- <http://www.linuxapps.com/>
- <http://www.freshmeat.net/>

## Secure programming

This whole guide exists because Linux and the software running on Linux systems is either insecurely written, or insecurely setup. Many issues, such as buffer overruns, are due to bad programming and carelessness. These problems become especially bad when the software in question is setuid to run as root, or any other privileged group. There are a variety of techniques, and other measures that can be taken to make software safer.

### Secure Linux Programming FAQ

This guide covers a lot of general techniques for secure programming as well as some Linux specific items. You can get it at: <http://www.dwheeler.com/secure-programs/>.

## Secure UNIX Programming FAQ

This document covers a variety of techniques to make programs more secure, as well as some pretty low level items like inherited trust, sharing credentials, and so on. This document is available at: <http://www.whitefang.com/sup/> and I highly recommend reading it if you plan to program in Linux (or UNIX in general).

## Secure Internet Programming

Secure Internet Programming (SIP) is a laboratory (for lack of a better word) that studies computer security, and more specifically problems with mobile code such as Java and ActiveX. They have a number of interesting projects going, and many publications online that make excellent reading. If you are going to be writing Java code I would say you have to visit this site: <http://www.cs.princeton.edu/sip/>.

## Writing Safe Setuid Programs

Writing Safe Setuid Programs is an extremely comprehensive work that covers most everything and is available in HTML format for easy reference. A must read for anyone that uses setuid software, let alone codes it. Available at: <http://nob.cs.ucdavis.edu/~bishop/secprog/>

## userv

userv allows programs to invoke other programs in a more secure manner than is typically used. It is useful for programs that require higher levels of access than a normal user, but you don't want to give root access to. Available at: <http://www.chiark.greenend.org.uk/~ian/userv/>.

## More

<http://www.w3.org/Security/Faq/www-security-faq.html>

<http://security.devx.com/>

<http://members.home.net/razvan.peteanu/>

<http://www.shmoo.com/securecode/>

<http://heap.nologin.net/aspsec.html>

<http://www.javaworld.com/javaworld/jw-12-1998/jw-12-securityrules.html>



## Stackguard

Stackguard is a set of patches for GCC that compile programs to prevent them from writing to locations in memory they shouldn' (simplistic explanation, the Stackguard website has much better details). Stackguard does break some functionality however, programs like gdb and other debuggers will fail, but this is generally not a concern for high security production servers. You can get Stackguard from: <http://www.immunix.org/>.

---

[Back](#)

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Encryption

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

## Encrypting data files and email

Several encryption programs are also available to encrypt your data, some at the file level (PGP, GnuPG, etc.) and some at the drive level (Cryptographic File System for example). These systems are very appropriate for the storage of secure data, and to some degree for the transmission of secure data. However both ends will require the correct software, compatible versions, and an exchange of public keys will somehow have to take place, which is unfortunately, an onerous task for most people. In addition to this you have no easy way of trusting someone's public key unless you receive it directly from them (such as at a key signing party), or unless it is signed by someone else you trust (but how do you get the trusted signer' s key securely?). Systems for drive encryption such as CFS (Cryptographic FileSystem) are typically easy to implement, and only require the user to provide a password or key of some form to access their files. There is a really good article on choosing key sizes at <http://www.cryptosavvy.com/> which raises some issues you probably hadn' t considered. I would recomend reading it.

## **GnuPG (Gnu Privacy Guard)**

GnuPG is covered in the filesystem section [here](#).

### **pgp4pine**

pgp4pine is a PGP shell for pine that allows easy usage of PGP/GnuPG from within pine. Signing / encrypting and so on is made easier. You can get it from: <http://pgp4pine.flatline.de/>

### **Netscape Messenger**

Netscape Messenger supports X.509 certificates, as do most Windows mailer programs.

## **Sources of random data**

In order for encryption to be effective, especially on a large scale such as IPsec across many hosts, good sources of random, cryptographically secure data are needed. In Linux we have /dev/random and /dev/urandom which are good but not always great. Part of the equation is measuring 'random' events, manipulating that data and then making it available (via (u)random). These random events include: keyboard and mouse input, interrupts, drive reads, etc.

However, as many servers have no keyboard/mouse, and new "blackbox" products often contain no harddrive, sources of random data become harder to find. Some sources, like network activity, are not entirely appropriate because the attacks may be able to measure it as well (granted this would be a very exotic attack, but enough to worry people nonetheless). There are several sources of random data that can be used (or at least they appear random), radioactive decay and radio frequency manipulations are two popular ones. Unfortunately the idea of sticking a radioactive device in a computer makes most people nervous. And using manipulated radio frequencies is prone to error, and the possibility of outside manipulation. For most of us, this isn't a real concern, however for IPsec gateway servers handling many connections it can be a problem. One potential solution is the PIII, which has a built in random number generator that measures thermal variance in the CPU, I think as we progress, solutions like this will become more common.

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Limiting and monitoring users

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Many security problems arise from the simple fact that you must allow user access to your systems. In some cases users can be malicious (i.e. internal attacker, university students, etc.) or they may accidentally expose their password to an attacker or simply use a weak password (like their username). In any event it is all too possible for users to attack a system, so it is advisable to monitor users, and limit the amount of damage they can do.

## Limiting users

Users require resources like CPU time, memory and drive space to do their work. On many systems it is possible for users to hog resources, reducing the usefulness of the system to other users or in some cases actually bringing the server to a slow halt or crashing it. Users can also inadvertently use up more resources than they mean to, limiting their resources can prevent several classes of problems. Programs can crash, generating huge core dumps, or go crazy and user up all the available memory. Something to remember: global limits apply to root, so be careful! If you do not allow root to run enough processes for example cron jobs may fail or you may not even be able to log in to fix any problems.

## PAM

Almost all Linux distributions ship with PAM support making it universally available. PAM limits provide a single standardized interface to setting user limits, instead of having to write complex shell configuration files (such as `/etc/profile`) you simply edit the "limits.conf" file. As well applying limits selectively through the command shell is very difficult, whereas with PAM applying limits globally, on groups or on individual users is quite simple. Documentation is available on PAM usually in the `/usr/share/doc/` tree. To enable PAM limits you need to add a line such as:

```
session      required      /lib/security/pam_limits.so
```

to the appropriate Pam configuration file (i.e. `/etc/pam.d/sshd`). You can then define limits, typically these are in `/etc/security/limits.conf` or a similar location. Because most of these limits are enforced by the shell the system cannot log all violations of limits (i.e. you will be notified in

syslog when a user exceeds the number of times they are allowed to login, however you will not receive a warning if the user tries to use more disk space than they are allowed to).

The available limits are:

**core** -- Limits the core file size (KB); usually set to 0 for most users to prevent core dumps.

**data** -- Maximum data size (KB).

**fsize** -- Maximum file size (KB).

**memlock** -- Maximum locked-in-memory address space (KB).

**nofile** -- Maximum number of open files.

**rss** -- Maximum resident set size (KB).

**stack** -- Maximum stack size (KB).

**cpu** -- Maximum CPU time (MIN).

**nproc** -- Maximum number of processes.

**as** -- Address space limit.

**maxlogins** -- Maximum number of logins for this user or group.

**priority** -- The priority to run user process with.

For example you can limit the amount of memory that user "bob" is allowed to use:

```
user          hard    memlock    4096
```

This would place a hard (absolute) limit of 4 megabytes on memory usage for "bob". Limits can be placed on users by listing the user name, groups by using the syntax "@group" or globally by using "\*".

core files can be created when a program crashes. They have been used in security exploits, overwriting system files, or by containing sensitive information (such as passwords). You can easily disable core dumps using PAM, and generally speaking most users will not notice, however if you have software developers they may complain.

```
*            hard    core        0
```

fsize is generally a good idea to set, many users will have a large filesystem quota (i.e. tens of megabytes to several hundred or several gigabytes), however if they are allowed to create a single file that is abnormally large they can easily hog disk I/O resources (i.e. create a large file and copy/delete the copy repeatedly). Setting this limit globally can also prevent an attacker from trying to fill up the partitions your log files are stored on, for example if you only have a single / partition an attacker can easily fill it up by generating a lot of log events.

```
@notroot     hard    data        102400
```

Of course limiting CPU time is one of the classic administrative tasks, this is very useful for preventing run-away processes from eating up all the cpu time, and it ensures that if a user leaves something running in background (such as a packet sniffer) it will eventually be killed. Limiting CPU time

will have several side effects however, once of which will be limiting the amount of time a user can spend logged in (eventually they will run out of CPU time and the session will be killed), this can lead to problems if users spend long periods logged in. As well depending on the CPU(s) present in your machine the limits can vary greatly (one minute on a 386 is quite a bit different then one minute on a 1.3 GHz Athalon).

```
@students      hard      cpu          2
```

Limiting the number of times a user can login is strongly advised, for most situations users should not need to log in to a server more then once, and allowing them to do so let' \$them use more resources then you might intend. As well it can be used to detect suspicious activity, if users know they can only login once then attempts to log in multiple times can be viewed as suspicious activity (i.e. an attacker with a stolen password trying to access the account).

```
@users         hard      maxlogins    1
```

Additionally when someone violated this limit it will be logged in syslog:

```
Apr 15 15:09:26 stench PAM_unix[9993]: (sshd) session opened for user test by (uid=0)
Apr 15 15:09:32 stench pam_limits[10015]: Too many logins (max 1) for test
```

Soft limit violations will not be logged (i.e. a soft limit of 1 and a hard limit of 2).

## Bash

Bash has built in limits, accessed via "ulimit". Any hard limits cannot be set higher, so if you have limits defined in /etc/profile, or in the users .bash\_profile (assuming they cannot edit/delete .bash\_profile) you can enforce limits on users with Bash shells. This is useful for older Linux distributions that lack PAM support (however this is increasingly rare and PAM should be used if possible). You must also ensure that the user cannot change their login shell, if they use "chsh" to change their shell to ksh for example the next time they login they will have no limits (assuming you cave not put limits on ksh). Documentation is available on ulimit, log in using bash and issue:

```
[root@server /root]# help ulimit
ulimit: ulimit [-SHacdflmpstuv] [limit]
    Ulimit provides control over the resources available to processes
    started by the shell, on systems that allow such control.  If an
    option is given, it is interpreted as follows:

    -S      use the `soft' resource limit
    -H      use the `hard' resource limit
    -a      all current limits are reported
    -c      the maximum size of core files created
    -d      the maximum size of a process's data segment
    -f      the maximum size of files created by the shell
    -l      the maximum size a process may lock into memory
```

```

-m      the maximum resident set size
-n      the maximum number of open file descriptors
-p      the pipe buffer size
-s      the maximum stack size
-t      the maximum amount of cpu time in seconds
-u      the maximum number of user processes
-v      the size of virtual memory

```

If LIMIT is given, it is the new value of the specified resource. Otherwise, the current value of the specified resource is printed. If no option is given, then -f is assumed. Values are in 1024-byte increments, except for -t, which is in seconds, -p, which is in increments of 512 bytes, and -u, which is an unscaled number of processes.

To disallow core files (by setting the maximum size to 0) for example you would add:

```
ulimit -Hc 0
```

To set limits globally you would need to edit "/etc/profile", of course this will also affect root, so be careful! To set limits on groups you would need to add scripting to "/etc/profile" that would check for the user's membership in a group and then apply the statements, however doing this with PAM is recommended as it is a lot simpler.

## Quota

Quota allows you to enforce user and group limits on disk usage. Quotas are most often used to prevent users from hogging disk space that is shared with other users (i.e. "/home"). There are several benefits to using quota instead of PAM/shell limitations; the first being that you can apply it selectively to disk systems, i.e. you can place a limit on /home/ but not on /tmp/. Additionally you can place limits on the number of inodes used as well as space, running out of inodes is as bad as running out of space since you won't be able to create new files. To see how much space a disk has (in kilobytes):

```

[root@server /]# df -k
Filesystem      lk-blocks      Used Available Use% Mounted on
/dev/hda2       2925900        948936  1828336   35% /
/dev/hda1       49743          2485    44690    6% /boot

```

And to show inode usage:

```

[root@dildo /]# df -i
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/hda2       371680   58737   312943  16% /

```

```
/dev/hda1          12880      27   12853    1% /boot
```

Quota support must be compiled into the kernel (this is the default with most vendors), and the tools must be installed, usually in a package called "quota". Once you have done this you need to edit your "/etc/fstab" file and add either the keyword "grpquota" or "usrquota" in the options:

```
LABEL=/           /           ext2    defaults          1 1
LABEL=/home       /home       ext2    defaults,grpquota 1 2
```

Then for each filesystem that you have enabled quotas on you must create two files in the root of the filesystem:

```
[root@dildo /]# touch /home/quota.user
[root@dildo /]# touch /home/quota.group
```

These files should only be accessible to root. You will then need to reboot the system.

Typically you will only need to place quota on filesystems directly writeable to the user, usually this is:

```
/tmp
/var/tmp
/home
```

To edit user and group quotas you use the program "edquota", for example:

```
[root@dildo /]# edquota -u username
```

This will then let you edit the appropriate quota.user or quota.group file (these files should not be edited directly!). The editor uses vi by default and the data looks like:

```
Quotas for user username:
/dev/hda1: blocks in use: 255, limits (soft = 5120, hard = 10240)
          inodes in use: 123, limits (soft = 1000, hard = 1500)
```

Soft limits will generate warnings and hard limits will stop the user from using more disk space or inodes. Documentation on this can be found in "man quota" or in the Quota mini-howto available at: <http://www.linux.com/howto/mini/Quota.html>.

## Monitoring users

Monitoring users may be a requirement of your security policy, or you may have discovered an account that has been compromised and you wish to monitor the attacker when they use it. Unfortunately while this used to be relatively easy there are now a number of new issues that make monitoring users problematic. The increased usage of encrypted protocols like SSH and SSL mean you can no longer simply run a packet sniffer and monitor the attacker. The change in Linux' s tty' to pseudo terminals has "broken" many older software packages as well. Typically the best way to monitor users

is transparently, packet sniffers are ideal for this as you need not make any changes on the server at all. The second best is to use some form of monitoring on the server, this used to be possible with ttysnoop however ttysnoop does not appear to be actively maintained anymore. While you can use tricks like shell logging this will be obvious to a skilled attacker as the first thing they usually do is disable shell logging. If you are monitoring legitimate users you should inform them (this is a legal requirement in some countries and states). If you are monitoring an attacker and intend to use the log files as evidence you should probably consult a lawyer to make sure it is done correctly.

## sshsniff

This is actually a really useful tool I stumbled across. It allows you to monitor all traffic going in and coming out of a process, such as an interactive shell. The bad news is that it does not appear to be completely reliable. While testing it sshsniff reliably captured keystrokes sent to the shell and the replies, however when files were cat'ed, i.e. "cat /etc/passwd" the file was not always displayed on the monitoring shell:

```
[test@server test]$ ???\033???[???Acat /etc/passwd???  
--- SIGCHLD (Child exited) ---  
[test@server test]$ ???\033???[???Acat /etc/passwd???  
--- SIGCHLD (Child exited) ---  
[test@server test]$ ???\033???[???Acat /etc/passwd???
```

```
[pid 1417]  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:  
daemon:x:2:2:daemon:/sbin:  
adm:x:3:4:adm:/var/adm:  
lp:x:4:7:lp:/var/spool/lpd:  
sync:x:5:0:sync:/sbin:/bin/sync  
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown  
halt:x:7:0:halt:/sbin:/sbin/halt  
mail:x:8:12:mail:/var/spool/mail:  
operator:x:11:0:operator:/root:  
games:x:12:100:games:/usr/games:  
ftp:x:14:50:FTP User:/home/ftp:/bin/true  
nobody:x:99:99:Nobody:/:  
rpc:x:32:32:Portmapper RPC user:/:/bin/false  
postfix:x:89:89:/:/var/spool/postfix:/bin/true  
test:x:500:500:/:/home/test:/bin/bash  
+++ exited (status 0) +++
```

```
[pid 1380]  
--- SIGCHLD (Child exited) ---
```

sshsniff is not perfect but it is able to bind to any PID and not just terminals, making it very flexible. You can download it from:

<http://www.psychoid.lam3rz.de/exploits.html>. One note there is a file called col.c that contains an entry for /etc/passwd, it does not appear to be used but it does raise some questions. If you plan to use this code I strongly suggest you audit it.

## **dsniff**

With protocols like SSH and SSL being widely deployed simple packet sniffers are no longer as useful as they once were. So people started writing more complex packet sniffers, like dsniff. dsniff is capable of doing man in the middle attacks against SSL and SSH. If you control the server then you can easily proxy connections (as you have access to the secret keys on the server) and thus decrypt the attackers SSH session and monitor what is going on. dsniff is not capable of dealing with all version of the SSH protocol however, so forcing only protocol 1 support on the server can solve this, in your sshd\_config file:

```
Protocol 1
```

is specified and not:

```
Protocol 2,1
```

dsniff is available from: <http://www.monkey.org/~dugsong/dsniff/>

## **Firewalling**

One new capability in Linux 2.4 that should prove quite useful is the ability to filter some types of outgoing packet based on who owns the process that created them. Of course this is not 100% effective as some types of packets are not owned by anyone specifically (i.e. ICMP responses). Using the various flags:

```
--uid-owner  
--gid-owner  
--pid-owner  
--sid-owner
```

which are discussed in more detail in the firewall section, documentation is available via "man iptables".

---

[Back](#)

Last updated on 1/9/2001

## Viruses

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Linux is not as susceptible to viruses in the same ways that a Dos/Windows or Mac platform is. In UNIX, security controls are a fundamental part of the operating system. For example users are not allowed to write promiscuously to any location in memory that they choose to, something that Dos/Windows and the Mac allow.

To be fair there are viruses for UNIX. However the only Linux one I have seen was called "bliss", had an uninstall option ("--uninstall-please") and had to be run as root to be effective. Or to quote an old Unix favorite "if you don' t know what a executable does, don' tun it as root". Worms are much more prevalent in the UNIX world, the first major occurrence being the Morris Internet worm which exploited a vulnerability in sendmail. Current Linux worms exploit broken versions of imapd, sendmail, WU-FTPD and other daemons. The simplest fix is to keep up to date and not make daemons accessible unless necessary. These attacks can be very successful especially if they find a network of hosts that are not up to date, but typically their effectiveness fades out as people upgrade their daemons. In general I would not specifically worry about these two items, and there is definitely no need to buy anti-virus software for Linux.

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

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

## Disinfection of viruses / worms / trojans

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

The easiest method for dealing with viruses and the like is to use system integrity tools such as tripwire, L5, and Gog&Magog, you will be able to

easily find which files have been compromised and restore/replace/update them. There are also many Anti-Virus scanners available for Linux (but generally speaking there aren't any Linux viruses).

## **Virus Scanners for Linux**

As stated above viruses aren't a real concern in the Linux world, however virus scanners that run on Linux can be useful. Filtering email / other forms of content at the gateways to your network (everyone has Windows machines) can provide an extra line of defense since the platforms providing the defense against the threat cannot be compromised by that threat (hopefully). You may also wish to scan files stored on Linux file servers that are accessed by Windows clients. Luckily there are several good anti-virus programs available for Linux.

### **Sophos Anti-Virus**

Sophos Anti-Virus is a commercial virus scanner that runs on a variety of Windows and UNIX platforms. It is free for personal use and relatively inexpensive for commercial use. You can get it at: <http://www.sophos.com/>.

### **AntiVir**

AntiVir is another commercial virus scanner that runs on a variety of Windows platforms and Linux. You can get it from: <http://www.hbedv.com/>.

### **InterScan VirusWall**

Trend Micro has ported this product to Linux and offers it for free download on their site. You can get it from: <http://www.anti-virus.com/products/isvw/>.

### **F-Secure Anti-Virus**

Data Fellow' s has ported their anti-virus scanner to Linux as well. You can get it at: <http://www.datafellows.com/products/>

## AVP

Kaspersky lab's has also ported their anti-virus scanner over to Linux, currently in beta, available at: <http://www.kaspersky.com/products.asp>

## Virus scanning of email

Also see the [email server section](#) for setting up [virus scanning of incoming email](#) (very useful if you have windows clients).

[www.openantivirus.org](http://www.openantivirus.org)

---

### Back

Last updated on 1/9/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Virtual private networks

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

There are a variety of VPN solutions for Linux. I would strongly advise using IPSec if possible since it is the emerging standard for VPN' on the Internet, and will be incorporated with IPv6. On the other hand if you are behind a firewall and want to tunnel out the SSH based solution and so on will do the trick, whereas IPSec will typically fail (since the packet headers are being rewritten).

## **IP Security (IPSec)**

If possible use IPSec. It is available on almost all modern operating systems, supports host to host, host to network and network to network configurations as well as a wide variety of authentication options.

## **PPTP (Point to Point Tunneling Protocol)**

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

## **CIPE (Crypto IP Encapsulation)**

CIPE is a free IP level encryption scheme, meant for use between routers. It is appropriate for 'bridging' networks securely together over insecure networks (like the Internet). The official cite for CIPE is at: <http://sites.inka.de/~W1011/devel/cipe.html>. I would however recommend FreeS/WAN as a better long term solution. CIPE is very easy to setup for two servers but anything more then two servers becomes a configuration nightmare.

## **Zebedee**

Zebedee provides encryption of TCP traffic between hosts and is available for UNIX and windows. You can get it from: <http://www.winton.org.uk/zebedee/>.

---

[Back](#)

Last updated on 4/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

# The Linux kernel

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

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

In addition to this we have hardware problems like the Pentium F00F bug, and problems inherent to the TCP-IP protocol, the Linux kernel has it's work cut out for it. Kernel versions are labeled as X.Y.Z, Z are minor revision numbers, Y define whether the kernel is a test (odd number) or production (even number), and X defines the major revision (we have had 0, 1 and 2 so far). I would highly recommend running kernel 2.2.x, as of December 1999 this is 2.2.13. The 2.2.x series of kernel has major improvements over the 2.0.x series. Using the 2.2.x kernels also allows you access to newer features such as ipchains (instead of ipfwadm) and other advanced security features. The 2.0.x series has also been officially discontinued as of June 1999. To find out what the latest kernel(s) are simply finger @linux.kernel.org:

```
[seifried@mail kernel-patches]$ finger @finger.kernel.org
[zeus.kernel.org]
```

```
The latest stable version of the Linux kernel is:      2.4.9
The latest prepatch (alpha) version *appears* to be:  2.4.10-pre4
```

## Compiling and installing a kernel:

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

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

```
cd /usr/src
```

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

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

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

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

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

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux
image=/boot/vmlinuz-2.4.9
    label=linux
    root=/dev/hda1
    read-only
image=/boot/vmlinuz-2.4.5
    label=linuxold
    root=/dev/hda1
    read-only
```

Once you have finished editing /etc/lilo.conf you must run /sbin/lilo to rewrite the MBR (Master Boot Record). When LILO runs you will see output similar to:

```
Added linux *
Added linuxold
```

It will list the images that are listed on the data in the MBR and indicate with a \* which is the default (typically the default to load is the first image listed, unless you explicitly specify one using the default directive).

## **Kernel versions**

Currently the stable kernel release series is 2.4.x, and the development series is 2.5.x. The 2.3.x development series of kernels is not recommended, there are many problems and inconsistencies. The 2.2.x series of kernel while old and lacking some features is relatively solid, unfortunately the upgrade from 2.2.x to 2.4.x is a pretty large step, I would advise caution. Several software packages must be updated, libraries, ppp, modutils and others (they are covered in the kernel docs / rpm dependencies / etc.). Additionally keep the old working kernel, add an entry in lilo.conf for it as "linuxold" or something similar and you will be able to easily recover in the event 2.2.x doesn't work out as expected. Don't expect the 2.4.x series to be bug free, flaws will be found and older versions will become obsolete, like every piece of software in the world.

There are a variety of kernel level patches that can enhance the security of a Linux system. Some prevent buffer overflow exploits, other provide strong crypto.

## **Kernel patches**

There are a variety of kernel patches directly related to security.

### **Secure Linux kernel patch**

This patch solves a number of issues and provides another level of security for the system. The patch is available for the 2.0 and 2.2 kernel series. You can get it from: <http://www.openwall.com/linux/>.

### **International kernel patch**

This patch (over a megabyte in size!) adds a huge amount of strong crypto and related items. It includes several encryption algorithms that were AES candidates (including MARS from IBM). You can get it from: <http://www.kerneli.org/>. <http://sourceforge.net/projects/cryptoapi/>

## **Linux Intrusion Detection System Patch (LIDS)**

This patch adds a number of interesting capabilities, primarily aimed at attack detection. You can "lock" file mounts, firewall rules, and a variety of other interesting options are available. You can get it from: <http://www.lids.org/>

## **RSBAC**

Rule Set Based Access Control is a comprehensive set of patches and utilities to control various aspects of the system, from filesystem ACL's and up. You can get it from: <http://www.rsbac.de/rsbac/>.

## **LOMAC**

LOMAC (Low Water-Mark Mandatory Access Control for Linux) is a set of kernel patches to enhance Linux security. You can get it at: <ftp://ftp.tislabs.com/pub/lomac/>.

## **auditd**

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

## **Fork Bomb Defuser**

A loadable kernel module that allows you to control the maximum number of processes per user, and the maximum number of forks, very useful for shell servers with untrusted users. You can get it from: <http://rexgrep.tripod.com/rexfbdmain.htm>.

## **Debugging the Linux kernel**

## **KDB v0.6 (Built-in Kernel Debugger)**

An SGI kernel debugger, available at: <http://oss.sgi.com/projects/kdb/>.

## **kGDB (Remote kernel debugger)**

SGI has written a tool that allows you to do kernel debugging, remotely which is a big step up from being tied to the console. You can get it at: <http://oss.sgi.com/projects/kgdb/>.

---

[Back](#)

Last updated on 4/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## **Security techniques**

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

Chrooting

<http://www.bpfh.net/simes/computing/chroot-break.html>

---

[Back](#)

Last updated 31/8/2001

Copyright Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org) 2001

# Checklists

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

## Internet connection checklist

- Turn off all unnecessary services
  - Use firewalling to block access to services if possible
  - Use TCP\_WRAPPERS to restrict access to services
  - Run nmap and nessus against the host minimally
  - SSL wrap services such as POP and IMAP
  - Use SSH instead of Telnet
  - Ensure software is up to date
- 

[Back](#)

Last updated on 4/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

Distribution information

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

## [Vendor contact information](#)

[Caldera](#)

Debian

[NetMAX](#)

[Red Hat](#)

Slackware

Stormix

[SuSE](#)

[TurboLinux](#)

---

[Back](#)

Last updated on 31/8/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)

## Appendices

By Kurt Seifried [kurt@seifried.org](mailto:kurt@seifried.org)

---

There are of course numerous resources online. Here is a short list of some of the better ones.

## Appendix A: Books and magazines

Sendmail - <http://www.oreilly.com/catalog/sendmail2/>

Linux Network Admin Guide (NAG) - <http://www.oreilly.com/catalog/linag/>

Running Linux - <http://www.oreilly.com/catalog/runux2/noframes.html>

DNS & BIND - <http://www.oreilly.com/catalog/dns3/>

Apache - <http://www.oreilly.com/catalog/apache2/>

Learning The Bash Shell - <http://www.oreilly.com/catalog/bash2/>

Building Internet Firewalls - <http://www.oreilly.com/catalog/fire/>

Computer Crime - <http://www.oreilly.com/catalog/crime/>

Computer Security Basics - <http://www.oreilly.com/catalog/csb/>

Cracking DES - <http://www.oreilly.com/catalog/crackdes/>

Essential System Administration - <http://www.oreilly.com/catalog/esa2/>

Linux in a nutshell - <http://www.oreilly.com/catalog/linuxnut2/>  
Managing NFS and NIS - <http://www.oreilly.com/catalog/nfs/>  
Managing Usenet - <http://www.oreilly.com/catalog/musenet/>  
PGP - <http://www.oreilly.com/catalog/pgp/>  
Practical Unix and Internet Security - <http://www.oreilly.com/catalog/puis/>  
Running Linux - <http://www.oreilly.com/catalog/runux2/>  
Using and Managing PPP - <http://www.oreilly.com/catalog/umppp/>  
Virtual Private Networks - <http://www.oreilly.com/catalog/vpn2/>

Red Hat/SAMS also publish several interesting books:

Maximum RPM (available as a postscript document on <http://www.rpm.org/>)

Red Hat User' s Guide(available as HTML on <ftp://ftp.redhat.com/>)

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

Magazines:

Linux Journal (of course, monthly)

Sys Admin (intelligent articles, monthly)

Perl Journal (quarterly)

Information Security - <http://www.infosecuritymag.com/>

## **Appendix C: Other Linux security documentation**

Firewalling and Proxy Server HOWTO

<http://metalab.unc.edu/LDP/HOWTO/Firewall-HOWTO.html>

Linux IPCHAINS HOWTO

<http://www.ibiblio.org/LDP/HOWTO/IPCHAINS-HOWTO.html>

Linux Security HOWTO

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

Linux Shadow Password HOWTO

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

The Linux CIPE + Masquerading mini-HOWTO

<http://www.ibiblio.org/LDP/HOWTO/mini/Cipe+Masq.html>

Firewall Piercing mini-HOWTO

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

Quota mini-HOWTO

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

Secure POP via SSH mini-HOWTO

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

The VPN HOWTO (using SSH)

<http://metalab.unc.edu/LDP/HOWTO/mini/VPN.html>

Red Hat Knowledge Base

<http://www.redhat.com/cgi-bin/support?faq>

## **Appendix D: Online security documentation**

Bugtraq Archives

<http://www.geek-girl.com/bugtraq/>

CERT Incident Reporting Guidelines

[http://www.cert.org/tech\\_tips/incident\\_reporting.html](http://www.cert.org/tech_tips/incident_reporting.html)

Site Security Handbook

<http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/21xx/2196>

Guidelines for the Secure Operation of the Internet

<http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/12xx/1281>

How to Handle and Identify Network Probes

<http://www.network-defense.com/papers/probes.html>

Free Firewall and related tools (large)

[http://sites.inka.de/sites/lina/freefire-l/index\\_en.html](http://sites.inka.de/sites/lina/freefire-l/index_en.html)

Internet FAQ Consortium (You want FAQ's? We got FAQ's!)

<http://www.faqs.org/>

An Architectural Overview of UNIX Network Security

<http://www.alw.nih.gov/Security/Docs/network-security.html>

The human side of computer security (an article on social engineering)

<http://www.sunworld.com/sunworldonline/swol-07-1999/swol-07-security.html>

IBM Redbooks

<http://www.redbooks.ibm.com/>

General security research and development

<http://www.sekure.net/>

Some general whitepapers and articles

<http://www.enteract.com/~lspitz/pubs.html>

COAST Library

<http://www.cerias.purdue.edu/coast/coast-library.html>

Coast hotlist (hugelist of resources)

<http://www.cerias.purdue.edu/coast/hotlist/>

D.O.E. Sysworks

<http://members.aol.com/jpeschel/index.htm>

## **Appendix E: General security sites**

SANS

<http://www.sans.org/>

Computer Security Information

<http://www.alw.nih.gov/Security/security.html>

8 Little Green Men

<http://www.8lgm.org/>

Robert' Cryptography, PGP & Privacy Links

<http://www.interlog.com/~rguerra/www/>

Cryptome

<http://cryptome.org>

PacketStorm

<http://www.packetstormsecurity.net/>

COAST

<ftp://coast.cs.purdue.edu/pub/>

.rain.forest.puppy

<http://www.wiretrip.net/rfp/>

InfoWar

<http://www.infowar.com/>

## Appendix F: General Linux sites

Linux.com

<http://www.linux.com/>

Linux.org

<http://www.linux.org/>

Linux Administration Made Easy (LAME)

<http://www.LinuxNinja.com/linux-admin/>

---

[Back](#)

Last updated on 4/10/2001

Copyright Kurt Seifried 2001 [kurt@seifried.org](mailto:kurt@seifried.org)