Gentoo Security Handbook

Kim Nielsen *Author*
John P. Davis *Editor*
Eric R. Stockbridge *Editor*
Carl Anderson *Editor*
Jorge Paulo *Editor*
Sven Vermeulen *Editor*
Benny Chuang *Editor*
Sune Jeppesen *Editor*
Tiemo Kieft *Editor*
Zack Gilburd *Editor*
Dan Margolis *Editor*

Updated February 20, 2007

Content:

A. System Security

1. Pre-Installation Concerns

1.a. Physical Security

No matter how many safeguards you implement, they can all be easily circumvented by an attacker with physical access to your computer. Despite this, there are at least some measures that can be taken to provide a degree of security against an attacker with physical access to your machine. Putting your hardware in a locked closet prevents an attacker from simply unplugging it and carting it off. Locking your computer's case is also a good idea, to make sure that an attacker cannot simply walk away with your hard drive. To prevent an attacker from booting from another disk, nicely

circumventing your permissions and login restrictions, try setting the hard drive as the first boot device in your BIOS, and setting a BIOS password. It is also important to set a LILO or GRUB boot password, to prevent a malicious user from booting into single-user mode and gaining complete access to your system. This is covered in more detail in Chapter 3, under Setting a GRUB password and Setting a LILO password.

1.b. Daemon/Service Planning

Start by documenting what services this machine should run. This will help you compose a better partition scheme for your system, and allow you to better plan your security measures. Of course, this is unnecessary if the machine serves a single simple purpose, such as a desktop, or a dedicated firewall. In those cases, you should not be running *any* services, except perhaps sshd.

This list can also be used to aid system administration. By keeping a current list of version information, you will find it much easier to keep everything up to date if a remote vulnerability is discovered in one of your daemons.

1.c. Partitioning Schemes

Partitioning rules:

- Any directory tree a user should be able to write to (e.g. `/home`, `/tmp`) should be on a separate partition and use disk quotas. This reduces the risk of a user filling up your whole filesystem. Portage uses `/var/tmp` to compile files, so that partition should be large.
- Any directory tree where you plan to install non-distribution software on should be on a separate partition. According to the File Hierarchy Standard, this is `/opt` or `/usr/local`. If these are separate partitions, they will not be erased if you have to reinstall the system.
- For extra security, static data can be put on a separate partition that is mounted read-only. For the truly paranoid, try using read-only media like CD-ROM.

1.d. The root user

The user 'root' is the most vital user on the system and should not be used for anything except when absolutely necessary. If an attacker gains root access, the only way to ever trust your system again is to reinstall.

Golden rules about 'root'

- Always create a user for everyday use and if this user needs to have root access, add the user to the group 'wheel'. This makes it possible for a normal user to `su` to root.
- Never run X or any other user application as root. root should only be used when absolutely necessary; if a vulnerability exists in an application running as a user, an attacker can gain user level access. But if that application is running as root, the attacker gains root access.
- Always use absolute paths when logged in as root (or always use `su -`, which replaces the environmental variables of the user with those of root, while being sure root's `PATH` only includes protected directories like `/bin` and `/sbin`). It's possible to trick root into running a different application rather than the one meant to be run. If root's `PATH` is protected or root only uses absolute paths, we can be sure this won't happen.
- If a user only needs to run a few commands as root, instead of everything that root normally can do, consider using `sudo` instead. Just be careful who you give this access to, as well!
- Never leave the terminal when you are logged in as root.

Gentoo has some default protection against normal users trying to `su` to root. The default PAM setting requires that a user be a member of the group "wheel" in order to be able to `su`.

1.e. Security policies

There are several reasons to draft a security policy for your system(s) and network.

- A good security policy allows you to outline security as a "system", rather than simply a jumble of different features. For example, without a policy an administrator might decide to turn off telnet, because it transmits unencrypted passwords, but leave on FTP access, which has the same weakness. A good security policy allows you to identify which security measures are worthwhile, and which are not.
- In order to diagnose problems, conduct audits, or track down intruders, it may be necessary to intercept network traffic, inspect the login and command history of users, and look in home directories. Without outlining this in print, and making users aware of this, such actions may actually be illegal and put *you* in legal jeopardy.
- Hijacked user accounts pose one of the most common threats to system security. Without explaining to users why security is important, and how to practice good security (such as not writing passwords on a Post-It note on their

desks), it is unlikely you will have any hope of secure user accounts.
- A well-documented network and system layout will aid you, as well as law enforcement forensics examiners, if need be, in tracing an intrusion and identifying weaknesses after the fact. A security policy "issue" banner, stating that your system is a private network and all unauthorized access is prohibited, will also help ensure your ability to properly prosecute an intruder, once he is caught.

The need for a good security policy is hopefully now more than clear.

The policy itself is a document, or several documents, that outlines the network and system features (such as what services are provided), acceptable use and forbidden use, security "best practices", and so forth. All users should be made aware of your security policy, as well as changes you make to keep it up to date. It is important that you take the time to help users understand your policy and why that policy needs to be signed or what will happens if they act directly against the policy (the policy should also state this). This should be repeated at least once a year, since the policy can change (but also as a reminder to the user of the policy itself).

Note: Create policies that are easy to read and be very precise on every subject.

A security policy should at least contain the following subjects:

- Acceptable use
  - Screen savers
  - Password handling
  - Software download and installation
  - Information stating if the users are being monitored
  - Use of anti-virus software
- Handling of sensitive information (any written form, paper or digital)
  - Clean desk and locked up classified information
  - PC shutdown before leaving
  - Use of encryption
  - Handling of keys to trusted co-workers
  - Handling of confidential material when traveling
- Handling of computer equipment when traveling
  - Laptop handling during travels and hotel stays

Different users may require different levels or types of access, and as such your policy may vary to accommodate them all.

The security policy can become huge, and vital information can easily be forgotten. The IT-staff's policy could contain information that is confidential for the ordinary user, so it is wise to split it up into smaller policies; e.g. Acceptable Use Policy, Password policy, Email policy and Remote Access policy.

You can find example policies at The SANS Security Policy Project. If you have a small network and think these policies are too much you should look at the Site Security Handbook.

2. Tightening Security

2.a. USE flags

The `make.conf` file contains user defined USE flags and `/etc/make.profile/make.defaults` contains the default USE flags for Gentoo Linux. For this guide's purposes, the important flags are `pam` (Pluggable Authentication Modules), `tcpd` (TCP wrappers), and `ssl` (Secure Socket Layer). These are all in the default USE flags.

2.b. Password protecting GRUB

GRUB supports two different ways of adding password protection to your boot loader. The first uses plain text, while the latter uses md5+salt encryption.

Code Listing 1: /boot/grub/grub.conf

```
timeout 5
password changeme
```

This will add the password `changeme`. If no password is entered at boot, GRUB will simply use the default boot setting.

When adding an md5 password, you must convert your password into crypt format, which is the same format used in `/etc/shadow`. For more information see `man crypt`. The encrypted password *changeme*, for example, could look like this: `$1$T7/dgdIJ$dJM.n2wZ8RG.oEiIOwJUs`.

You can encrypt your password directly at the GRUB shell:

Code Listing 2: md5crypt in grub shell

```
#/sbin/grub

GRUB version 0.92 (640K lower / 3072K upper memory)

   [ Minimal BASH-like line editing is supported. For the first word, TAB lists
     possible command completions. Anywhere else TAB lists the possible
     completions of a device/filename. ]

grub> md5crypt

Password: ********
(Typed changeme at the prompt)
Encrypted: $1$T7/dgdIJ$dJM.n2wZ8RG.oEiIOwJUs.

grub> quit
```

Then, cut and paste your password to `/boot/grub/grub.conf`.

Code Listing 3: /boot/grub/grub.conf

```
timeout 5
password --md5 $1$T7/dgdIJ$dJM.n2wZ8RG.oEiIOwJUs.
```

The 5 seconds timeout becomes handy if the system is remote and should be able to reboot without any keyboard interaction. Learn more about GRUB passwords by executing `info grub`.

2.c. Password protecting LILO

LILO also supports two ways of handling passwords: global and per-image, both in clear text.

The global password is set at the top of the configuration file, and applies to every boot image:

Code Listing 4: /etc/lilo.conf

```
password=changeme
restricted
delay=3
```

The per-image password is set as below:

Code Listing 5: /etc/lilo.conf

```
image=/boot/bzImage
      read-only
      password=changeme
      restricted
```

If the `restricted` option is not entered, it will prompt for a password every time.

In order to store the new information in `lilo.conf`, you must run `/sbin/lilo`.

2.d. Restricting Console Usage

The `/etc/securetty` file allows you to specify which `tty` (terminal) devices root is allowed to login in from.

We suggest that you comment out all lines except `vc/1` if you are using devfs and all lines except `tty1` if you are using udev. This will ensure that root only can login once and only on one terminal.

Note: Users in the group "wheel" can still `su -` to become root on other TTYs.

Code Listing 6: /etc/securetty

```
(For devfs)
vc/1
(For udev)
tty1
```

3. Logging

3.a. Introduction

Extra logging should be added to catch warnings or errors that might indicate an ongoing attack or a successful compromise. Attackers often scan or probe before attacking.

It's also vital that your log files are easily readable and manageable. Gentoo Linux lets you choose between 3 different loggers when installing.

3.b. Logging: Syslogd

Syslogd is the most common logger for Linux and Unix in general. It has some log rotation facilities, but using `/usr/sbin/logrotate` in a cron job (logrotate is configured in `/etc/logrotate.conf`) might prove to be more powerful as `logrotate` has many features. How often log rotation should be done depends on the system load.

Below is the standard `syslog.conf` with some added features. We have uncommented the `cron` and `tty` lines and added a remote logging server. To further enhance security you could add logging to two places.

Code Listing 1: /etc/syslog.conf

```
#  /etc/syslog.conf      Configuration file for syslogd.
#
#                       For more information see syslog.conf(5)
#                       manpage.
#                       This is from Debian, we are using it for now
#                       Daniel Robbins, 5/15/99

#
# First some standard logfiles.  Log by facility.
#

auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          /var/log/mail.log
user.*                          -/var/log/user.log
uucp.*                          -/var/log/uucp.log
local6.debug                    /var/log/imapd.log

#
# Logging for the mail system. Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err

# Logging for INN news system
#
news.crit                       /var/log/news/news.crit
news.err                        /var/log/news/news.err
news.notice                     -/var/log/news/news.notice

#
# Some `catch-all' logfiles.
#
*.=debug;\
        auth,authpriv.none;\
        news.none;mail.none     -/var/log/debug
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          -/var/log/messages

#
# Emergencies and alerts are sent to everybody logged in.
#
*.emerg                         *
*.=alert                        *

#
# I like to have messages displayed on the console, but only on a virtual
# console I usually leave idle.
#
daemon,mail.*;\
       news.=crit;news.=err;news.=notice;\
       *.=debug;*.=info;\
       *.=notice;*.=warn       /dev/tty8

#Setup a remote logging server
*.*                        @logserver

# The named pipe /dev/xconsole is for the `xconsole' utility.  To use it,
# you must invoke `xconsole' with the `-file' option:
#
#    $ xconsole -file /dev/xconsole [...]
#
# NOTE: adjust the list below, or you'll go crazy if you have a reasonably
#      busy site..
#
#daemon.*,mail.*;\
#       news.crit;news.err;news.notice;\
#       *.=debug;*.=info;\
#       *.=notice;*.=warn       |/dev/xconsole

local2.*                --/var/log/ppp.log
```

Attackers will most likely try to erase their tracks by editing or deleting log files. You can make it harder for them by

logging to one or more remote logging servers on other machines. Get more info about syslogd by executing `man syslog`.

3.c. Metalog

Metalog by Frank Dennis is not able to log to a remote server, but it does have advantages when it comes to performance and logging flexibility. It can log by program name, urgency, facility (like syslogd), and comes with regular expression matching with which you can launch external scripts when specific patterns are found. It is very good at taking action when needed.

The standard configuration is usually enough. If you want to be notified by email whenever a password failure occurs use one of the following scripts.

For postfix:

Code Listing 2: /usr/local/sbin/mail_pwd_failures.sh for postfix

```
#! /bin/sh
echo "$3" | mail -s "Warning (program : $2)" root
```

For netqmail:

Code Listing 3: /usr/local/sbin/mail_pwd_failures.sh for netqmail

```
#!/bin/sh
echo "To: root
Subject:Failure (Warning: $2)
$3
" | /var/qmail/bin/qmail-inject -f root
```

Remember to make the script executable by issuing `/bin/chmod +x /usr/local/sbin/mail_pwd_failures.sh`

Then uncomment the command line under "Password failures" in `/etc/metalog/metalog.conf` like:

Code Listing 4: /etc/metalog/metalog.conf

```
command  = "/usr/local/sbin/mail_pwd_failures.sh"
```

3.d. Syslog-ng

Syslog-ng provides some of the same features as syslog and metalog with a small difference. It can filter messages based on level and content (like metalog), provide remote logging like syslog, handle logs from syslogd (even streams from Solaris), write to a TTY, execute programs, and it can act as a logging server. Basically it is the best of both loggers combined with advanced configuration.

Below is a classic configuration file slightly modified.

Code Listing 5: /etc/syslog-ng/syslog-ng.conf

```
options { chain_hostnames(off); sync(0); };

#source where to read log
source src { unix-stream("/dev/log"); internal(); };
source kernsrc { file("/proc/kmsg"); };

#define destinations
destination authlog { file("/var/log/auth.log"); };
destination syslog { file("/var/log/syslog"); };
destination cron { file("/var/log/cron.log"); };
destination daemon { file("/var/log/daemon.log"); };
destination kern { file("/var/log/kern.log"); };
destination lpr { file("/var/log/lpr.log"); };
destination user { file("/var/log/user.log"); };
destination mail { file("/var/log/mail.log"); };

destination mailinfo { file("/var/log/mail.info"); };
destination mailwarn { file("/var/log/mail.warn"); };
destination mailerr { file("/var/log/mail.err"); };

destination newscrit { file("/var/log/news/news.crit"); };
destination newserr { file("/var/log/news/news.err"); };
destination newsnotice { file("/var/log/news/news.notice"); };

destination debug { file("/var/log/debug"); };
destination messages { file("/var/log/messages"); };
destination console { usertty("root"); };
destination console_all { file("/dev/tty12"); };
destination xconsole { pipe("/dev/xconsole"); };

#create filters
filter f_authpriv { facility(auth, authpriv); };
filter f_syslog { not facility(authpriv, mail); };
filter f_cron { facility(cron); };
filter f_daemon { facility(daemon); };
filter f_kern { facility(kern); };
filter f_lpr { facility(lpr); };
filter f_mail { facility(mail); };
filter f_user { facility(user); };
filter f_debug { not facility(auth, authpriv, news, mail); };
filter f_messages { level(info..warn)
        and not facility(auth, authpriv, mail, news); };
filter f_emergency { level(emerg); };

filter f_info { level(info); };
filter f_notice { level(notice); };
filter f_warn { level(warn); };
filter f_crit { level(crit); };
filter f_err { level(err); };
filter f_failed { match("failed"); };
filter f_denied { match("denied"); };

#connect filter and destination
log { source(src); filter(f_authpriv); destination(authlog); };
log { source(src); filter(f_syslog); destination(syslog); };
log { source(src); filter(f_cron); destination(cron); };
log { source(src); filter(f_daemon); destination(daemon); };
log { source(kernsrc); filter(f_kern); destination(kern); };
log { source(src); filter(f_lpr); destination(lpr); };
log { source(src); filter(f_mail); destination(mail); };
log { source(src); filter(f_user); destination(user); };
log { source(src); filter(f_mail); filter(f_info); destination(mailinfo); };
log { source(src); filter(f_mail); filter(f_warn); destination(mailwarn); };
log { source(src); filter(f_mail); filter(f_err); destination(mailerr); };

log { source(src); filter(f_debug); destination(debug); };
log { source(src); filter(f_messages); destination(messages); };
log { source(src); filter(f_emergency); destination(console); };

#default log
log { source(src); destination(console_all); };
```

Syslog-ng is very easy to configure, but it is also very easy to miss something in the configuration file since it is huge. The author still promises some extra features like encryption, authentication, compression and MAC (Mandatory Access Control) control. With these options it will be a perfect for network logging, since the attacker cannot spy on the log.

And syslog-ng does have one other advantage: it does not have to run as root!

3.e. Log analysis with Logcheck

Of course, keeping logs alone is only half the battle. An application such as Logcheck can make regular log analysis much easier. Logcheck is a script, accompanied by a binary called `logtail`, that runs from your cron daemon and checks your logs against a set of rules for suspicious activity. It then mails the output to root's mailbox.

Logcheck and logtail are part of the `app-admin/logsentry` package.

Logcheck uses four files to filter important log entries from the unimportant. These files are `logcheck.hacking`, which contains known hacking attack messages, `logcheck.violations`, which contains patterns indicating security violations, `logcheck.violations.ignore`, which contains keywords likely to be matched by the violations file, allowing normal entries to be ignored, and `logcheck.ignore`, which matches those entries to be ignored.

Warning: Do not leave `logcheck.violations.ignore` empty. Logcheck uses `grep` to parse logs, some versions of which will take an empty file to mean wildcard. All violations would thus be ignored.

4. Mounting Partitions

4.a. Mounting partitions

When mounting an `ext2`, `ext3`, or `reiserfs` partition, you have several options you can apply to the file `/etc/fstab`. The options are:

- `nosuid` - Will ignore the SUID bit and make it just like an ordinary file
- `noexec` - Will prevent execution of files from this partition
- `nodev` - Ignores devices

Unfortunately, these settings can easily be circumvented by executing a non-direct path. However, setting `/tmp` to noexec will stop the majority of exploits designed to be executed directly from `/tmp`.

Code Listing 1: /etc/fstab

```
/dev/sda1 /boot ext2 noauto,noatime 1 1
/dev/sda2 none swap sw 0 0
/dev/sda3 / reiserfs notail,noatime 0 0
/dev/sda4 /tmp reiserfs notail,noatime,nodev,nosuid,noexec 0 0
/dev/sda5 /var reiserfs notail,noatime,nodev 0 0
/dev/sda6 /home reiserfs notail,noatime,nodev,nosuid 0 0
/dev/sda7 /usr reiserfs notail,noatime,nodev,ro 0 0
/dev/cdroms/cdrom0 /mnt/cdrom iso9660 noauto,ro 0 0
proc /proc proc defaults 0 0
```

Warning: Placing `/tmp` in `noexec` mode can prevent certain scripts from executing properly.

Note: For disk quotas see the Quotas section.

Note: I do not set `/var` to `noexec` or `nosuid`, even if files normally are never executed from this mount point. The reason for this is that netqmail is installed in `/var/qmail` and must be allowed to execute and access one SUID file. I setup `/usr` in read-only mode since I never write anything there unless I want to update Gentoo. Then I remount the file system in read-write mode, update and remount again.

Note: Even if you do not use netqmail, Gentoo still needs the executable bit set on `/var/tmp` since ebuilds are made here. But an alternative path can be setup if you insist on having `/var` mounted in `noexec` mode.

5. User/Group Limitations

5.a. /etc/security/limits.conf

Controlling resource usage can be very effective when trying to prevent a local Denial of Service or restricting the maximum allowed logins for a group or user. However, too strict settings will impede on your system's behavior and will result in program failures so make sure that you check each setting first.

Code Listing 1: /etc/security/limits.conf

```
*     soft core 0
*     hard core 0
*     hard nproc 15
*     hard rss 10000
*     -    maxlogins 2
@dev hard core 100000
@dev soft nproc 20
@dev hard nproc 35
@dev -    maxlogins 10
```

If you find yourself trying to set `nproc` or `maxlogins` to 0, maybe you should delete the user instead. The example above sets the group `dev` settings for processes, core file and `maxlogins`. The rest is set to a default value.

Note: `/etc/security/limits.conf` is part of the PAM package and will only apply to packages that use PAM.

5.b. /etc/limits

`/etc/limits` is very similar to the limit file `/etc/security/limits.conf`. The only difference is the format and that it only works on users or wild cards (not groups). Let's have a look at a sample configuration:

Code Listing 2: /etc/limits

```
*   L2 C0 U15 R10000
kn L10 C100000 U35
```

Here we set the default settings and a specific setting for the user kn. Limits are part of the sys-apps/shadow package. It is not necessary to set any limits in this file if you have disabled `pam` in `make.conf` or not configured PAM properly.

5.c. Quotas

Warning: Make sure the file systems you are working with support quotas. In order to use quotas on ReiserFS, you must patch your kernel with patches available from Namesys. User tools are available from the Linux DiskQuota project. While quotas do work with ReiserFS, you may encounter other issues while trying to use them--you have been warned!

Putting quotas on a file system restricts disk usage on a per-user or per-group basis. Quotas are enabled in the kernel and added to a mount point in `/etc/fstab`. The kernel option is enabled in the kernel configuration under `File systems->Quota support`. Apply the following settings, rebuild the kernel and reboot using the new kernel.

Start by installing quotas with `emerge quota`. Then modify your `/etc/fstab` and add `usrquota` and `grpquota` to the partitions that you want to restrict disk usage on, like in the example below.

Code Listing 3: /etc/fstab

```
/dev/sda1 /boot ext2 noauto,noatime 1 1
/dev/sda2 none swap sw 0 0
/dev/sda3 / reiserfs notail,noatime 0 0
/dev/sda4 /tmp ext3 noatime,nodev,nosuid,noexec,usrquota,grpquota 0 0
/dev/sda5 /var ext3 noatime,nodev,usrquota,grpquota 0 0
/dev/sda6 /home ext3 noatime,nodev,nosuid,usrquota,grpquota 0 0
/dev/sda7 /usr reiserfs notail,noatime,nodev,ro 0 0
/dev/cdroms/cdrom0 /mnt/cdrom iso9660 noauto,ro 0 0
proc /proc proc defaults 0 0
```

On every partition that you have enabled quotas, create the quota files (`aquota.user` and `aquota.group`) and place them in the root of the partition.

Code Listing 4: Creating the quota files

```
# touch /tmp/aquota.user
# touch /tmp/aquota.group
# chmod 600 /tmp/aquota.user
# chmod 600 /tmp/aquota.group
```

This step has to be done on every partition where quotas are enabled. After adding and configuring the quota files, we need to add the `quota` script to the boot run level.

Important: XFS does all quota checks internally, and does *not* need the `quota` script added to the boot runlevel. There may be other filesystems not listed in this document with similar behavior, so please read the manpages for your filesystem to learn more about how it handles quota checks.

Code Listing 5: Adding quota to the boot runlevel

```
# rc-update add quota boot
```

We will now configure the system to check the quotas once a week by adding the following line to `/etc/crontab`:

Code Listing 6: Adding quota check to crontab

```
0 3 * * 0 /usr/sbin/quotacheck -avug.
```

After rebooting the machine, it is time to setup the quotas for users and groups. `edquota -u kn` will start the editor defined in $EDITOR (default is nano) and let you edit the quotas of the user kn. `edquota -g` will do the same thing for groups.

Code Listing 7: Setting up quota's for user kn

```
Quotas for user kn:
/dev/sda4: blocks in use: 2594, limits (soft = 5000, hard = 6500)
          inodes in use: 356, limits (soft = 1000, hard = 1500)
```

For more detail read `man edquota` or the Quota mini howto.

5.d. /etc/login.defs

If your security policy states that users should change their password every other week, change the value `PASS_MAX_DAYS` to 14 and `PASS_WARN_AGE` to 7. It is recommended that you use password aging since brute force methods can find any password, given enough time. We also encourage you to set `LOG_OK_LOGINS` to yes.

5.e. /etc/login.access

The `login.access` file is also part of the sys-apps/shadow package, which provides a login access control table. This table is used to control who can and cannot login based on user name, group name or host name. By default, all users on the system are allowed to login, so the file consists only of comments and examples. Whether you are securing your server or workstation, we recommend that you setup this file so no one other than yourself (the admin) has access to the console.

Note: These settings do not apply for root.

Code Listing 8: /etc/login.access

```
-:ALL EXCEPT wheel sync:console
-:wheel:ALL EXCEPT LOCAL .gentoo.org
```

Important: Be careful when configuring these options, since mistakes will leave you with no access to the machine if you do not have root access.

Note: These settings do not apply to SSH, since SSH does not execute `/bin/login` per default. This can be enabled by setting `UseLogin yes` in `/etc/ssh/sshd_config`.

This will setup login access so members of the wheel group can login locally or from the gentoo.org domain. Maybe too paranoid, but better to be safe than sorry.

6. File Permissions

6.a. World readable

Normal users should not have access to configuration files or passwords. An attacker can steal passwords from databases or web sites and use them to deface--or even worse, delete--data. This is why it is important that your file permissions are correct. If you are sure that a file is only used by root, assign it with the permissions `0600` and assign

the file to the correct user with `chown`.

6.b. World/Group writable

Code Listing 1: Finding world-writable files and directories

```
# find / -type f \( -perm -2 -o -perm -20 \) -exec ls -lg {} \; 2>/dev/null >writable.txt
# find / -type d \( -perm -2 -o -perm -20 \) -exec ls -ldg {} \; 2>/dev/null >>writable.txt
```

This will create a huge file with permission of all files having either write permission set to the group or everybody. Check the permissions and eliminate world writable files to everyone, by executing `/bin/chmod o-w` on the files.

6.c. SUID/SGID files

Files with the SUID or SGID bit set execute with privileges of the *owning* user or group and not the user executing the file. Normally these bits are used on files that must run as root in order to do what they do. These files can lead to local root compromises (if they contain security holes). This is dangerous and files with the SUID or SGID bits set should be avoided at any cost. If you do not use these files, use `chmod 0` on them or unmerge the package that they came from (check which package they belong to by using `equery`; if you do not already have it installed simply type `emerge gentoolkit`). Otherwise just turn the SUID bit off with `chmod -s`.

Code Listing 2: Finding setuid files

```
# find / -type f \( -perm -004000 -o -perm -002000 \) -exec ls -lg {} \; 2>/dev/null >suidfiles.tx
```

This will create a file containing a list of all the SUID/SGID files.

Code Listing 3: List of setuid binaries

```
/bin/su
/bin/ping
/bin/mount
/bin/umount
/var/qmail/bin/qmail-queue
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/crontab
/usr/bin/chage
/usr/bin/expiry
/usr/bin/sperl5.6.1
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/procmail
/usr/bin/suidperl
/usr/lib/misc/pt_chown
/usr/sbin/unix_chkpwd
/usr/sbin/traceroute
/usr/sbin/pwdb_chkpwd
```

By default Gentoo Linux does not have a lot of SUID files (though this depends on what you installed), but you might get a list like the one above. Most of the commands should not be used by normal users, only root. Switch off the SUID bit on `ping`, `mount`, `umount`, `chfn`, `chsh`, `newgrp`, `suidperl`, `pt_chown` and `traceroute` by executing `chmod -s` on every file. Don't remove the bit on `su`, `qmail-queue` or `unix_chkpwd`. Removing setuid from those files will prevent you from `su`'ing and receiving mail. By removing the bit (where it is safe to do so) you remove the possibility of a normal user (or an attacker) gaining root access through any of these files.

The only SUID files that I have on my system are `su`, `passwd`, `gpasswd`, `qmail-queue`, `unix_chkpwd` and `pwdb_chkpwd`. But if you are running X, you might have some more, since X needs the elevated access afforded by SUID.

6.d. SUID/SGID binaries and Hard links

A file is only considered deleted when there are no more links pointing to it. This might sound like a strange concept, but consider that a filename like `/usr/bin/perl` is actually a link to the inode where the data is stored. Any number of links can point to the file, and until all of them are gone, the file still exists.

If your users have access to a partition that isn't mounted with `nosuid` or `noexec` (for example, if `/tmp`, `/home`, or

`/var/tmp` are not separate partitions) you should take care to ensure your users don't create hard links to SUID or SGID binaries, so that after Portage updates they still have access to the old versions.

Warning: if you have received a warning from portage about remaining hard links, and your users can write to a partition that allows executing SUID/SGID files, you should read this section carefully. One of your users may be attempting to circumvent your update by keeping an outdated version of a program. If your users cannot create their own SUID files, or can only execute programs using the dynamic loader (partitions mounted `noexec`), you do not have to worry.

Note: Users do not need read access to a file to create a link to it, they only need read permission to the directory that contains it.

To check how many links a file has, you can use the `stat` command.

Code Listing 4: Stat command

```
$ stat /bin/su
  File: `/bin/su'
  Size: 29350           Blocks: 64         IO Block: 131072 regular file
Device: 900h/2304d      Inode: 2057419     Links: 1
Access: (4711/-rws--x--x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2005-02-07 01:59:35.000000000 +0000
Modify: 2004-11-04 01:46:17.000000000 +0000
Change: 2004-11-04 01:46:17.000000000 +0000
```

To find the SUID and SGID files with multiple links, you can use `find`.

Code Listing 5: Finding multiply linked suid/sgid binaries

```
$ find / -type f \( -perm -004000 -o -perm -002000 \) -links +1 -ls
```

7. PAM

7.a. PAM

PAM is a suite of shared libraries that provide an alternative way providing user authentication in programs. The `pam` USE flag is turned on by default. Thus the PAM settings on Gentoo Linux are pretty reasonable, but there is always room for improvement. First install cracklib.

Code Listing 1: Installing cracklib

```
# emerge cracklib
```

Code Listing 2: /etc/pam.d/passwd

```
auth     required pam_unix.so shadow nullok
account  required pam_unix.so
password required pam_cracklib.so difok=3 retry=3 minlen=8 dcredit=-2 ocredit=-2
password required pam_unix.so md5 use_authtok
session  required pam_unix.so
```

This will add the cracklib which will ensure that the user passwords are at least 8 characters and contain a minimum of 2 digits, 2 other characters, and are more than 3 characters different from the last password. This forces the user to choose a good password (password policy). Check the PAM documentation for more options.

Code Listing 3: /etc/pam.d/sshd

```
auth     required pam_unix.so nullok
auth     required pam_shells.so
auth     required pam_nologin.so
auth     required pam_env.so
account  required pam_unix.so
password required pam_cracklib.so difok=3 retry=3 minlen=8 dcredit=-2 ocredit=-2 use_authtok
password required pam_unix.so shadow md5
session  required pam_unix.so
session  required pam_limits.so
```

Every service not configured with a PAM file in `/etc/pam.d` will use the rules in `/etc/pam.d/other`. The defaults are set to `deny`, as they should be. But I like to have a lot of logs, which is why I added `pam_warn.so`. The last

configuration is pam_limits, which is controlled by /etc/security/limits.conf. See the /etc/security/limits.conf section for more on these settings.

Code Listing 4: /etc/pam.d/other

```
auth     required pam_deny.so
auth     required pam_warn.so
account  required pam_deny.so
account  required pam_warn.so
password required pam_deny.so
password required pam_warn.so
session  required pam_deny.so
session  required pam_warn.so
```

8. TCP Wrappers

8.a. TCP Wrappers

This is a way of controlling access to services normally run by inetd (which Gentoo does not have), but it can also be used by xinetd and other services.

Note: The service should be executing tcpd in its server argument (in xinetd). See the chapter on xinetd for more information.

Code Listing 1: /etc/hosts.deny

```
ALL:PARANOID
```

Code Listing 2: /etc/hosts.allow

```
ALL: LOCAL @wheel
time: LOCAL, .gentoo.org
```

As you can see the format is very similar to the one in /etc/login.access. Tcpd supports a specific service; it does not overlap with /etc/login.access. These settings only apply to services using tcp wrappers.

It is also possible to execute commands when a service is accessed (this can be used when activating relaying for dial-in users) but it is not recommended, since people tend to create more problems than they are trying to solve. An example could be that you configure a script to send an e-mail every time someone hits the deny rule, but then an attacker could launch a DoS attack by keep hitting the deny rule. This will create a lot of I/O and e-mails so don't do it!. Read the man 5 hosts_access for more information.

9. Kernel Security

9.a. Removing functionality

The basic rule when configuring the kernel is to remove everything that you do not need. This will not only create a small kernel but also remove the vulnerabilities that may lie inside drivers and other features.

Also consider turning off loadable module support. Even though it is possible to add root kits without this features, it does make it harder for normal attackers to install root kits via kernel modules.

9.b. The proc filesystem

Many kernel parameters can be altered through the /proc file system or by using sysctl.

To dynamically change kernel parameters and variables on the fly, you need CONFIG_SYSCTL defined in your kernel. This is on by default in a standard 2.4 kernel.

Code Listing 1: Deactivate IP forwarding

```
# /bin/echo "0" > /proc/sys/net/ipv4/ip_forward
```

Make sure that IP forwarding is turned off. We only want this for a multi-homed host. It's advised to set or unset this flag before all other flags since it enabled/disables other flags as well.

Code Listing 2: Drop ping packets

```
# /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

This will cause the kernel to simply ignore all ping messages (also known as ICMP type 0 messages). The reason for this is that an IP packet carrying an ICMP message can contain a payload with information other than you think. Administrators use ping as a diagnostic tool and often complain if it is disabled, but there is no reason for an outsider to be able to ping. However, since it sometimes can be handy for insiders to be able to ping, you can disable ICMP type 0 messages in the firewall (allowing local administrators to continue to use this tool).

Code Listing 3: Ignore broadcast pings

```
# /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

This disables response to ICMP broadcasts and will prevent Smurf attacks. The Smurf attack works by sending an ICMP type 0 (ping) message to the broadcast address of a network. Typically the attacker will use a spoofed source address. All the computers on the network will respond to the ping message and thereby flood the host at the spoofed source address.

Code Listing 4: Disable source routed packets

```
# /bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
```

Do not accept source routed packets. Attackers can use source routing to generate traffic pretending to originate from inside your network, but that is actually routed back along the path from which it came, so attackers can compromise your network. Source routing is rarely used for legitimate purposes, so it is safe to disable it.

Code Listing 5: Disable redirect acceptance

```
# /bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
# /bin/echo "0" > /proc/sys/net/ipv4/conf/all/secure_redirects
```

Do not accept ICMP redirect packets. ICMP redirects can be used to alter your routing tables, possibly to a malicious end.

Code Listing 6: Protect against bad error messages

```
# /bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```

Enable protection against bogus error message responses.

Code Listing 7: Enable reverse path filtering

```
# for i in /proc/sys/net/ipv4/conf/*; do
        /bin/echo "1" > $i/rp_filter
done
```

Turn on reverse path filtering. This helps make sure that packets use legitimate source addresses by automatically rejecting incoming packets if the routing table entry for their source address does not match the network interface they are arriving on. This has security advantages because it prevents IP spoofing. We need to enable it for each `net/ipv4/conf/*` otherwise source validation isn't fully functional.

Warning: However turning on reverse path filtering can be a problem if you use asymmetric routing (packets from you to a host take a different path than packets from that host to you) or if you operate a non-routing host which has several IP addresses on different interfaces.

Code Listing 8: Log all spoofed, source routed and redirect packets

```
# /bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
```

Log spoofed packets, source routed packets and redirect packets.

All these settings will be reset when the machine is rebooted. I suggest that you add them to `/etc/sysctl.conf`, which is automatically sourced by the `/etc/init.d/bootmisc` init script.

The syntax for /etc/sysctl.conf is pretty straightforward. Strip off the /proc/sys/ from the previously mentioned paths and substitute / with .:

Code Listing 9: Translating to sysctl.conf

```
(Manual using echo):
/bin/echo "0" > /proc/sys/net/ipv4/ip_forward

(Automatic in sysctl.conf:)
net.ipv4.ip_forward = 0
```

9.c. Grsecurity

The patch from Grsecurity is standard in the sys-kernel/hardened-sources but is disabled by default. Configure your kernel as you normally do and then configure the Grsecurity options. An in-depth explanation on the available Grsecurity options is available on the Gentoo Hardened project page.

Recent hardened-sources provide the 2.* version of Grsecurity. For more information on this improved Grsecurity patch set, please consult the documentation available on the Grsecurity home page.

9.d. Kerneli

Kerneli is a patch that adds encryption to the existing kernel. By patching your kernel you will get new options such as cryptographic ciphers, digest algorithms and cryptographic loop filters.

Warning: The kerneli patch is currently not in a stable version for the latest kernel, so be careful when using it.

9.e. Other kernel patches

- The OpenWall Project
- Linux Intrusion Detection System
- Rule Set Based Access Control
- NSA's security enhanced kernel
- Wolk

And there are probably a lot more.

10. Securing Services

10.a. Apache

Apache (1.3.26) comes with a pretty decent configuration file but again, we need to improve some things, like binding Apache to one address and preventing it from leaking information. Below are the options that you should apply the configuration file.

If you did not disable ssl in your /etc/make.conf before installing Apache, you should have access to an ssl enabled server. Just add the following line to enable it.

Code Listing 1: /etc/conf.d/apache

```
HTTPD_OPTS="-D SSL"
```

Code Listing 2: /etc/apache/conf/apache.conf

```
#Make it listen on your ip
Listen 127.0.0.1
BindAddress 127.0.0.1
#It is not a good idea to use nobody or nogroup -
#for every service not running as root
#(just add the user apache with group apache)
User apache
Group apache
#Will keep apache from telling about the version
ServerSignature Off
ServerTokens Prod
```

Apache is compiled with `--enable-shared=max` and `--enable-module=all`. This will by default enable all modules, so you should comment out all modules in the `LoadModule` section (`LoadModule` and `AddModule`) that you do not use. Restart the service by executing `/etc/init.d/apache restart`.

Documentation is available at http://www.apache.org.

10.b. Bind

One can find documentation at the Internet Software Consortium. The BIND 9 Administrator Reference Manual is also in the `doc/arm`.

The newer BIND ebuilds support chrooting out of the box. After emerging `bind` follow these simple instructions:

Code Listing 3: Chrooting BIND

```
ebuild /var/db/pkg/net-dns/bind-9.2.2-r2/bind-9.2.2-r2.ebuild config\`"
(Before running the above command you might want to change the chroot
directory in /etc/conf.d/named. Otherwise /chroot/dns will be used.)
(You might need to substitute the version number with the current version number )
```

10.c. Djbdns

Djbdns is a DNS implementation on the security of which its author is willing to bet money. It is very different from how Bind 9 works but worth a try. More information can be obtained from http://www.djbdns.org.

10.d. FTP

Generally, using FTP (File Transfer Protocol) is a bad idea. It uses unencrypted data (ie. passwords are sent in clear text), listens on 2 ports (normally port 20 and 21), and attackers are frequently looking for anonymous logins for trading warez. Since the FTP protocol contains several security problems you should instead use `sftp` or HTTP. If this is not possible, secure your services as well as you can and prepare yourself.

10.e. Mysql

If you only need local applications to access the `mysql` database, uncomment the following line in `/etc/mysql/my.cnf`.

Code Listing 4: Disable network access

```
skip-networking
```

Then we disable the use of the LOAD DATA LOCAL INFILE command. This is to prevent against unauthorized reading from local files. This is relevant when new SQL Injection vulnerabilities in PHP applications are found.

Code Listing 5: Disable LOAD DATA LOCAL INFILE in the [mysqld] section

```
set-variable=local-infile=0
```

Next, we must remove the sample database (test) and all accounts except the local `root` account.

Code Listing 6: Removing sample database and all unnecessary users

```
mysql> drop database test;
mysql> use mysql;
mysql> delete from db;
mysql> delete from user where not (host="localhost" and user="root");
mysql> flush privileges;
```

Warning: Be careful with the above if you have already configured user accounts.

Note: If you have been changing passwords from the MySQL prompt, you should always clean out `~/.mysql_history` and `/var/log/mysql/mysql.log` as they store the executed SQL commands with passwords in clear text.

10.f. Proftpd

Proftpd has had several security problems, but most of them seem to have been fixed. Nonetheless, it is a good idea to apply some enhancements:

Code Listing 7: /etc/proftpd/proftpd.conf

```
ServerName "My ftp daemon"
#Don't show the ident of the server
ServerIdent on "Go away"

#Makes it easier to create virtual users
RequireValidShell off

#Use alternative password and group file (passwd uses crypt format)
AuthUserFile "/etc/proftpd/passwd"
AuthGroupFile "/etc/proftpd/group"

# Permissions
Umask 077

# Timeouts and limitations
MaxInstances 30
MaxClients 10 "Only 10 connections allowed"
MaxClientsPerHost 1 "You have already logged on once"
MaxClientsPerUser 1 "You have already logged on once"
TimeoutStalled 10
TimeoutNoTransfer 20
TimeoutLogin 20

#Chroot everyone
DefaultRoot ~

#don't run as root
User  nobody
Group nogroup

#Log every transfer
TransferLog /var/log/transferlog

#Problems with globbing
DenyFilter \*.*/
```

One can find documentation at http://www.proftpd.org.

10.g. Pure-ftpd

Pure-ftpd is an branch of the original trollftpd, modified for security reasons and functionality by Frank Dennis.

Use virtual users (never system accounts) by enabling the `AUTH` option. Set this to `-lpuredb:/etc/pureftpd.pdb` and create your users by using `/usr/bin/pure-pw`.

Code Listing 8: /etc/conf.d/pure-ftpd

```
AUTH="-lpuredb:/etc/pureftpd.pdb"

## Misc. Others ##
MISC_OTHER="-A -E -X -U 177:077 -d -4 -L100:5 -I 15"
```

Configure your `MISC_OTHER` setting to deny anonymous logins (`-E`), chroot everyone (`-A`), prevent users from reading or writing to files beginning with a . (dot) (`-X`), max idle time (`-I`), limit recursion (`-L`), and a reasonable `umask`.

Warning: Do *not* use the `-w` or `-W` options! If you want to have a warez site, stop reading this guide!

One can find documentation at http://www.pureftpd.org.

10.h. Vsftpd

Vsftpd (short for very secure ftp) is a small ftp daemon running a reasonably default configuration. It is simple and does not have as many features as pureftp and proftp.

Code Listing 9: /etc/vsftpd

```
anonymous_enable=NO
local_enable=YES

#read only
write_enable=NO

#enable logging of transfers
xferlog_std_format=YES

idle_session_timeout=20
data_connection_timeout=20
nopriv_user=nobody

chroot_list_enable=YES
chroot_list_file=/etc/vsftpd/chrootlist

ls_recurse_enable=NO
```

As you can see, there is no way for this service to have individual permissions, but when it comes to anonymous settings it is quite good. Sometimes it can be nice to have an anonymous ftp server (for sharing open source), and vsftpd does a really good job at this.

10.i. Netqmail

Netqmail is often considered to be a very secure mail server. It is written with security (and paranoia) in mind. It does not allow relaying by default and has not had a security hole since 1996. Simply `emerge netqmail` and go configure!

10.j. Samba

Samba is a protocol to share files with Microsoft/Novell networks and it should *not* be used over the Internet. Nonetheless, it still needs securing.

Code Listing 10: /etc/samba/smb.conf

```
[global]
  #Bind to an interface
  interfaces = eth0 10.0.0.1/32

  #Make sure to use encrypted password
  encrypt passwords = yes
  directory security mask = 0700

  #allow traffic from 10.0.0.*
  hosts allow = 10.0.0.

  #Enables user authentication
  #(don't use the share mode)
  security = user

  #Disallow privileged accounts
  invalid users = root @wheel

  #Maximum size smb shows for a share (not a limit)
  max disk size = 102400

  #Uphold the password policy
  min password length = 8
  null passwords = no

  #Use PAM (if added support)
  obey pam restrictions = yes
  pam password change = yes
```

Make sure that permissions are set correct on every share and remember to read the documentation.

Now restart the server and add the users who should have access to this service. This is done though the command `/usr/bin/smbpasswd` with the parameter `-a`.

10.k. ssh

The only securing that OpenSSH needs is turning on a stronger authentication based on public key encryption. Too many sites (like http://www.sourceforge.net, http://www.php.net and http://www.apache.org) have suffered unauthorized

intrusion due to password leaks or bad passwords.

Code Listing 11: /etc/ssh/sshd_config

```
#Only enable version 2
Protocol 2

#Disable root login. Users have to su to root
PermitRootLogin no

#Turn on Public key authentication
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys

#Disable .rhost and normal password authentication
HostbasedAuthentication no
PasswordAuthentication no
PermitEmptyPasswords no

#Only allow userin the wheel or admin group to login
AllowGroups wheel admin

#In those groups only allow the following users
#The @<domainname> is optional but replaces the
#older AllowHosts directive
AllowUsers kn@gentoo.org bs@gentoo.org

#Logging
SyslogFacility AUTH
LogLevel INFO

ListenAddress 127.0.0.1
```

Also verify that you don't have `UsePAM yes` in your configuration file as it overrides the public key authentication mechanism.

Now all that your users have to do is create a key (on the machine they want to login from) with the following command:

Code Listing 12: Create a DSA keypair

```
# /usr/bin/ssh-keygen -t dsa
```

And type in a pass phrase.

Code Listing 13: Output of ssh-keygen

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/kn/.ssh/id_dsa):[Press enter]
Created directory '/home/kn/.ssh'.
Enter passphrase (empty for no passphrase): [Enter passphrase]
Enter same passphrase again: [Enter passphrase again]
Your identification has been saved in /home/kn/.ssh/id_dsa.
Your public key has been saved in /home/kn/.ssh/id_dsa.pub.
The key fingerprint is:
07:24:a9:12:7f:83:7e:af:b8:1f:89:a3:48:29:e2:a4 kn@knielsen
```

This will add two files in your `~/.ssh/` directory called `id_dsa` and `id_dsa.pub`. The file called `id_dsa` is your private key and should be kept from other people than yourself. The other file `id_dsa.pub` is to be distributed to every server that you have access to. Add the key to the users home directory in `~/.ssh/authorized_keys` and the user should be able to login:

Code Listing 14: Adding the id_dsa.pub file to the authorized_keys file

```
$ scp id_dsa.pub other-host:/var/tmp/currenthostname.pub
$ ssh other-host
password:
$ cat /var/tmp/currenthostname.pub >> ~/.ssh/authorized_keys
```

Now your users should guard this private key well. Put it on a media that they always carry with them or keep it on their workstation (put this in the password policy).

For more information go to the OpenSSH web site.

10.l. Using xinetd

xinetd is a replacement for `inetd` (which Gentoo does not have), the Internet services daemon. It supports access control based on the address of the remote host and the time of access. It also provide extensive logging capabilities, including server start time, remote host address, remote user name, server run time, and actions requested.

As with all other services it is important to have a good default configuration. But since `xinetd` is run as root and supports protocols that you might not know how they work, we recommend not to use it. But if you want to use it anyway, here is how you can add some security to it:

Code Listing 15: Install xinetd

```
# emerge xinetd tcp-wrappers
```

And edit the configuration file:

Code Listing 16: /etc/xinetd.conf

```
defaults
{
 only_from = localhost
 instances = 10
 log_type = SYSLOG authpriv info
 log_on_success = HOST PID
 log_on_failure = HOST
 cps = 25 30
}

# This will setup pserver (cvs) via xinetd with the following settings:
# max 10 instances (10 connections at a time)
# limit the pserver to tcp only
# use the user cvs to run this service
# bind the interfaces to only 1 ip
# allow access from 10.0.0.*
# limit the time developers can use cvs from 8am to 5pm
# use tpcd wrappers (access control controlled in
# /etc/hosts.allow and /etc/hosts.deny)
# max_load on the machine set to 1.0
# The disable flag is per default set to no but I like having
# it in case of it should be disabled
service cvspserver
{
 socket_type = stream
 protocol = tcp
 instances = 10
 protocol = tcp
 wait = no
 user = cvs
 bind = 10.0.0.2
 only_from = 10.0.0.0
 access_times = 8:00-17:00
 server = /usr/sbin/tcpd
 server_args = /usr/bin/cvs --allow-root=/mnt/cvsdisk/cvsroot pserver
 max_load = 1.0
 log_on_failure += RECORD
 disable = no
}
```

For more information read `man 5 xinetd.conf`.

10.m. X

By default Xorg is configured to act as an Xserver. This can be dangerous since X uses unencrypted TCP connections and listens for xclients.

Important: If you do not need this service disable it!

But if you depend on using your workstation as a Xserver use the `/usr/X11R6/bin/xhost` command with caution. This command allows clients from other hosts to connect and use your display. This can become handy if you need an X application from a different machine and the only way is through the network, but it can also be exploited by an attacker. The syntax of this command is `/usr/X11R6/bin/xhost +hostname`

Warning: Do not ever use the `xhost +` feature! This will allow any client to connect and take control of your X. If an attacker can get access to your X, he can log your keystrokes and take control over your desktop. If you have to use it always remember to specify a host.

A more secure solution is to disable this feature completely by starting X with `startx -- -nolisten tcp` or disable it permanently in the configuration.

Code Listing 17: /usr/X11R6/bin/startx

```
defaultserverargs="-nolisten tcp"
```

To make sure that `startx` does not get overwritten when emerging a new version of Xorg you must protect it. Add the following line to `/etc/make.conf`:

Code Listing 18: /etc/make.conf

```
CONFIG_PROTECT_MASK="/usr/X11R6/bin/startx"
```

If you use a graphical login manager you need a different approach.

For `gdm` (Gnome Display Manager)

Code Listing 19: /etc/X11/gdm/gdm.conf

```
[server-Standard]
command=/usr/X11R6/bin/X -nolisten tcp
```

For `xdm` (X Display Manager) and `kdm` (Kde Display Manager)

Code Listing 20: /etc/X11/xdm/Xservers

```
:0 local /usr/bin/X11/X -nolisten tcp
```

11. Chrooting and Virtual Servers

11.a. Chrooting

Chrooting a service is a way of limiting a service (or user) environment to only accessing what it should and not gaining access (or information) that could lead to root access. By running the service as another user than `root` (`nobody`, `apache`, `named`) an attacker can only access files with the permissions of this user. This means that an attacker cannot gain `root` access even if the services has a security flaw.

Some services like `pure-ftpd` and `bind` have features for chrooting, and other services do not. If the service supports it, use it, otherwise you have to figure out how to create your own. Lets see how to create a chroot, for a basic understanding of how chroots work, we will test it with `bash` (easy way of learning).

Create the `/chroot` directory with `mkdir /chroot`. And find what dynamic libraries that `bash` is compiled with (if it is compiled with `-static` this step is not necessary):

The following command will create a list of libraries used by `bash`.

Code Listing 1: Get listing of used libraries

```
# ldd /bin/bash
  libncurses.so.5 => /lib/libncurses.so.5 (0x4001b000)
  libdl.so.2 => /lib/libdl.so.2 (0x40060000)
  libc.so.6 => /lib/libc.so.6 (0x40063000)
  /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Now lets create the environment for `bash`.

Code Listing 2: Create chroot-environment for bash

```
# mkdir /chroot/bash
# mkdir /chroot/bash/bin
# mkdir /chroot/bash/lib
```

Next copy the files used by `bash` (`/lib`) to the chrooted `lib` and copy the bash command to the chrooted `bin` directory. This will create the exact same environment, just with less functionality. After copying try it out: `chroot /chroot/bash /bin/bash`. If you get an prompt saying `/` it works! Otherwise it will properly tell you what a file is missing. Some shared libraries depend on each other.

You will notice that inside the chroot nothing works except `echo`. This is because we have no other commands in out chroot environment than bash and `echo` is a build-in functionality.

This is basically the same way you would create a chrooted service. The only difference is that services sometimes rely on devices and configuration files in `/etc`. Simply copy them (devices can be copied with `cp -a`) to the chrooted environment, edit the init script to use chroot before executing. It can be difficult to find what devices and configuration files a services need. This is where the `strace` command becomes handy. Start the service with `/usr/bin/strace` bash and look for open, read, stat and maybe connect. This will give you a clue on what files to copy. But in most cases just copy the passwd file (edit the copy and remove users that has nothing to do with the service), `/dev/zero`, `/dev/log` and `/dev/random`.

11.b. User Mode Linux

Another way of creating a more secure environment is by running a virtual machine. A virtual machine, as the name implies, is a process that runs on top of your real operating system providing a hardware and operating system environment that appears to be its own unique machine. The security benefit is that if the server running on the virtual machine is compromised, only the virtual server is affected and not the parent installation.

For more information about how to setup User Mode Linux consult the User Mode Linux Guide.

12. Firewalls

12.a. A firewall

People often think that a firewall provides the ultimate security, but they are wrong. In most cases a misconfigured firewall gives less security than not having one at all. A firewall is also a piece of software and should be treated the same way as any other piece of software, because it is just as likely to contain bugs.

So think before implementing a firewall! Do you really need one? If you think you need one write a policy on how it should work, what type of firewall, and who should operate it. But first read this guide.

Firewalls are used for two purposes:

- To keep users (worms/attackers) out
- To keep users (employees/children) in

Basically there are three types of firewalls:

- Packet filtering
- Circuit relay
- Application gateway

A firewall should be a dedicated machine running no services (or `sshd` as the only one) and secured the way this guide recommends it be.

12.b. Packet filtering

All network traffic is sent in the form of packets. Large amounts of traffic is split up into small packets for easy handling and then reassembled when it arrives at its destination. In the packet header every packet contains information on how and where it should be delivered. And this information is exactly what a packing filtering firewall uses. Filtering is based on:

- Allow or disallow packets based on source/destination IP address
- Allow or disallow packets based on source/destination port
- Allow or disallow packets based on protocol
- Allow or disallow packets based on flags within a specific protocol

In other words, this filtering is based on all the data within the header of a packet and not its content.

Weaknesses:

- Address information in a packet can potentially be a bogus IP address (or as we say *spoofed* by the sender).
- Data or requests within the allowed packet may contain unwanted data that the attacker can use to exploit known bugs in the services on or behind the firewall
- Usually single point of failure

Advantages:

- Simple and easy to implement
- Can give warnings of a possible attack before it happens (ie. by detecting port scans)
- Good for stopping SYN attacks

Examples of free packet filters on Linux:

- Iptables
- Ipchains
- SmoothWall

Note: It is recommended that you use iptables. Ipchains is obsoleted.

12.c. Circuit relay

A circuit level gateway is a firewall that validates connections before allowing data to be exchanged. This means that it does not simply allow or deny packets based on the packet header but determines whether the connection between both ends is valid according to configurable rules before it opens a session and allows data to be exchanged. Filtering is based on:

- Source/destination IP address
- Source/destination port
- A period of time
- Protocol
- User
- Password

All traffic is validated and monitored, and unwanted traffic can be dropped.

Weakness:

- Operates at the Transport Layer and may require substantial modification of the programs that normally provide transport functions.

12.d. Application gateway

The application level gateway is a proxy for applications, exchanging data with remote systems on behalf of the clients. It is kept away from the public safely behind a DMZ (De-Militarized Zone: the portion of a private network that is visible through the firewall) or a firewall allowing no connections from the outside. Filtering is based on:

- Allow or disallow based on source/destination IP address
- Based on the packet's content
- Limiting file access based on file type or extension

Advantages:

- Can cache files, increasing network performance
- Detailed logging of all connections
- Scales well (some proxy servers can "share" the cached data)
- No direct access from the outside
- Can even alter the packet content on the fly

Weakness:

- Configuration is complex

Application gateways are considered to be the most secure solution since they do not have to run as root and the hosts behind them are not reachable from the Internet.

Example of a free application gateway:

- Squid

12.e. Iptables

In order to use iptables, it must be enabled in the kernel. I have added iptables as modules (the `iptables` command will load them as they are needed) and recompiled my kernel (but you may want to compile iptables in, if you intend to disable Loadable Kernel Modules as discussed previously). For more information on how to configure your kernel for iptables go to the Iptables Tutorial Chapter 5: Preparations. After you have compiled your new kernel (or while compiling the kernel), you must add the `iptables` command. Just `emerge iptables` and it should work.

Now test that it works by running `iptables -L`. If this fails something is wrong and you have to check you configuration once more.

Iptables is the new and heavily improved packet filter in the Linux 2.4.x kernel. It is the successor of the previous ipchains packet filter in the Linux 2.2.x kernel. One of the major improvements is that iptables is able to perform stateful packet filtering. With stateful packet filtering it is possible to keep track of each established TCP connection.

A TCP connection consists of a series of packets containing information about source IP address, destination IP address, source port, destination port, and a sequence number so the packets can be reassembled without losing data. TCP is a connection-oriented protocol, in contrast to UDP, which is connectionless.

By examining the TCP packet header, a stateful packet filter can determine if a received TCP packet is part of an already established connection or not and decide either to accept or drop the packet.

With a stateless packet filter it is possible to fool the packet filter into accepting packets that should be dropped by manipulating the TCP packet headers. This could be done by manipulating the SYN flag or other flags in the TCP header to make a malicious packet appear to be a part of an established connection (since the packet filter itself does not do connection tracking). With stateful packet filtering it is possible to drop such packets, as they are not part of an already established connection. This will also stop the possibility of "stealth scans", a type of port scan in which the scanner sends packets with flags that are far less likely to be logged by a firewall than ordinary SYN packets.

Iptables provides several other features like NAT (Network Address Translation) and rate limiting. Rate limiting is extremely useful when trying to prevent certain DoS (Denial of Service) attacks like SYN floods.

A TCP connection is established by a so called three-way handshake. When establishing a TCP connection the client-side sends a packet to the server with the SYN flag set. When the server-side receives the SYN packet it responds by sending a SYN+ACK packet back to the client-side. When the SYN+ACK is received the client-side responds with a third ACK packet in effect acknowledging the connection.

A SYN flood attack is performed by sending the SYN packet but failing to respond to the SYN+ACK packet. The client-side can forge a packet with a fake source IP address because it does not need a reply. The server-side system will add an entry to a queue of half-open connections when it receives the SYN packet and then wait for the final ACK packet before deleting the entry from the queue. The queue has a limited number of slots and if all the slots are filled it is unable to open any further connections. If the ACK packet is not received before a specified timeout period the entry will automatically be deleted from the queue. The timeout settings vary but will typically be 30-60 seconds or even more. The client-side initiates the attack by forging a lot of SYN packets with different source IP addresses and sends them to the target IP address as fast as possible and thereby filling up the queue of half-open connections and thus preventing other clients from establishing a legitimate connection with the server.

This is where the rate limit becomes handy. It is possible to limit the rate of accepted SYN packets by using the `-m limit --limit 1/s`. This will limit the number of SYN packets accepted to one per second and therefore restricting the SYN flood on our resources.

Note: Another option for preventing SYN floods are SYN cookies, which allow your computer to respond to SYN packets without filling space in the connection queue. SYN cookies can be enabled in the Linux kernel configuration, but they are considered experimental at this time.

Now some practical stuff!

When iptables is loaded in the kernel it has 5 hooks where you can place your rules. They are called `INPUT`, `OUTPUT`, `FORWARD`, `PREROUTING` and `POSTROUTING`. Each of these is called a chain and consists of a list of rules. Each rule says if the packet header looks like this, then here is what to do with the packet. If the rule does not match the packet the

next rule in the chain is consulted.

You can place rules directly in the 5 main chains or create new chains and add them to as a rule to an existing chain. Iptables supports the following options.

Option:	Description:
-A	Append
-D	Delete
-I	Insert
-R	Replace
-L	List
-F	Delete all rules in chain or all chains
-Z	Zero counters in chain or all chains
-C	Test this packet on chain
-N	Create a new user-defined chain
-X	Delete a user-defined chain
-P	Change policy on chain to target
-E	Change chain name
-p	Protocol
-s	Source address/mask
-d	Destination address/mask
-i	Input name (Ethernet name)
-o	Output name (Ethernet name)
-j	Jump (target for rule)
-m	Extended match (might use extension)
-n	Numeric output of addresses and ports
-t	Table to manipulate
-v	Verbose mode
-x	Expand numbers (display exact values)
-f	Match second or further fragments only
-V	Packet version
--line-numbers	Print line numbers when listing

First we will try to block all ICMP packets to our machine, just to get familiar with iptables.

Code Listing 1: Block all ICMP packets

```
# iptables -A INPUT -p icmp -j DROP
```

First we specify the chain our rule should be appended to, then the protocol of the packets to match, and finally the target. The target can be the name of a user specified chain or one of the special targets `ACCEPT`, `DROP`, `REJECT`, `LOG`, `QUEUE`, or `MASQUERADE`. In this case we use `DROP`, which will drop the packet without responding to the client.

Note: The `LOG` target is what's known as "non-terminating". If a packet matches a rule with the `LOG` target, rather than halting evaluation, the packet will continue to be matched to further rules. This allows you to log packets while still processing them normally.

Now try `ping localhost`. You will not get any response, since iptables will drop all incoming ICMP messages. You will also not be able to ping other machines, since the ICMP reply packet will be dropped as well. Now flush the chain to get ICMP flowing again.

Code Listing 2: Flush all rules

```
# iptables -F
```

Now lets look at the stateful packet filtering in iptables. If we wanted to enable stateful inspection of packets incoming on eth0 we would issue the command:

Code Listing 3: Accept packets that originate from an already established connection

```
# iptables -A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

This will accept any packet from an already established connection or related in the INPUT chain. And you could drop any packet that is not in the state table by issuing `iptables -A INPUT -i eth0 -m state --state INVALID -j DROP` just before the previous command. This enables the stateful packet filtering in iptables by loading the extension "state". If you wanted to allow others to connect to your machine, you could use the flag `--state NEW`. Iptables contains some modules for different purposes. Some of them are:

Module/Match	Description	Extended options
mac	**Matching extension for incoming packets mac address.**	**--mac-source**
state	**Enables stateful inspection**	**--state (states are ESTABLISHED,RELATED, INVALID, NEW)**
limit	**Rate matching limiting**	**--limit, --limit-burst**
owner	**Attempt to match various characteristics of the packet creator**	**--uid-owner userid --gid-owner groupid --pid-owner processid --sid-owner sessionid**
unclean	**Various random sanity checks on packets**	

Lets try to create a user-defined chain and apply it to one of the existing chains:

Code Listing 4: Creating a user defined chain

```
(Create a new chain with one rule)
# iptables -X mychain
# iptables -N mychain
# iptables -A mychain -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
(The default policy is all outgoing traffic is allowed. Incoming is dropped.)
# iptables -P OUTPUT ACCEPT
# iptables -P INPUT DROP
(And add it to the INPUT chain)
# iptables -A INPUT -j mychain
```

By applying the rule to the input chain we get the policy: All outgoing packets are allowed and all incoming packets are dropped.

One can find documentation at Netfilter/iptables documentation.

Lets see a full blown example. In this case my firewall/gateway policy states:

- Connections to the firewall are only allowed through SSH (port 22)
- The local network should have access to HTTP, HTTPS and SSH (DNS should also be allowed)
- ICMP traffic can contain payload and should not be allowed. Of course we have to allow some ICMP traffic.
- Port scans should be detected and logged
- SYN attacks should be avoided
- All other traffic should be dropped and logged

Code Listing 5: /etc/init.d/firewall

```
#!/sbin/runscript
IPTABLES=/sbin/iptables
IPTABLESSAVE=/sbin/iptables-save
IPTABLESRESTORE=/sbin/iptables-restore
FIREWALL=/etc/firewall.rules
DNS1=212.242.40.3
DNS2=212.242.40.51
#inside
IIP=10.0.0.2
IINTERFACE=eth0
LOCAL_NETWORK=10.0.0.0/24
#outside
OIP=217.157.156.144
OINTERFACE=eth1

opts="${opts} showstatus panic save restore showoptions rules"

depend() {
  need net
}

rules() {
  stop
  ebegin "Setting internal rules"

  einfo "Setting default rule to drop"
  $IPTABLES -P FORWARD DROP
  $IPTABLES -P INPUT   DROP
  $IPTABLES -P OUTPUT  DROP

  #default rule
  einfo "Creating states chain"
  $IPTABLES -N allowed-connection
  $IPTABLES -F allowed-connection
  $IPTABLES -A allowed-connection -m state --state ESTABLISHED,RELATED -j ACCEPT
  $IPTABLES -A allowed-connection -i $IINTERFACE -m limit -j LOG --log-prefix \
      "Bad packet from ${IINTERFACE}:"
  $IPTABLES -A allowed-connection -j DROP

  #ICMP traffic
  einfo "Creating icmp chain"
  $IPTABLES -N icmp_allowed
  $IPTABLES -F icmp_allowed
  $IPTABLES -A icmp_allowed -m state --state NEW -p icmp --icmp-type \
      time-exceeded -j ACCEPT
  $IPTABLES -A icmp_allowed -m state --state NEW -p icmp --icmp-type \
      destination-unreachable -j ACCEPT
  $IPTABLES -A icmp_allowed -p icmp -j LOG --log-prefix "Bad ICMP traffic:"
  $IPTABLES -A icmp_allowed -p icmp -j DROP

  #Incoming traffic
  einfo "Creating incoming ssh traffic chain"
  $IPTABLES -N allow-ssh-traffic-in
  $IPTABLES -F allow-ssh-traffic-in
  #Flood protection
  $IPTABLES -A allow-ssh-traffic-in -m limit --limit 1/second -p tcp --tcp-flags \
      ALL RST --dport ssh -j ACCEPT
  $IPTABLES -A allow-ssh-traffic-in -m limit --limit 1/second -p tcp --tcp-flags \
      ALL FIN --dport ssh -j ACCEPT
  $IPTABLES -A allow-ssh-traffic-in -m limit --limit 1/second -p tcp --tcp-flags \
      ALL SYN --dport ssh -j ACCEPT
  $IPTABLES -A allow-ssh-traffic-in -m state --state RELATED,ESTABLISHED -p tcp --dport ssh -j ACC

  #outgoing traffic
  einfo "Creating outgoing ssh traffic chain"
  $IPTABLES -N allow-ssh-traffic-out
  $IPTABLES -F allow-ssh-traffic-out
  $IPTABLES -A allow-ssh-traffic-out -p tcp --dport ssh -j ACCEPT

  einfo "Creating outgoing dns traffic chain"
  $IPTABLES -N allow-dns-traffic-out
  $IPTABLES -F allow-dns-traffic-out
  $IPTABLES -A allow-dns-traffic-out -p udp -d $DNS1 --dport domain \
      -j ACCEPT
  $IPTABLES -A allow-dns-traffic-out -p udp -d $DNS2 --dport domain \
     -j ACCEPT

  einfo "Creating outgoing http/https traffic chain"
  $IPTABLES -N allow-www-traffic-out
  $IPTABLES -F allow-www-traffic-out
  $IPTABLES -A allow-www-traffic-out -p tcp --dport www -j ACCEPT
  $IPTABLES -A allow-www-traffic-out -p tcp --dport https -j ACCEPT

  #Catch portscanners
  einfo "Creating portscan detection chain"
  $IPTABLES -N check-flags
```

Some advice when creating a firewall:

1. Create your firewall policy before implementing it
2. Keep it simple
3. Know how each protocol works (read the relevant RFC(Request For Comments))
4. Keep in mind that a firewall is just another piece of software running as root.
5. Test your firewall

If you think that iptables is hard to understand or takes to long to setup a decent firewall you could use Shorewall. It basically uses iptables to generate firewall rules, but concentrates on rules and not specific protocols.

12.f. Squid

Squid is a very powerful proxy server. It can filter traffic based on time, regular expressions on path/URI, source and destination IP addresses, domain, browser, authenticated user name, MIME type, and port number (protocol). I probably forgot some features, but it can be hard to cover the entire list right here.

In the following example I have added a banner filter instead of a filter based on porn sites. The reason for this is that Gentoo.org should *not* be listed as some porn site. And I do not want to waste my time trying to find some good sites for you.

In this case, my policy states:

- Surfing (HTTP/HTTPS) is allowed during work hours (mon-fri 8-17 and sat 8-13), but if employees are here late they should work, not surf
- Downloading files is not allowed (.exe, .com, .arj, .zip, .asf, .avi, .mpg, .mpeg, etc)
- We do not like banners, so they are filtered and replaced with a transparent gif (this is where you get creative!).
- All other connections to and from the Internet are denied.

This is implemented in 4 *easy* steps.

Code Listing 6: /etc/squid/squid.conf

```
# Bind to a ip and port
http_port 10.0.2.1:3128

# Standard configuration
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY

# Add basic access control lists
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255

# Add who can access this proxy server
acl localnet src 10.0.0.0/255.255.0.0

# And ports
acl SSL_ports port 443
acl Safe_ports port 80
acl Safe_ports port 443
acl purge method PURGE

# Add access control list based on regular
# expressions within urls
acl archives urlpath_regex "/etc/squid/files.acl"
acl url_ads url_regex "/etc/squid/banner-ads.acl"

# Add access control list based on time and day
acl restricted_weekdays time MTWHF 8:00-17:00
acl restricted_weekends time A 8:00-13:00

acl CONNECT method CONNECT

#allow manager access from localhost
http_access allow manager localhost
http_access deny manager

# Only allow purge requests from localhost
http_access allow purge localhost
http_access deny purge

# Deny requests to unknown ports
http_access deny !Safe_ports

# Deny CONNECT to other than SSL ports
http_access deny CONNECT !SSL_ports

# My own rules

# Add a page do be displayed when
# a banner is removed
deny_info NOTE_ADS_FILTERED url_ads

# Then deny them
http_access deny url_ads

# Deny all archives
http_access deny archives

# Restrict access to work hours
http_access allow localnet restricted_weekdays
http_access allow localnet restricted_weekends

# Deny the rest
http_access deny all
```

Next fill in the files you do not want your users to download files. I have added zip, viv, exe, mp3, rar, ace, avi, mov, mpg, mpeg, au, ra, arj, tar, gz and z files.

Code Listing 7: /etc/squid/files.acl

```
\.[Zz][Ii][pP]$
\.[Vv][Ii][Vv].*
\.[Ee][Xx][Ee]$
\.[Mm][Pp]3$
\.[Rr][Aa][Rr]$
\.[Aa][Cc][Ee]$
\.[Aa][Ss][Ff]$
\.[Aa][Vv][Ii]$
\.[Mm][Oo][Vv]$
\.[Mm][Pp][Gg]$
\.[Mm][Pp][Ee][Gg]$
\.[Aa][Uu]$
\.[Rr][Aa]$
\.[Aa][Rr][Jj]$
\.[Tt][Aa][Rr]$
\.[Gg][Zz]$
\.[Zz]$
```

Note: Please note the [] with upper and lowercase of every character. This is done so no one can fool our filter by accessing a file called AvI instead of avi.

Next we add the regular expressions for identifying banners. You will probably be a lot more creative than I:

Code Listing 8: /etc/squid/banner-ads.acl

```
/adv/.*\.gif$
/[Aa]ds/.*\.gif$
/[Aa]d[Pp]ix/
/[Aa]d[Ss]erver
/[Aa][Dd]/.*\.[GgJj][IiPp][FfGg]$
/[Bb]annerads/
/adbanner.*\.[GgJj][IiPp][FfGg]$
/images/ad/
/reklame/
/RealMedia/ads/.*
^http://www\.submit-it.*
^http://www\.eads.*
^http://ads\.
^http://ad\.
^http://ads02\.
^http://adaver.*\.
^http://adforce\.
adbot\.com
/ads/.*\.gif.*
_ad\..*cgi
/Banners/
/SmartBanner/
/Ads/Media/Images/
^http://static\.wired\.com/advertising/
^http://*\.dejanews\.com/ads/
^http://adfu\.blockstackers\.com/
^http://ads2\.zdnet\.com/adverts
^http://www2\.burstnet\.com/gifs/
^http://www.\.valueclick\.com/cgi-bin/cycle
^http://www\.altavista\.com/av/gifs/ie_horiz\.gif
```

And as the last part we want this file to be displayed when a banner is removed. It is basically a half html file with a 4x4 transparent gif image.

Code Listing 9: /etc/squid/errors/NOTE_ADS_FILTERED

```
<HTML>
<HEAD>
<META HTTP-EQUIV="REFRESH" CONTENT="0; URL=http://localhost/images/4x4.gif">
<TITLE>ERROR: The requested URL could not be retrieved</TITLE>
</HEAD>
<BODY>
<H1>Add filtered!</H1>
```

Note: Do not close the <HTML> <BODY> tags. This will be done by squid.

As you can see, Squid has a lot of possibilities and it is very effective at both filtering and proxying. It can even use alternative Squid proxies to scale on very large networks. The configuration I have listed here is mostly suited for a small network with 1-20 users.

But combining the packet filter (iptables) and the application gateway (Squid) is probably the best solution, even if Squid is located somewhere safe and nobody can access it from the outside. We still need to be concerned about attacks from the inside.

Now you have to configure your clients browsers to use the proxy server. The gateway will prevent the users from having any contact with the outside unless they use the proxy.

Note: In Mozilla Firefox this is done in Edit->Preferences->Advanced->Network.

It can also be done transparently by using iptables to forward all outbound traffic to a Squid proxy. This can be done by adding a forwarding/prerouting rule on the gateway:

Code Listing 10: Enable portforwarding to our proxyserver

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to proxyhost:3128
# iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT --to proxyhost:3128
```

Note: If the proxy is running on the packet filtering host--though this is not recommended, it may be necessary if you do not have enough spare machines--use a `REDIRECT` target instead of `DNAT` (`REDIRECT` directs packets to the localhost).

12.g. Lessons learned

We have learned that:

1. A firewall can be a risk in itself. A badly configured firewall is worse than not having one at all.
2. How to setup a basic gateway and a transparent proxy.
3. The key to a good firewall is to know the protocols you want do allow.
4. That IP traffic does not always contain legitimate data, e.g. ICMP packets, which can contain a malicious payload.
5. How to prevent SYN attack.
6. Filtering HTTP traffic by removing offensive pictures and downloads of viruses.
7. Combining packet filters and application gateways provides better control.

Now, if you *really* need to, go create a firewall that matches your needs.

13. Intrusion Detection

13.a. AIDE (Advanced Intrusion Detection Environment)

AIDE is a Host-Based Intrusion Detection System (HIDS), a free alternative to Tripwire (if you already know Tripwire you should have no difficulties learning the configuration file for AIDE). HIDS are used to detect changes to important system configuration files and binaries, generally by making a unique cryptographic hash for the files to be checked and storing it in a secure place. On a regular basis (such as once a day), the stored "known-good" hash is compared to the one generated from the current copy of each file, to determine if that file has changed. HIDS are a great way to detect disallowed changes to your system, but they take a little work to implement properly and make good use of.

The configuration file is based on regular expressions, macros and rules for files and directories. We have the following macros:

Macro	Description	Syntax
ifdef	If definded	@@ifdef "name"
ifndef	If not defined	@@ifndef "name"
define	Define a variable	@@define "name" "value"
undef	Undefine a variable	@@undef "name"
ifhost	if "hostname"	@@ifhost "hostname"
ifnhost	if not "hostname"	@@ifnhost "hostname"
endif	Endif must be used after any of the above macros except define and undef	@@endif

These macros become very handy if you have more than one Gentoo box and want to use AIDE on all of them. But not all machines run the same services or even have the same users.

Next we have sets of flags to check for on files and directories. These are a combination of permissions, file properties

and cryptographic hashes (i.e. checksums).

Flag	Description
p	permissions
i	inode
n	number of links
u	user
g	group
s	size
b	block count
m	mtime
a	atime
c	ctime
S	check for growing size
md5	md5 checksum
sha1	sha1 checksum
rmd160	rmd160 checksum
tiger	tiger checksum
R	p+i+n+u+g+s+m+c+md5
L	p+i+n+u+g
E	Empty group
>	Growing logfile p+u+g+i+n+S

And if AIDE is compiled with mhash support it supports a few other features:

Flag	Description
haval	haval checksum
gost	gost checksum
crc32	crc32 checksum

Now you can create you own rules based on the above flags by combining them like this:

Code Listing 1: Create a ruleset for AIDE

```
All=R+a+sha1+rmd160
Norm=s+n+b+md5+sha1+rmd160
```

The last thing we need to create our own configuration file is to see how to add a rule to a file or directory. To enter a rule, combine the file or directory name and the rule. AIDE will add all files recursively unless you specify an alternate rule.

Flag	Description
!	Don't add this file or directory.
=	Add this directory, but not recursively.

So lets watch a full blown example:

Code Listing 2: /etc/aide/aide.conf

```
@@ifndef TOPDIR
@@define TOPDIR /
@@endif

@@ifndef AIDEDIR
@@define AIDEDIR /etc/aide
@@endif

@@ifhost smbserv
@@define smbactive
@@endif

# The location of the database to be read.
database=file:@@{AIDEDIR}/aide.db

# The location of the database to be written.
database_out=file:aide.db.new

verbose=20
report_url=stdout

# Rule definition
All=R+a+sha1+rmd160
Norm=s+n+b+md5+sha1+rmd160

@@{TOPDIR} Norm
!@@{TOPDIR}etc/aide
!@@{TOPDIR}dev
!@@{TOPDIR}media
!@@{TOPDIR}mnt
!@@{TOPDIR}proc
!@@{TOPDIR}root
!@@{TOPDIR}sys
!@@{TOPDIR}tmp
!@@{TOPDIR}var/log
!@@{TOPDIR}var/run
!@@{TOPDIR}usr/portage
@@ifdef smbactive
!@@{TOPDIR}etc/smb/private/secrets.tdb
@@endif
=@@{TOPDIR}home Norm
```

In the above example we specify with some macros where the topdir starts and where the AIDE directory is. AIDE checks the `/etc/aide/aide.db` file when checking for file integrity. But when updating or creating a new file it stores the information in `/etc/aide/aide.db.new`. This is done so it won't automatically overwrite the old db file. The option `report_URL` is not yet implemented, but the author's intention was that it should be able to e-mail or maybe even execute scripts.

The AIDE ebuild now comes with a working default configuration file, a helper script and a crontab script. The helper script does a number of tasks for you and provides an interface that is a little more script friendly. To see all available options, try `aide --help`. To get started, all that needs to be done is `aide -i` and the crontab script should detect the database and send mails as appropriate every day. We recommend that you review the `/etc/aide/aide.conf` file and ensure that the configuration accurately reflects what is in place on the machine.

Note: Depending on your CPU, disk access speed, and the flags you have set on files, this can take some time.

Note: Remember to set an alias so you get roots mail. Otherwise you will never know what AIDE reports.

Now there is some risk inherent with storing the db files locally, since the attacker will (if they know that AIDE is installed) most certainly try to alter the db file, update the db file or modify `/usr/bin/aide`. So you should create a CD or other media and put on it a copy of the .db file and the AIDE binaries.

One can find information at the AIDE project page.

13.b. Snort

Snort is a Network Intrusion Detection System (NIDS). To install and configure it use the following examples.

Code Listing 3: /etc/conf.d/snort

```
PIDFILE=/var/run/snort_eth0.pid
MODE="full"
NETWORK="10.0.0.0/24"
LOGDIR="/var/log/snort"
CONF=/etc/snort/snort.conf
SNORT_OPTS="-D -s -u snort -dev -l $LOGDIR -h $NETWORK -c $CONF"
```

Code Listing 4: /etc/snort/snort.conf

```
(Step 1)
var HOME_NET 10.0.0.0/24
var EXTERNAL_NET any
var SMTP $HOME_NET
var HTTP_SERVERS $HOME_NET
var SQL_SERVERS $HOME_NET
var DNS_SERVERS [10.0.0.2/32,212.242.40.51/32]
var RULE_PATH ./

(Step 2)
preprocessor frag2
preprocessor stream4: detect_scans detect_state_problems detect_scans disable_evasion_alerts
preprocessor stream4_reassemble: ports all
preprocessor http_decode: 80 8080 unicode iis_alt_unicode double_encode iis_flip_slash full_whites
preprocessor rpc_decode: 111 32771
preprocessor bo: -nobrute
preprocessor telnet_decode

(Step 3)
include classification.config

(Step 4)
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/smtp.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/shellcode.rules
include $RULE_PATH/policy.rules
include $RULE_PATH/porn.rules
include $RULE_PATH/info.rules
include $RULE_PATH/icmp-info.rules
include $RULE_PATH/virus.rules
# include $RULE_PATH/experimental.rules
include $RULE_PATH/local.rules
```

Code Listing 5: /etc/snort/classification.config

```
config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1

# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
config classification: suspicious-login,An attempted login using a suspicious username was detecte
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web application
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: kickass-porn,SCORE! Get the lotion!,1
```

More information is at the Snort web site.

13.c. Detecting malware with chkrootkit

HIDS like AIDE are a great way to detect changes to your system, but it never hurts to have another line of defence. `chkrootkit` is a utility that scans common system files for the presence of rootkits--software designed to hide an intruder's actions and allow him to retain his access--and scans your system for likely traces of key loggers and other "malware". While `chkrootkit` (and alternatives like `rkhunter`) are useful tools, both for system maintenance and for tracking an intruder after an attack has occurred, they cannot guarantee your system is secure.

The best way to use `chkrootkit` to detect an intrusion is to run it routinely from `cron`. To start, emerge `app-admin/chkrootkit`. `chkrootkit` can be run from the command line by the command of the same name, or from `cron` with an entry such as this:

Code Listing 6: Schedule chkrootkit as a cronjob

```
0 3 * * * /usr/sbin/chkrootkit
```

14. Keeping up-to-date

14.a. Keeping up-to-date

Once you have successfully installed your system and ensured a good level of security you are not done. Security is an ongoing process; the vast majority of intrusions result from known vulnerabilities in unpatched systems. Keeping your system up-to-date is the single most valuable step you can take to greater security.

If you have a recent version of `portage` installed, you can first sync your portage tree with `emerge --sync` and then issue the command `glsa-check --list` to check if your system is up to date security-wise. `glsa-check` is part of `app-portage/gentoolkit`.

Code Listing 1: Example output of glsa-check -l

```
# glsa-check -l
WARNING: This tool is completely new and not very tested, so it should not be
used on production systems. It's mainly a test tool for the new GLSA release
and distribution system, it's functionality will later be merged into emerge
and equery.
Please read http://www.gentoo.org/proj/en/portage/glsa-integration.xml
before using this tool AND before reporting a bug.

[A] means this GLSA was already applied,
[U] means the system is not affected and
[N] indicates that the system might be affected.

200406-03 [N] sitecopy: Multiple vulnerabilities in included libneon ( net-misc/sitecopy )
200406-04 [U] Mailman: Member password disclosure vulnerability ( net-mail/mailman )
.......
```

Warning: The `glsa-check` is still experimental, so if security really is your top priority it would be wise to double check the list with other sources.

All lines with a `[A]` and `[U]` can be almost safely ignored as the system is not affected by this GLSA.

Important: Please note that the usual `emerge -vpuD world` will not pick up all package updates. You need to use `glsa-check` if you want to make sure all GLSAs are fixed on your system.

Code Listing 2: Check all GLSAs

```
(Check if your system is affected by GLSAs)
# glsa-check -t all
WARNING: This tool is completely new and not very tested, so it should not be
used on production systems. It's mainly a test tool for the new GLSA release
and distribution system, it's functionality will later be merged into emerge
and equery.
Please read http://www.gentoo.org/proj/en/portage/glsa-integration.xml
before using this tool AND before reporting a bug.

This system is affected by the following GLSA:
200504-06
200510-08
200506-14
200501-35
200508-12
200507-16

(See what packages would be emerged)
# glsa-check -p $(glsa-check -t all)
     (partial output)
Checking GLSA 200504-06
The following updates will be performed for this GLSA:
     app-arch/sharutils-4.2.1-r11 (4.2.1-r10)

     ************************************************************************

     Checking GLSA 200510-08
     The following updates will be performed for this GLSA:
          media-libs/xine-lib-1.1.0-r5 (1.1.0-r4)

(Apply required fixes)
# glsa-check -f $(glsa-check -t all)
```

If you have upgraded a running service, you should not forget to restart it.

Keeping your kernel up-to-date is also recommended.

If you want an email each time a GLSA is released subscribe to the `gentoo-announce` mailing list. Instructions for joining it and many other great mailing lists can be found Gentoo Linux Mailing List Overview.

Another great security resource is the Bugtraq mailing list.