23 November 2006

Unix Security — An Introduction

This document is intended to:

1. offer an introduction to security, mainly for Unix-like machines;
2. *outline*, for system administrators, "recipes" where appropriate.

src : `unix_security_intro.tex`, from the SGML.

Document History

1. First release: 2002 August 22, Simon Hood.
2. Still being updated...: 2003 August 20, Simon Hood.
3. On going...: 2004 July 07, Simon Hood.
4. Added some stuff on scanners and lsof: 2004 October, Simon Hood.

Feedback

If you think I've missed something, something is unclear or simply wrong, please email `simon.hood@umist.ac.uk`.

1. More

- linuxsecure.de[1]
- [2]
- [3]

2. Don’t Run Services You Don’t Need

Often machines are compromised through services that they have no need to run. The following are common services that can usually be stopped:

- finger (usually `/usr/sbin/in.fingerd`);
- ftp (`/usr/sbin/in.ftpd`);
- rexec, rlogin and rsh (`/usr/sbin/in.rexecd`, `in.rlogind` and `in.rshd`);
- talk (`/usr/sbin/in.talkd`);

To determine which services you are running — more accurately, which are listening on a TCP or UDP port, try this

```
netstat -a | grep LIST
```

which will give output something like this:

```
tcp        0      0 *:rsync              *:*               LISTEN
tcp        0      0 *:ftp                *:*               LISTEN
tcp        0      0 *:smtp               *:*               LISTEN
tcp        0      0 *:2306               *:*               LISTEN
tcp        0      0 *:www                *:*               LISTEN
tcp        0      0 *:ssh                *:*               LISTEN
```

2.1. How do I stop them?

This is easy. For many services, simply comment out the pertinent line in `/etc/inetd.conf` (for older Linux distributions and Solaris) or edit the appropriate file in `/etc/xinetd.d` (in recent Linux distributions); in other cases you will need to make simple changes to the `init` scripts on the machine.

[1] `http://www.linuxsecure.de`
[2]
[3]

2.1.1. Removing Services from inetd/xinetd

On a Solaris 7 or 8 machine (or older distribution of Linux), simply comment out the relevant lines in /etc/inetd.conf, for example — lines beginning with a # are comments —

```
#shell   stream  tcp    nowait  root   /usr/sbin/in.rshd       in.rshd
#login   stream  tcp6   nowait  root   /usr/sbin/in.rlogind    in.rlogind
#exec    stream  tcp    nowait  root   /usr/sbin/in.rexecd     in.rexecd

#talk    dgram   udp    wait    root   /usr/sbin/in.talkd      in.talkd

#ftp     stream  tcp    nowait  root   /usr/sbin/in.ftpd       in.ftpd
#telnet  stream  tcp    nowait  root   /usr/sbin/in.telnetd    in.telnetd
```

and then restart the inetd daemon

```
prompt]$ kill -HUP <PID>
```

To determine the PID on Solaris use ps -e,

```
prompt]$ ps -e | grep inetd
171 ?       0:00 inetd
```

giving, in this case, 171, and on Linux,

```
prompt]$ ps ax | grep inetd
904  ?        S      0:00 inetd
6713 pts/0    S      0:00 grep inetd
```

giving, in this case, 904.

N.B. These changes will have no effect on *clients* — you will still be able to connect to a remote machine via telnet or FTP.

On more recent Linux distributions inetd has been replaced with xinetd. Each service has a corresponding xinetd configuration file within /etc/xinetd.d, for example, /etc/xinetd.d/telnet. To stop a given service, edit the corresponding file and change

```
disable         = no
```

to

```
disable         = yes
```

then restart xinetd:

```
/etc/init.d/xinetd restart
```

2.1.2. Changing `init` Scripts

Some services/daemons are started via scripts which live in `/etc/rc.d/init.d` (or `/etc/init.d`), or more precisely via links to scripts within this directory from one of `/etc/rc[1-6].d`, the directory name corresponding to the current runlevel. (RedHat linux usually runs at level 3 or 5 — 3 for command-line logins and 5 for graphical logins; Solaris usually runs at level 2 or 3 — 2 is multi-user state and 3 is extended multi-user state.)

For example, on a RedHat box one might find the following files:

```
/etc/init.d/sendmail
/etc/rc0.d/K30sendmail
/etc/rc1.d/K30sendmail
/etc/rc2.d/S80sendmail
/etc/rc3.d/S80sendmail
/etc/rc4.d/S80sendmail
/etc/rc5.d/S80sendmail
/etc/rc6.d/S80sendmail
```

Such init scripts can be called upon to start and stop services — roughly, `S` means `sendmail start` and `K` means `sendmail stop`. To prevent `sendmail` starting (as a daemon) on future boots simply remove the `S80sendmail` links from appropriate directories/runlevels. To stop the current `sendmail` service/daemon type `/etc/init.d/sendmail stop`.

2.1.3. Determining the Daemon Corresponding to a Given Port

It is not always obvious which daemon or process is responsible for listening on a particular port. For example, suppose that `netstat -a | grep LIST` gives

```
  *.22                 *.*                 0      0     0      0 LISTEN
  *.32797              *.*                 0      0     0      0 LISTEN
```

and only the first (SSH) is wanted. The solution is to use `lsof`:

```
prompt> lsof | grep 32797
mountd    525    root   8u  inet 0x30000721a68   0t0   TCP *:32797 (LISTEN)
```

2.2. Common Intruder Targets

ftp Servers

FTP servers have a notorious history and as such are a common target for would-be intruders:

- Rather than risk running your own FTP server a safer option might be to make use someone elses — which is maintained by experts with time to spend.
- Use `scp` or `sftp` instead.
- Wrap [Page 12] or firewall [Page 27] the service so that only a small number of hosts have access to the FTP server.
- Use an FTP proxy — this increases your protection against buffer overflows and also allows you to restrict which FTP commands are executed by clients (see Linux Journal Issue 104/December 2002, Paranoid Penguind by Mick Bauer).

`sendmail`

At the time of writing, `sendmail` has had few recent security-related problems and patches, but its history is not so good — it's had more than its fair share of holes (e.g., those due to buffer overflows). In addition, `sendmail` is notoriously difficult to configure, compounding other problems.

- First, most hosts do not need to receive mail from remote machines (clients mostly connect to remote IMAP servers these days) and so `sendmail` need not be run in daemon mode (need not be started via the init scripts and need not listen — usually on port 25).
- If `sendmail` is needed, it is usually only for sending and receiving mail locally, for example, to see the output of `cron` scripts. In this case `sendmail` can be configured to listen to `127.0.0.1` only — in fact this is part of the standard configuration now, viz this extract from `/etc/mail/sendmail.mc`:

  ```
  dnl # The following causes sendmail to only listen on the IPv4
  dnl # loopback address 127.0.0.1 and not on any other network devices.
  dnl # Remove the loopback address restriction to accept email from the
  dnl # internet or intranet.
  DAEMON_OPTIONS(‘Port=smtp,Addr=127.0.0.1, Name=MTA’)dnl
  ```

- As for all services wrap [Page 12] or firewall [Page 27] the appropriate port so that only a small number of hosts have access.
- Finally, the Sendmail Restricted Shell, `smrsh`, can be used. It is intended as a replacement for `sh` for use in the "prog" mailer in `sendmail` configuration files. It sharply limits the commands that can be run using the `|program` syntax thus improving the security of your systemm. (If an intruder can get `sendmail` to run a program without going through an alias or forward file, `smrsh` limits the set of programs that he or she can execute.)

YP/NIS

Quoting from *Building Internet Firewalls*, O'Reilly:

NIS/YP is designed to provide distributed access to centralized administrative information (such as host tables, password files, site-wide email aliases, and so on) shared by machines at a site.

The main problem with NIS/YP is that its security isn't good enough to adequately protect some of the data it contains. In particular, a site's NIS/YP servers generally contain the shared password file (equivalent to the /etc/passwd file on a single system) for the site, complete with encrypted passwords. All you need to get data from an NIS/YP server is the NIS/YP domain name with which the data is associated. An attacker who can talk to a site's NIS/YP server, and who can guess what the site has chosen as its NIS/YP domain name (often the same as, or a derivative of, their regular Internet domain name), can request any information the server has. If the attacker gets their shared password file, these passwords can be cracked at the attacker's leisure.

NOTE: NIS/YP transfers include the encrypted passwords even if the machines are configured to use shadow passwords and the encrypted passwords are not readable on the NIS/YP server.

A few NIS/YP servers (notably Sun's) support a configuration file called securenets. This allows you to use IP address authentication to control which hosts your NIS/YP server will release data to.

...end quote. To see the kind of data which is being referred to above, on your host type

```
ypcat passwd.adjunct.byname
```

(the map name may vary) and you will see the encrypted passwords for each user.

- Use NIS+ instead of NIS/YP (though *many* people don't like NIS+ and it has never really caught on.
- Block access to your NI/YP server from any host that does not need access via a packet filter.
- If central administration of accounts is required, authenticate via an LDAP server, or alternative, so that passwords simply are not stored on the NIS/YP machine.

3. Secure X Windows

3.1. `xhost`

Use the `xhost` utility properly! i.e., switch a access restrictions and allow only known and trusted hosts to connect. For example:

```
xhost -
xhost +trusted_friend.dom.net
xhost +good_colleague.domain.org
```

Never, ever, type `xhost +`. Ever.

3.2. `-nolisten tcp`

Modern X-servers have the `-nolisten <proto>` option, including that from XFree*6, Xorg and Solaris 9 and above — not Solaris 8 and below. See the man page on either for details (`man Xserver`, not `man X`). If you are starting the X server manually (unlikely),

```
/usr/bin/X11/X -nolisten tcp
```

or (more likely) either

```
xinit -- -nolisten tcp     # ...note the "--"
```

or

```
startx -- -nolisten tcp     # ...note the "--"
```

The latter is a commonly used script for calling `xinit` with a variable called `serverargs`, or similar; set this to include the `-nolisten tcp` option.

4. Patch Your System

Apply available patches! Patches are made available for download by every(?) vendor, including RedHat, SuSE and Sun, and also by every reputable Linux and BSD distro. Apply them, particularly those which relate to remotely-exploitable vulnerabilities. You know it's worth it!

In most cases tools are available which will handle dependency-checking and which can be invoked by `cron` (daily).

5. Patch Your System: Linux

Most Linux distributions are RPM-based or Debian-based. `rpm` itself can be used for patching but is not good — better to use `yum` or the RPM version of `apt-get`. For Debian-based distros, simply use `apt-get`.

5.1. `rpm`-based Distros

A patch/update for a part RPM can be applied in the usual way, for example,

```
rpm -Uvh apache-1.3.22-6.i386.rpm
```

or

```
rpm -Uvh openssh*
```

where, in the latter case, there are 5 separate RPMs to upgrade.

A simple procedure to ensure all patches have been applied is to download all updated RPMs from the appropriate Web site, e.g. from RedHat errata[4], then *freshen* (see `rpm` `man`-page):

```
rpm -Fvh ./*.rpm
```

Only those RPMs which are already installed will be updated; others will be ignored.

Frankly, though, `rpm` is a crap way to patch anything (or even install a new package) since it does not handle dependency-checking. Better to use `yum` or the RPM version of `apt-get`.

5.2. RedHat/Fedora

Patches for RedHat can be found at RedHat's[5] Web site. (These patches are complete, replacement `rpms`.) The RedHat Update service (`up2date`) can be used to automate the patching process, but requires registration — other solutions exist which do not require this. (Again, `yum` and `apt-get`.)

5.3. SuSE: YaST[2]

SuSE is an RPM-based distribution, but handles patching properly via a proper patching system (contrast RedHat who simply supply updated complete packages/`rpms`). The patches can be downloaded and installed by use of the SuSE setup/config tool, `yast2`. An option is available within `yast2` to download and install patches, whenever they become available, automatically.

[4] `http://www.redhat.com/apps/support/errata/`
[5] `http://www.redhat.com/apps/support/errata/`

5.4. `apt-get`

Debian and Debian-based distributions have had `apt-get` and friends for years. The `apt` commands download and install packages *and handle dependencies* automatically:

`apt-get update`

Download package lists. Do this often.

`apt-get dist-upgrade`

Update all packages.

`apt-get install <package(s)>`

Install/upgrade specific package(s) (and its dependencies, if missing any).

`apt-cache search <word>`

Search all known packages entries (descriptions etc.) for `word`.

`apt-cache show <package>`

Show basic information about a package.

See `man apt-get` and `man apt-cache` for more details.

It is easy to call `apt-get update` and `apt-get upgrade` from `cron` (daily). There is a GUI frontend for `apt` called Synaptic (`apt-get install synaptic`).

5.5. `yum`

From the Web site[6]: Yum is an automatic updater and package installer/remover for RPM systems. It automatically computes dependencies and figures out what things should occur to install packages. It makes it easier to maintain groups of machines without having to manually update each one using `rpm`.

[6] `http://linux.duke.edu/projects/yum/`

Basic usage:

`yum list`

List all available packages.

`yum check-update` **or** `yum list updates`

See if there are updated packages available.

`yum update`

Update all installed packages that have a newer version available.

`yum install <package(s)>`

Install specific packages and their dependencies.

`yum search <word>`

Search all known packages entries (descriptions etc) for word.

`yum info <package>`

Show basic information about a package.

6. Patch Your System: Solaris

Patches for Solaris can be found at `sunsolve.sun.com`[7] (follow the links to *Product Patches* and then *Recommended and Security Patches*). The usual route is: download the latest patch-cluster containing all recommended and security patches, `unzip` the `.zip` file; `cd` into the created directory and run the installation script, `./install_cluster` — this may take a while. *Then reboot* — if you do not the system may be left in an inconsistent state and not all patches will have taken effect.

The Solaris Patch Manager, downloadable from sunsolve.sun.com[8] helps automate the process.

[7] `http://sunsolve.sun.com/`
[8] `https://sunsolve.sun.com/patchpro/patchpro.html`

Example

```
df -bk                  # Check there is sufficient space in /, /usr and
                        # /opt for the patches to be applied, and in /var
                        # for storage of old stuff so that patches can
                        # be rolled back if necessary.

ftp ftp.sunsolve.com    # Get the latest patch recommended cluster.
cd /pub/patches
bin
get 9_Recommended.zip
quit

unzip 9_Recommended.zip # Unpack/uncompress the thing and move into the
cd 9_Recommended        # newly-created directory.

su nobody               # The user "nobody" needs read access to the
ls -l                   # new stuff for the patching to proceed --- check.
exit

./install_cluster       # Or "./install_cluster -nosave" if space is
                        # lacking under /var.
```

7. Run Secure Services: `sshd`, `ssh` and Friends

Most of the UMIST network is switched; consequently sending passwords (and other sensitive data) over the network in clear text (unencrypted) should not be a problem — though there are some hubs, so this is not true everywhere.

However, sending unencryted passwords over the Internet is in general a bad idea — both telnet and FTP do precisely this. Users who dial-up to UMIST or otherwise access remote machines should use protocols which offer encryption; these include SSH, SCP and SFTP. To use these protocols you'll need a suitable client, on your local machine and a suitable daemon/server to be running on the remote machine.

Cosmos (`cosmos.umist.ac.uk`) has a SSH daemon running so that users with an ISD Unix account can securely access this machine.

7.1. `ssh` Clients

`ssh` clients are freely available for download and installation for both Linux and Solaris. (In fact most distributions of Linux come with an `ssh` client ready-installed.) The two most popular clients are that from the Open SSH organisation and from SSH Communications.

The OpenSSH client is available for free download from `www.openssh.org`[9] as both source (to be compiled) and Linux RPM binaries. Solaris binaries may be downloaded from `sunfreeware.com`[10].

[9] `http://www.openssh.org/`
[10] `http://sunfreeware.com`

The SSH Communications client for Unix is available for free download (for academic and other non-commercial uses) from `www.ssh.com`[11] in source form, for compilation and installation. An MS Windows client, in binary format, is also available.

PuTTY is a second freely-available SSH client for MS Windows and is freely-available from `www.chiark.greenend.org.uk`[12].

7.2. Accessing Cosmos via `ssh`

To access Cosmos via SSH you will need a client which supports the SSH2 protocol and moreover understands *keyboard-interactive* authentication, e.g., PuTTY 0.52 or above. For more details see the Cosmos documentation.

7.3. `ssh` Daemons

SSH daemons for Unix machines are available from both the OpenSSH organisation (free) and SSH Communications (free for non-commercial use). Most Linux distributions come with the OpenSSH implementation already installed — binaries will be available from your usual mirror site; Solaris binaries from `sunfreeware.com`[13]; and the OpenSSH source code is available from `www.openssh.org`[14].

7.4. Security Holes in the SSH Deamon: OpenSSH vs SSH Communications Security

As of the time of writing the OpenSSH daemon has been beset with problems: a sequence of security-holes have been found. These have each been fixed — *ensure you have the very latest daemon version installed* if you use the OpenSSH daemon — but clearly the code is less mature than that from SSH Communications. Currently ISD recommends that you use the daemon from SSH Communications.

8. Wrap Your Services: `inetd` and `xinetd`

Most services on a machine to which one can connect remotely may be “wrapped”: when a connection is made, a security service looks at it, and if it satisfies given criteria the connection is passed on to the OS service in question, such as telnet or FTP; if not the connection is dropped. The criteria usually consist of access control lists — does the connection come from a known, trusted host or domain? Such a security service can come in the form of a *super-daemon* (possibly with auxilliary software), e.g., `inetd` (with TCP Wrappers) or `xinetd`, or in the form of a *proxy*.

[11] `http://www.ssh.com`
[12] `http://www.chiark.greenend.org.uk/~sgtatham/putty/`
[13] `http://sunfreeware.com`
[14] `http://www.openssh.org`

8.1. TCP Wrappers

On older Linuces and on Solaris, service wrapping is usually done with TCP Wrappers. The source can be freely downloaded from `porcupine.org`[15]; binaries are available from `sunfreeware.com`[16] for Solaris and with most older Linux distributions.

After compilation and linking, if necessary, there are two ways in which installation and configuration can be completed:

1. Move network daemons to some other directory and fill the resulting holes with copies of the wrapper programs. This approach involves no changes to system configuration files. (See the supplied `README` for more details.)

2. Install the wrapper daemon, `tcpd`, (e.g., in `/usr/sbin`); leave the network daemons alone and modify the inetd configuration file, `/etc/inetd.conf`: for example, commented out unwrapped services and the corresponding wrapped services might look like this:

   ```
   ## ftp     stream  tcp6  nowait  root  /usr/sbin/in.ftpd     in.ftpd
   ## telnet  stream  tcp6  nowait  root  /usr/sbin/in.telnetd  in.telnetd
   #
   ftp     stream  tcp  nowait  root  /usr/sbin/tcpd  in.ftpd
   telnet  stream  tcp  nowait  root  /usr/sbin/tcpd  in.telnetd
   ```

   Then restart `inetd` as described above.

Your binary package may lead you in one direction or the other.

Finally, the associated access-control lists, `/etc/hosts.allow` and `/etc/hosts.deny`, must be edited, for example:

```
# -- /etc/hosts.deny --- a default-deny stance :
in.telnetd: ALL
in.ftpd: ALL
in.rexecd: ALL
in.rlogind: ALL
in.rshd: ALL
```

and

```
# -- /etc/hosts.allow --- let a few friends in :
in.telnetd: myhost.umist.ac.uk, friend.umist.ac.uk
in.ftpd: friend.umist.ac.uk
in.rexecd: friend.umist.ac.uk, collegue.dept.umist.ac.uk
```

This should be simple, but be warned, some versions of TCP Wrappers can be fussy over details of whitespace and comments within the configuration files — use the utility `tcpdchk` (which comes with TCP Wrappers) to check the files' syntax.

[15] `http://www.porcupine.org/`
[16] `http://sunfreeware.com/`

8.2. xinetd

On recent Linux distributions `xinetd` access control lists have replaced TCP Wrappers. (The required daemons and configuration files should already be installed.) The configuration files are to be found in `/etc/xinetd.d`.

Example Configuration Files

After changing configuration files restart `xinetd` as described above.

A standard `xinetd` configuration file which can be trivially-modified for FTP and other services — note the use of Cosmos as a gateway machine so that this machine can be accessed globally:

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
#              unencrypted username/password pairs for authentication.
service telnet
{
    flags            = REUSE
    socket_type      = stream
    wait             = no
    user             = root
    server           = /usr/sbin/in.telnetd
    log_on_failure  += USERID
    disable          = no


    only_from        = 192.168.1.3      # my PC
    only_from       += 192.168.1.123    # my trusted friend's machine
    only_from       += 192.168.1.67     # my co-researcher's machine
    only_from       += 130.88.99.10     # cosmos --- use as
                                        #     globally-accessible gateway
}
```

If you have a need to run a SMTP service, wrap it like this — note the server arguments:

```
    service smtp
      {
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = root
        server          = /usr/lib/sendmail
        server_args     = -bs
  #NO!# server_args     = -bd -q15m
        disable         = no
        instances       = 10
        nice            = 10
        log_on_failure += HOST
        only_from       = 127.0.0.1
        only_from      += 130.88.99.10   # cosmos --- for testing
        #
        only_from      += 130.88.119.65  # rainstorm  }
        only_from      += 130.88.120.65  # downpour   } the ISD mail
        only_from      += 130.88.119.66  # cloudburst } routers
        only_from      += 130.88.120.66  # deluge     }
        no_access       = 0.0.0.0
      }
```

9. `chroot`ing

9.1. What is a `chroot` jail?

To minimise the consequences of compromise of a daemon, run publicly-accessible services within a "sandpit". The simplest way to do this is to make use of the `chroot` command, which wraps the `chroot()` system call: `chroot()` changes the root directory of a process (`/proc/<procno>/root`). Should an intruder manage to gain a shell on a host via a `choot`ed daemon, they will have to break out of the "jail" too in order to set up their own services or cause (other) system damage.

9.2. How to `chroot` a daemon — outline

Recipes exist for some daemons/services, notably Apache and Bind, but for most it's a case of following the following guidelines and being prepared for a little trial and error:

- use `ldd` to determine which shared-object libraries are used by a daemon (and since `.so`s can depend on others, recursive use may be needed);
- determine what other files are opened by the process(es) involved by using `strace` and perhaps `ltrace` (e.g., Linux), or `struss` (e.g., Solaris) —
- `lsof` might also prove useful;
- finally, `strace` or `struss` can be used to check for other problems, for example,

```
strace squid >& /tmp/squid.strace
fgrep "Err NOENT" /tmp/squid.strace
```

If you follow a `chroot` HOWTO (e.g. that the widely-available ones for BIND and Apache, or that for Squid below) and it doesn't work for you, judicious use of `strace` (and perhaps `fgrep`, as above) will likely be your saviour.

9.3. An Example — `chroot`ing The Squid Proxy (with PAM Authentication and `syslogd`)

Here we outline how to `chroot` the Squid Web proxy. (Squid actually contains a configuration option to `chroot` itself upon startup. Nevertheless, this example well illustrates the principles involved.)

N.B. *These notes were written from memory, after* `chroot`*ing Squid and have not been tested (by building a second jail from scratch by following these notes). So, corrections welcome.*

1. First, make the `chroot` directory, e.g.,

   ```
   prompt> mkdir /chroot_squid
   ```

   and the directories below this (`bin`, `dev`, `etc`, `lib`, `usr`, `usr/local`...).

2. Build and install Squid in the usual place, e.g., `/usr/local/squid`.

3. Determine which shared-object libraries `ldd` indicates are needed

   ```
   ldd /usr/local/squid/sbin/squid

   libcrypt.so.1 => /lib/libcrypt.so.1
   libpthread.so.0 => /lib/libpthread.so.0
   libm.so.6 => /lib/libm.so.6
   libresolv.so.2 => /lib/libresolv.so.2
   libnsl.so.1 => /lib/libnsl.so.1
   libc.so.6 => /lib/libc.so.6
   /lib/ld-linux.so.2 => /lib/ld-linux.so.2
   ```

   and copy these into the `/chroot_squid` tree, i.e., to `/chroot_squid/lib`.

In simple cases of `chroot`ing — if you are lucky — this will be sufficient; mostly it is not. In this case

```
chroot /chroot_squid /usr/local/squid/sbin/squid
```

will fail.

4. Run

   ```
   strace -o /tmp/squid.strace -v -f -r -e trace=open -tt \
                                            /usr/local/squid/sbin/squid
   ```

   and, after a few seconds, kill the `strace`/`squid` process and examine the contents of `/tmp/squid.strace`, e.g.,

   ```
   fgrep \.so /tmp/squid.strace
   ```

which gives us this list:

```
/lib/libcrypt.so.1
/lib/libpthread.so.0
/lib/libm.so.6
/lib/libresolv.so.2
/lib/libnsl.so.1
/lib/libc.so.6
/lib/libnss_files.so.2
/lib/libnss_compat.so.2
/lib/libnss_nis.so.2
/lib/libnsl.so.1
/lib/libnss_dns.so.2
```

Copy the extra libraries from `/lib` to `/chroot_squid/lib`.

5. Next up, Squid can be configured to use PAM authentication — here is the pertinent line from `squid.conf`:

```
auth_param basic program /usr/local/squid/libexec/pam_auth
```

and `ldd /usr/local/squid/libexec/pam_auth` gives some further libraries to copy to our `chroot_squid` tree:

```
libpam.so.0 => /lib/libpam.so.0
libdl.so.2 => /lib/tls/libdl.so.2
```

Supporting PAM configuration and libraries must also be installed:

```
/chroot_squid/etc/pam.d/squid

/chroot_squid/lib/security/pam_ldap.so
/chroot_squid/lib/security/pam_unix_auth.so
```

6. Ok, so now we're feeling confident, so we try to start Squid,

```
chroot /chroot_squid /usr/local/squid/sbin/squid
```

but again it fails. It's time for a brute force:

```
strace -o /tmp/squid.strace -v -f -r /usr/local/squid/sbin/squid
```

and examining the (tedious) output shows two `devices` that are required in out `chroot` jail:

```
/dev/null
/dev/log
```

For the first ls -l /dev/null

```
crw-rw-rw-  1 root root 1, 3 2005-03-21 08:39 /dev/null
```

so

```
cd /chroot_squid/dev
mknod null c 1 3
```

7. To "enable" `/dev/log`, i.e., `syslog`ging for Squid under out `chroot` jail, we note the relevant comments in the widely-available "Chroot Bind Howto" and also examine the `syslogd` man page. Hence, we edit `/etc/init.d/syslogd` to contain

   ```
   SYSLOGD="-m 0 -a /chroot_squid/dev/log"
   ```

   which is equivalent to

   ```
   /sbin/syslogd -m 0 -a /chroot_squid/dev/log
   ```

   We now have our two required devices:

   ```
   ls -l /chroot_squid/dev

   srw-rw-rw-    1 root     root           0 2004-06-21 18:08 log
   crw-rw-rw-    1 root     root       1,   3 2004-06-21 18:23 null
   ```

   and Squid will now start in the `chroot` jail.

9.4. More

Docs – examples :

```
-- Bind-Chroot-Howto (Debian):
       http://www.falkotimme.com/howtos/debian_bind_chroot/index.php

-- apache: http://www.linux.com/article.pl?sid=04/05/24/1450203

-- http://www.l0t3k.org/security/docs/chrooting/:

    -- 'chroot' an Apache tree with Linux and Solaris (Howto)
       Published on 2001-02-26 - by Denice Deatrich, Denice Deatrich.

    -- Apache chrooting made simple
       Published on 2004 - by Ivan Ristic, Ivan Ristic.

    -- Chroot-BIND HOWTO
       Published on December 01, 2001 - by Scott Wunsch, losurs.org.

    -- Chroot-BIND8 HOWTO
       Published on July 01, 2001 - by Scott Wunsch, losurs.org.

    -- Chrooting daemons and system processes HOW-TO
       Published on October 21, 2002 - by Jonathan, www.nuclearelephant.com.

    -- Linux Process Containment &#150; A practical look at chroot and
           User Mode Linux
       Published on June 03, 2003 - by Paul Lessard, SANS Institute.

    -- Setting up chrooted sftp in linux
       Published on 2003 - by James Dennis, James Dennis.
```

10. Breaking Out of and Securing `chroot` Jails

It is not difficult to break out of a `chroot` jail if it is carelessly setup — especially if an intruder can get `root` privileges within the jail.

10.1. Breaking Out Using `chroot()` and `chdir()`

It is possible to break out of a `chroot` jail given `root` privileges — at least on Linux and Solaris, and any other "unix" whose `chroot()` system call works the same, or a similar, way: on most unices, the location of the process's root is stored within its entry in the process table — `chroot()` simply changes this. Hence, in principle, one simply needs to make a sequence of system-calls to `chroot()` (and perhaps `chdir()`) to break out.

(The above procedure will not work on suitably configured FreeBSD v4 and above, or, for example, .)

To break out of a `chroot` jail an intruder will need:

1. Access to a `chroot` jail through a security hole in, e.g., a daemon/service running from the jail: remember to patch the contents of the jail as well as the OS outside; audit those CGI scripts

2. `root`-privilege escalation by some means: again, patch the jail; beware of `setuid root` executables;

3. Ability to upload, or build and run, an executable based on the code outlined below:

```
/* This code based on that from http://www.bpfh.net/simes/ */
....
....
....
```

   More an explanation of the above outline code, see the full version[17] which includes comments and error-detection.

10.2. Breaking Out Using Other Methods

```
-- chroot-breaking buffer overflows
```

If it is possible to create device nodes within the `chroot` jail then a break out is possible by creating your own `/dev/hda` (or other disk node) or `/dev/kmem` — in the latter case it is possible to patch the kernel as it is running and then anything is possible. (Root privilege is required.)

10.3. Influence from *Within* the Jail

```
-- syslog (/dev/log) from within jail, a la squid example above;
-- buffer overflow in syslog?
```

[17] `chroot_break_out.tex`

10.4. Making `chroot` Jails More Secure

Quoting from Chuvakin's article (see below):

. . . if there is no root user defined within the chroot environment, no SUID binaries, no devices, and the daemon itself dropped root privileges right after calling chroot() call (like in the code below), breaking out of chroot appears to be impossible.

So, while there is no such thing as a perfectly secure `chroot` jail, good policy would appear to be:

- ensure software within the jail is patched as well as the host OS;
- do not run daemons as `root`;
- avoid setuid executables;
- ensure `root` (UID 0) does not even exist within the jail;
- if possible, mount your jail on a separate partition with the following options set: `nodev`, `noexec`, `nosuid`;
- install the minimum possible within the jail — offer an intruder nothing to work with;
- use `chattr -i` to help prevent the creation of new files (or nodes);
- apply the   to your kernel, in particular those related to `chroot` jails.

(Until recently it was possible to run binaries on a `noexec`-mounted partition on Linux using a command like `/lib/ld*.so <executable>`. This trick fails since Linux 2.4.25/2.6.0.)

10.5. More

Docs – Securing chroot jails and breaking out of them:

```
 -- How to break out of a chroot() jail
    Published on May 12, 2002 - by Simes, Simes.
      -- http://www.bpfh.net/simes/computing/chroot-break.html

 -- Using Chroot Securely
    Published on October 02, 2002 - by Anton Chuvakin, Guardian Digital,

 -- Runtime Kernel Kmem Patching, Silvio Cesare
      -- http://www.big.net.au/ silvio/runtime-kernel-kmem-patching.txt

 -- Linux on-the-fly kernel patching without LKM, from Phrack Inc, at
      -- http://www.phrack.org/phrack/58/p58-0x07
```

11. `chattr`ing

```
 -- e.g.,
        chattr -R +i /chroot_squid/bin
                                  /dev/
                                  /etc/
                                  /lib/
                                  /usr/

        but not /chroot_squid/var
```

12. `chroot`ing Your Users (`sshd`)

Why? e.g., for gateway machines...

http://www.adg.us/computers/chrsh.html

- chroot openssh (or ssh.com's ssh), or
- chrootssh

The former is non-trivial, the latter is easy.

chrootssh.sourceforge.net[18]

```
-- chroot openssh
    -- /home/simonh/src/openssh-3.9p1-chroot
    -- /opt/sbin/ssh
    -- /opt/etc/ssh*
    -- listening on port 22

    -- /etc/passwd:

        -- e.g., simonhtest:x:1001:100::/chroot/./home/simonhtest:/bin/bash

    -- requires publickey authentication
        -- place keys in /chroot/home/username/.ssh/authorized_keys2, NOT
           in /home/username... as the latter WON'T work!
```

13. Application-Level Proxies

Servers such as Web and FTP should be proxied: a proxy, in this sense, will be a daemon that listens on port 80 or 21 (or other port depending on the service) *in place of the usual server*, checks that any request received is ok (e.g., does not contain a buffer-overload attempt) and only then passes the request onto said server. (The server will receive the request from the proxy via a different port, e.g., 8080 or 2121, which is blocked to the outside world.)

For further protection such a proxy can be run from `inetd` or `xinetd`.

- For a nice and easy set up try Micro Proxy[19].
- Alternatively, try `httpdproxy`[20] which is written in Perl and makes use of Netcat (`nc`): it's to customize but not fast enough for a busy site.
- The classic proxy is SOCKS, but the SOCKS path is not an easy one.
- Zorp[21]: again not for the faint-hearted as this will probably mean kernel customisation.

[18] `http://chrootssh.sourceforge.net/`
[19] `http://www.acme.com/software/micro_proxy/`
[20] `../_isd_httpd_proxy`
[21] `http://www.balabit.com/products/zorp_gpl/`

14. Apache: modsecurity

From the Web site[22]: *ModSecurity is an open source intrusion detection and prevention engine for web applications (or a web application firewall). Operating as an Apache Web server module or standalone, the purpose of ModSecurity is to increase web application security…*

14.1. Introductory Material

- *Introducing mod_security*[23], by Ivan Ristic (2003/11/26).
- *Web Security Appliance With Apache and mod_security*[24], by Ivan Ristic (2003/10/21).

14.2. Documentation

The reference manual and other documentation[25] can be found at the Mod-Security Web site.

14.3. Download and Installation

14.3.1. Debian

```
apt-get install libapache-mod-security
```

14.3.2. Source

Get the source code from modsecurity.org/download[26] and follow the instructions given in the reference manual[27].

14.4. Configuration and Loading

First, the module must be loaded by Apache when it starts, so in `httpd.conf` (or whatever your Apache configuration file is called)

```
LoadModule security_module /usr/lib/apache/1.3/mod_security.so
```

and *perhaps*

```
AddModule mod_security.c
```

Secondly, the module must be configured, so in `http.conf` (or whatever…) add

```
Include mod_security.conf
```

[22] `http://www.modsecurity.org/`
[23] `http://www.onlamp.com/pub/a/apache/2003/11/26/mod_security.html`
[24] `http://www.securityfocus.com/infocus/1739`
[25] `http://www.modsecurity.org/documentation/index.html`
[26] `http://www.modsecurity.org/download/`
[27] `http://www.modsecurity.org/documentation/`

or *perhaps*

```
<IfModule mod_security.c>
  Include mod_security.conf
</IfModule>
```

with `mod_security.conf`

```
  # This is a simple illustrative example only;  mod_security can do far
  # more --- see the Reference Manual for more.

  # -- PART ONE: filter configuration :

  SecFilterEngine On
      # filter every request

  SecFilterScanPOST On
      # scan body (or POST) payload (disabled by default)

  SecFilterDefaultAction "deny,log,status:404"
      # sets default action to return 404 (not found) --- the default default
      # is 403 (forbidden)

  # -- PART TWO: rules :

  SecFilter passwd
  SecFilter shadow
      # block requests including "passwd" or "shadow" in the
      # request string
```

14.5. Logs — Testing

Logs appear in `/var/log/error.log` — using a browser to attempt access to `http://mctalby.mc.man.ac.uk/~mc/passwd` yields:

```
  [Wed Jan  4 13:54:54 2006] [error] [client 130.88.201.157] \
      mod_security: Access denied with code 404. Pattern match "passwd" \
      at THE_REQUEST. [hostname "mctalby.mc.man.ac.uk"] [uri "/ mc/passwd"]
```

14.6. More

`mod_security` can do much more, including:

- path normalisation;
- null-byte attack prevention (against C-based software);
- handle regular expressions;
- filter on CGI variables (e.g., `REMOTE_ADDR`, `QUERY_STRING`);

14.7. Actions

For each matching rule, `mod_security` can perform *actions*:

- continue with the request, or not;
- change the flow or fules (e.g., skip or jump);
- and more.

14.8. Dynamic Firewalling

Use the `exec` action to execute a script which `INSERT`s rules into IPTables (or other firewall) which given rules are matched — dangerous. This can be *used* by those attacking your server to perform a DoS on your machine…

14.9. Regular Expressions

<FONT COLOR="red"> N.B. *Apache 1.3 uses POSIX regular expressions; Apache 2.n uses PCRE (Perl-compatible regular expressions).* </FONT>

14.10. Variables

Many such variables exist, such as `REMOTE_HOST` and `REQUEST_METHOD`, which correspond to HTTP MIME headers. There are several additional variables, including:

`THE_REQUEST`	The full HTTP request line sent by the browser to the server, e.g., `GET /index.html HTTP/1.1`. This does not include any additional headers sent by the browser.
`REQUEST_URI`	The resource requested in the HTTP request line. In the example above, this would be `/index.html`.

14.11. Example: Default Deny

For the truly paranoid, one can implement a default-deny set of rules, like this:

```
# ---------------------------------------------------------------------------------------------█
---------
# -- Configuration :
# ---------------------------------------------------------------------------------------------█
---------

        ##
        ## see "Configuration" from the "ModSecurity for Apache User Guide"
        ## for more details
        ##

SecFilterEngine On
    # ...On = analyse every request


SecFilterScanPOST On
    # ...On = turn on scanning of body payload (or POST payload) --- default is off


SecFilterSelective HTTP_Content-Type "!(^$|^application/x-www-form-urlencoded$|^multipart/form-█
data;)"
    # ...mod_security supports only two types of body:
    #
    #        application/x-www-form-urlencoded (used to transfer form data)
    #        multipart/form-data (used for file transfers)
    #
    #    so allow only these (few web apps use other types)...


SecFilterSelective HTTP_Transfer-Encoding "!^$"
    # ...block chunked-transfer-encoding of requests (not of responses) since mod_security does not█
    #    yet support chunked requests (but then nor do any browsers yet either)...


SecFilterDefaultAction "deny,log,status:402"
    # ...default action for a matching rule --- don't accede to the request, log the request (it's█
    #    denial, that is) and respond with 404 ("Not found")...
    #
    #    the default default is 403 ("Forbidden") which sounds bad to me (indicates existence)...█


SecFilterCheckURLEncoding On
    # ...On = turn on URL encoding checks (block attacks which use %XY where X, Y are _not_
    #    in [0-9], [a-f])


SecFilterCheckUnicodeEncoding On
    # ...On = check encoding of characters is correct for UTF-8


##SecFilterForceByteRange 32 126
    # ...This directive allows only one range to be specified. But one can cheat --- filter on█
    #    multiple ranges --- e.g. thusly:
    #
    #        SecFilterSelective THE_REQUEST "!^[\x0a\x0d\x20-\x7e]+$"
    #
    #     ...allows characters 10, 13 and 32-126...
    #
    # http://www.ascii.cl/htmlcodes.htm :
    #
    #  32 -- 126
    #  NOT 127
    #  NOT 128 -- 159
    #  160 -- 255
    #
SecFilterSelective THE_REQUEST "!^[\x20-\x7e\xa0-\xff]+$"
```

15. Packet Filters and Firewalls

15.1. What's a packet filter?

Data (including connection attempts) arriving from the Internet at your computer travels across the network in *packets*. Both Solaris and Linux can filter these packets against administrator-configured ordered lists of rules.

15.2. IPFilter

IP Filter is a freely-available stateful firewall/packet-filter. It comes with FreeBSD and NetBSD (and until recently OpenBSD, which has switched to PF); it also runs on many other Unix-like operating systems, including Solaris. For more information see the HOWTO at obfuscation.org[28] or the FAQ at phildev.net[29].

15.2.1. Installation on Solaris

Binaries of IP Filter — Solaris packages — can be downloaded from maraudingpirates.org[30], for Solaris 7, 8 and 9. These can be installed in the usual way: `gunzip` the `.gz` files and then `pkgadd -d <filename>`, e.g.,

```
gunzip ipf-3.4.28-Sol8-sparc-64bit.pkg.gz
pkgadd -d ipf-3.4.28-Sol8-sparc-64bit.pkg

    1  ipf      IP Filter
                (sparc) 3.4.23
    2  ipfx     IP Filter (64-bit)
                (sparc) 3.4.23

    Select package(s) you wish to process (or 'all' to process
    all packages). (default: all) [?,??,q]:
```

— the `.pkg` file actually contains two packages: `ipf` and `ipfx`; the latter is the 64bit version.

If you get a message such as

```
Cannot attach to device
```

during the post-installation phase remove both packages (use `pkgrm`) and install `ipfx` before `ipf` (i.e., choose `2`, then `1`).

15.2.2. Configuration — Init Scripts

The Solaris packages from `maraudingpirates.org` contain suitable `init` scripts and these are installed as `/etc/init.d/ipfboot` and `/etc/rc3.d/S65ipfboot` (the latter being installed as a copy of the former, in the usual Solaris-oriented way).

[28] `http://www.obfuscation.org/ipf/`
[29] `http://www.phildev.net/ipf/`
[30] `http://www.maraudingpirates.org/ipfilter/`

This script must be altered if it is to be used in conjunction with the shell scripts given below. The following init script can be used instead:

```
  #!/bin/sh
  #
  PIDFILE=/etc/opt/ipf/ipmon.pid

  id=`/usr/sbin/modinfo 2>&1 | awk '/ipf/ { print $1 } ' - 2>/dev/null`
  if [ -f $PIDFILE ] ; then
      pid=`cat $PIDFILE 2>/dev/null`
  else
      pid=`/bin/ps -e 2>&1 | awk '/ipmon/ { print $1 } ' - 2>/dev/null`
  fi


  case "$1" in
start)
if [ x"$pid" != x ] ; then
kill -TERM $pid 2>/dev/null
fi
if [ x$id != x ] ; then
modunload -i $id 2>/dev/null
fi
modload /usr/kernel/drv/ipf

ipf   -Fa -f /etc/ipf.rules
ipnat -CF -f /etc/ipnat.rules

ipmon -Ds
;;

stop)
ipf   -Fa
ipnat -CF
;;

reload)
ipf -Fa -f /etc/ipf.rules
                ipnat -CF -f /etc/ipnat.rules
;;

*)
echo "Usage: $0 (start|stop|reload)" >&2
exit 1
;;

  esac
  exit 0
```

15.2.3. Configuration — Firewall Scripts

One or two files are required: the firewalling rules are necessary and usually contained in `/etc/ipf.rules` ; optionally network-address-translation rules can be set, usually in `/etc/ipnat.rules`. (See the corresponding lines in the above init script in both cases.)

Below is a simple default deny firewall which can easily be modified. Many more examples and explanation can be found in the HOWTO linked-to above.

```
# ----------------------------------------------------------------------------
# -- default deny :

block in on hme0  # -- from any to any (no "quick")

# ----------------------------------------------------------------------------
# -- loopback :

pass out quick on lo0
pass in  quick on lo0

# ----------------------------------------------------------------------------
# -- OUT :

pass out quick on hme0 proto tcp  from any to any keep state
pass out quick on hme0 proto udp  from any to any keep state
pass out quick on hme0 proto icmp from any to any keep state

# ----------------------------------------------------------------------------
# -- allow new tcp connections from trusted hosts and keep-state on them :

pass in quick on hme0 proto tcp from 192.168.1.7  to any flags S keep state
pass in quick on hme0 proto tcp from 192.168.1.8  to any flags S keep state
pass in quick on hme0 proto tcp from 192.168.1.15 to any flags S keep state

# ---------------------------------------------------------------------------
# ...allow unprivileged port access --- the simplest way to handle protocols
#    which IPFilter connection-tracking cannot cope with, e.g, FTP :

pass in quick on hme0 proto tcp from 130.88.0.0/16 to any port > 1023
```

A better solution to handling FTP is to set up some NAT rules — *omit the last line of the script above* and add this to `/etc/ipnat.rules`:

```
# -- handle moronic FTP :  this lets _clients_ on Cosmos get data back in
#    an Active FTP session;  Passive transfers should work already
#    via TCP state-keeping) :

map hme0 0/0 -> 0/32 proxy port 21 ftp/tcp
```

15.2.4. Starting/Stopping

Finally you need to start the filter and NAT either directly,

```
    ipf  -Fa -f /etc/ipf.rules
    ipnat -CF -f /etc/ipnat.rules
```

or by using the initialisation script,

```
    /etc/init.d/ipfboot start
```

15.2.5. Related Utilities

To list current filter rules use:

```
ipfstat -i    # -- for rules affecting inbound traffic
ipfstat -o    #                     ...outbound...
```

15.3. Netfilter and `iptables`

Linux kernels v2.4 and greater come with the Netfilter (IPTables) stateful firewall (and network address transation) utility:

- Netfilter Homepage[31]
- [32]
- [33]

15.3.1. Init Scripts and Configuration

Configuration of Netfilter IPTables depends on your distribution. On a RedHat box, to manually configure the firewall simply edit the rules within `/etc/sysconfig/iptables` and then restart by using the supplied init script

```
/etc/init.d/iptables restart
```

and confirm via

```
iptables -L          #  Redhat: /sbin/iptables;  SuSE: /usr/sbin/iptables
```

`/etc/sysconfig/iptables` is actually a Gnome Lokkit file. (Lokkit is a utility that provides a firewall configuration based on a small number of simple questions. It is designed for "the average" Linux user.)

Things are someone different on Debian boxes — more pure Netfilter/IPTables (though frontends, such as those mentioned below, can of course be used). Pertinant files are:

```
/etc/init.d/iptables               # init script
/etc/default/iptables              # iptables config
/var/lib/iptables/<ruleset-name>  # saved rule sets
```

The procedure is as follow:

1. Set up a firewall via, e.g., a shell script (like that given below).
2. Save the rule-set

   ```
   /etc/init.d/iptables save edward
   ```

3. Alternative rule sets can be saved, e.g.,

   ```
   /etc/init.d/iptables save edward.v2>
   ```

[31] `http://www.netfilter.org/`
[32] `http://www.netfilter.org/documentation/index.html#documentation-howto`HOWTO
[33] `http://www.netfilter.org/documentation/index.html#documentation-faq`FAQ

and then, e.g., the first can be reloaded

```
        /etc/init.d/iptables load edward
```

Here is a simple shell script which will set up an `iptables`-based firewall which example which can be trivially-modified for most machines:

```
#!/bin/sh

# -- where's the iptables binary?
IPT="/sbin/iptables"

# -- to start with, clean out the bath :
for i in filter nat mangle
do
    $IPT -t $i -F
    $IPT -t $i -X
done


# -- if its related to something that's already started, allow it :
$IPT -t filter -A INPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A OUTPUT -m state --state NEW,ESTABLISHED     -j ACCEPT

# -- let me talk to myself :
$IPT -t filter -A INPUT -s 192.168.1.2 -j ACCEPT
$IPT -t filter -A INPUT -s 127.0.0.1 -j ACCEPT


# -- udp :
$IPT -t filter -A INPUT  -p udp -j DROP

# -- icmp (ping, for example) :
$IPT -t filter -A INPUT -p icmp -j DROP


# -- if it's one of us, that's ok :
$IPT -t filter -A INPUT -s 192.168.1.3   -m state --state NEW -j ACCEPT  # -- me
$IPT -t filter -A INPUT -s 192.168.1.123 -m state --state NEW -j ACCEPT  # -- trusted friend
$IPT -t filter -A INPUT -s 130.88.99.10 -m state --state NEW -j ACCEPT   # -- cosmos


# -- every other (new) connection can go in the bin :
$IPT -t filter -A INPUT -m state --state NEW -j DROP

# -- default policies :
$IPT -t filter -P INPUT DROP
$IPT -t filter -P OUTPUT ACCEPT
$IPT -t filter -P FORWARD DROP
```

Finally you need to start the filter; the best way to do this is to use the boot-script:

```
    /etc/init.d/iptables start
```

N.B. You may need to hack this script for it to work successfully with the given configuration script — see the example boot-script given in *Unix Security — A Survey* for how to do this.

16. Graphical Firewall Front Ends

In addition, many front ends exist which aim to make the configuration of a good firewall with Netfilter/IPTables easier. Two of these are:

- FERM[34], a firewall rule parser: ferm compiles ready to go firewall-rules from a structured rule-setup. . . the possibility to provide a structured description of a firewall. . . you can now write logically and coherent rules using a C-style nesting structure, and let ferm create all rules for you. . . ferm will also aid in modularizing firewalls. . .

- Firewall Builder[35] is a GUI for building firewalls. From the Linux Journal (Paranoid Penguin, Mick Bauer) article on FB, LJ109: Firewall Builder is a good firewall GUI indeed. It lets you define host, network and service objects that can be used and reused in as many different firewall rulesets as you like; it displays your rules in an instinctive and clear way; and because it's intentionally OS-agnostic, you can use Firewall Builder to generate rulesets not only for Netfilter/iptables, but also for FreeBSD's ipfilter, OpenBSD's pf and even Cisco PIX firewalls.

17. Pinprick Firewalls

A default-deny stance on in-bound traffic makes a good firewall; a *default-deny stance on out-bound traffic* in addition to this makes a better one. Firewalls in which not a single packet is allowed in or out of a machine unless it matches a tightly-specified rule are called pinprick (or pinhole) firewalls.

[34] `http://ferm.sourceforge.net/ferm.html`
[35] `http://www.fwbuilder.org/`

17.1. Example

This example is implemented using IPTables. Points to note — the machine running this firewall is:

1. running an HTTP server for which access is granted to all;
2. running an SSH server for which access is granted to a strictly limited list of hosts;
3. mounting a disk via NFS from 130.88.254.254;
4. using FTP to get patches from a remote, anonymous site;
5. blocking most out-bound new connections from clients on the machine — only DNS lookups and SSH connections (at least, those that look like SSH connections) are allowed.

```
#!/bin/sh

# -- where's the iptables binary?
IPT="/sbin/iptables"


# -------------------------------------------------------------------------------------------
# -- to start with, clean out the bath :
# -------------------------------------------------------------------------------------------

for i in filter nat mangle
do
    $IPT -t $i -F
    $IPT -t $i -X
done


# -------------------------------------------------------------------------------------------
# -- create our tables :
# -------------------------------------------------------------------------------------------

    # ...we use separate tables for different things, since it's faster...

$IPT -N TCP_IN
$IPT -N TCP_OUT
$IPT -N UDP_IN
$IPT -N UDP_OUT


# -------------------------------------------------------------------------------------------
# -- me/local :
# -------------------------------------------------------------------------------------------

    # ...allow me to talk to myself on both loopback and external-facing devices...

$IPT -t filter -A INPUT  -s 127.0.0.1 -j ACCEPT
$IPT -t filter -A OUTPUT -s 127.0.0.1 -j ACCEPT

$IPT -t filter -A INPUT   -i lo  -p tcp  -s 130.88.253.254  -j ACCEPT
$IPT -t filter -A OUTPUT  -o lo  -p tcp  -s 130.88.253.254  -j ACCEPT


# -------------------------------------------------------------------------------------------
# -- remote debug :
# -------------------------------------------------------------------------------------------

    # ...occasionally we are crazy enough to modify the IPTables rules remotely, so offer
    #    a safety net:  allow in a hardened machine to sort it out :

$IPT -t filter -A INPUT  -s 130.88.253.255 -j ACCEPT
$IPT -t filter -A OUTPUT -d 130.88.253.255 -j ACCEPT


# -------------------------------------------------------------------------------------------
# -- handle TCP and UDP :
# -------------------------------------------------------------------------------------------

    # ...TCP and UDP traffic, in-bound and out-bound, is handled by the
    #    appropriate table, so we jump to it...

$IPT -t filter -A INPUT  -p tcp -i eth0 -d 130.88.253.254  -j TCP_IN
$IPT -t filter -A OUTPUT -p tcp -o eth0 -s 130.88.253.254  -j TCP_OUT

$IPT -t filter -A INPUT  -p udp -i eth0  -d 130.88.253.254  -j UDP_IN
$IPT -t filter -A OUTPUT -p udp -o eth0  -s 130.88.253.254  -j UDP_OUT
```

18. Practical Steps to Developing and Testing a Firewall (on an In-Service Machine)

. . . log. . .

19. Router ACLs

. . . Peter Smith. . .

20. What's listening on your machine? — nmap

You should know about **every** *process which is listening to a port on your machine — what it's called and* **why** *its listening.* Unless you need it, stop it or remove it [Page 3].

`netstat` can be used on the machine (e.g., `netstat -a | grep LIST`) but this is based on the assumption that the the host has not been compromised (any decent intruder will trojan `netstat` in order to hide their presence).

A better approach is to scan for open ports from a second machine. This second machine must be able to "see through" any firewall — alternatively, simple turn any such firewall *temporarily*. `nmap`[36] is ideal for this. For example, to scan privileged ports on `dog.sub.domain` from `cat.sub.domain`:

```
cat> nmap -vv -sT -p 1--1023 dog.sub.domain

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host dog.sub.domain (130.88.???.???) appears to be up ... good.
Initiating Connect() Scan against dog.sub.domain (130.88.???.???)
Adding open port 25/tcp
Adding open port 787/tcp
Adding open port 111/tcp
Adding open port 587/tcp
Adding open port 22/tcp
Bumping up senddelay by 10000 (to 10000), due to excessive drops
The Connect() Scan took 30 seconds to scan 1023 ports.
Interesting ports on eric.umist.ac.uk (130.88.99.9):
(The 1018 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
111/tcp    open        sunrpc
587/tcp    open        submission
787/tcp    open        unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 30 seconds
```

Commonly-used nmap options:

```
-v, -vv, -vvv     verbose, very verbose...
-sT, -sU          TCP scan, UDP scan
-p m-n            range to scan (to scan all ports, omit this)
```

[36] `http://www.insecure.org/nmap/`

Armed with information such as that above, you should identify the daemon responsible for each open port (e.g., by using `lsof` — see LSOF: Which process, which port? [Page 36])

21. Vulnerability Detection — Nessus

The Internet community contains a plethora of information on network and local OS vulnerabilities. Nessus[37] is scanner which looks for these and reports on them in detail.

22. Periodic Scans and `nmapsql`

Periodically scan your own machines (from a host with complete access through the firewall) for open ports and compare output to that from a base established immediately after the installation. Notice any differences and ensure you know the reason for them:

A few weeks ago you port-scanned your machine as part of a security audit with the following results:

```
Port      State     Service
22/tcp    open      ssh
25/tcp    open      smtp
```

Today you get

```
Port      State     Service
22/tcp    open      ssh
25/tcp    open      smtp
2105/tcp  open      unknown
```

This is strong evidence that you've been hacked! One can use regular scans of machines as an intrusion detection method.

Ideally one would want automated scans (easy, use `cron`) and a system for storing results of scans of many machines and for determining differences from previous scans. `nmapsql` is just such a system:

- An introduction to `nmapsql` may be found in an article in Linux Journal LJ126[38].
- The project home-page is at Sourceforge[39].

23. Which process, which port?

From a trusted installation of `netstat`, or from a remote scan using `nmap` you have a list of open ports on a machine. So what daemon/software is responsible for each port?

[37] `http://www.nessus.org`
[38] `http://interactive.linuxjournal.com/Magazines/LJ126/7314.html`
[39] `http://sourceforge.net/projects/nmapsql`

`lsof` output looks like this:

```
COMMAND     PID        USER   FD   TYPE     DEVICE     SIZE       NODE NAME
init          1        root  cwd    DIR       3,66     1024          2 /
init          1        root  rtd    DIR       3,66     1024          2 /
init          1        root  txt    REG       3,66    31432     106384 /sbin/init
init          1        root  mem    REG       3,66    90088      85918 /lib/ld-2.3.2.so
init          1        root  mem    REG       3,66  1244080      85954 /lib/libc-2.3.2.so
init          1        root   10u  FIFO       3,66                70570 /dev/initctl
.
.
snortsam- 28305        root  cwd    DIR       3,69     4096     115564 /usr/local/src/snortsam-█
2.25
snortsam- 28305        root  rtd    DIR       3,66     1024          2 /
snortsam- 28305        root  txt    REG       3,69    95541     115672 /usr/local/src/snortsam-█
2.25/snortsam-debug
snortsam- 28305        root  mem    DEL       3,66                85938 /lib/ld-2.3.2.so.dpkg-█
new
snortsam- 28305        root  mem    DEL       3,66                85955 /lib/libpthread-0.10.so.dpkg-█
new
snortsam- 28305        root  mem    DEL       3,66                85941 /lib/libc-2.3.2.so.dpkg-█
new
snortsam- 28305        root   0u    CHR     136,30                   32 /dev/pts/30
snortsam- 28305        root   1u    CHR     136,30                   32 /dev/pts/30
snortsam- 28305        root   2u    CHR     136,30                   32 /dev/pts/30
snortsam- 28305        root   3u   IPv4   85171520                  TCP *:898 (LISTEN)
.
.
```

Remember that everything in Unix (and Linux) is a file — including network sockets.

We are interested in files of `TYPE IPv4`, or, on other versions of `lsof`, `TYPE inet`. So:

```
lsof | egrep "inet|IPv4"

ssh       28296      simonh    3u  IPv4   85171489                  TCP localhost:33450->localhost:ssh
sshd      28297        root    4u  IPv4   85171490                  TCP localhost:ssh->localhost:33450
snortsam- 28305        root    3u  IPv4   85171520                  TCP *:898 (LISTEN)
firefox-b 31980          mc    3u  IPv4  223586443                  TCP localhost:57567->localhost:601
t
.
.
```

The name of the executed file responsible for a corresponding open port is listed on the left.

24. Example: Hardening a Solaris Installations

The Solaris installation programme does not have a “build me a secure box” option. These notes outline how a standard Solaris 9 installation can be secured: the object is to have a box which can be accessed remotely via SSH and therefore used as a general purpose Unix server.

24.1. Patch and Firewall

First patch [Page 10] and firewall [Page 27] the machine.

24.2. Network-Related Daemons

In this section, our aim is to permanently stop network-related daemons which are not required, or better still, remove all software related to them. (Removal means fewer tools for intruders to play with.)

Whilst there are high-level, graphical tools which help with Solaris software package administration, we require something more fine-grained; in addition, it is beneficial to understand the Solaris package system so we opt for use of the command-line tools such as `pkginfo` and `pkgrm`. And `/var/sadm/install/contents` tells us which files belong to which packages.

The remainder of this section reflects the rough journal of the securing of and a real installation. As such it is not a textbook example, but a real one!

Initial Scan of Open Ports

Scanning the Solaris box with `nmap` shows a scary number of possible entry points for would-be hackers:

```
7/tcp      open        echo
9/tcp      open        discard
13/tcp     open        daytime
19/tcp     open        chargen
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     open        telnet
37/tcp     open        time
79/tcp     open        finger
111/tcp    open        sunrpc
512/tcp    open        exec
513/tcp    open        login
514/tcp    open        shell
515/tcp    open        printer
540/tcp    open        uucp
587/tcp    open        submission
898/tcp    open        unknown
4045/tcp   open        lockd
6112/tcp   open        dtspc
7100/tcp   open        font-service
32771/tcp  open        sometimes-rpc5
32772/tcp  open        sometimes-rpc7
32773/tcp  open        sometimes-rpc9
32774/tcp  open        sometimes-rpc11
32775/tcp  open        sometimes-rpc13
32776/tcp  open        sometimes-rpc15
32777/tcp  open        sometimes-rpc17
32778/tcp  open        sometimes-rpc19
32779/tcp  open        sometimes-rpc21
32780/tcp  open        sometimes-rpc23
```

`inetd`

Start simple: SSH does not run out of the super-server, `inetd`, so we comment out *almost everything* from `/etc/inetd.conf` and

```
kill -HUP <inetd pid>
```

Network-Related Packages (`r`-commands; `finger`)

Next, look for network-related packages.

```
pkginfo | grep -i network
```

yields three SUNW packages to uninstall — shown also are packages which depend on those we want to remove (*indentation indicates a package dependency* — an indented package depends on the above, less indented package):

```
SUNWrcmdc        Remote Network Client Commands
    SUNWppm        Solaris Print Manager
        SUNWmp        MP Print Filter
    SUNWscpu        Source Compatibility, (Usr)
        SUNWbcp        SunOS 4.x Binary Compatibility
            SUNWscbcp        SPARCompilers Binary Compatibility Libraries

SUNWrcmdr        Remote Network Server Commands (Root)
SUNWrcmds        Remote Network Server Commands (Usr)
```

Removing all the above, and bringing the machine into single-user mode and backup, the following are gone (good!):

```
79/tcp     open        finger
512/tcp    open        exec                    # rexec
513/tcp    open        login                   # rlogin
514/tcp    open        shell                   # rsh
```

Printer Daemon

Next, we target the printer daemon:

```
pkginfo | grep -i printer
```

yields candidates: </PRE> SUNWpsu Solaris Print - LP Server, (usr) SUNWpsr Solaris Print - LP Server, (root) SUNWpcu Solaris Print - Client, (usr) SUNWscplp Solaris Print - Source Compatibility SUNWpcr Solaris Print - Client, (root) </PRE> Removal, moving to single-user mode and bringing the system back up we find

```
515/tcp    open        printer
```

has gone. </PRE>

UUCP, FTP, Telnet and NTP

And now, UUCP, FTP, Telnet and NTP:

```
pkginfo | grep -i uucp
pkginfo | grep -i ftp
pkginfo | grep -i telnet
pkginfo | grep -i ntp
```

```
yielded

     SUNWbnur       Networking UUCP Utilities, (Root)
     SUNWbnuu       Networking UUCP Utilities, (Usr)

     SUNWftpr       FTP Server, (Root)
     SUNWftpu       FTP Server, (Usr)

     SUNWtnetr      Telnet Server Daemon (Root)
     SUNWtnetc      Telnet Command (client)
     SUNWtnetd      Telnet Server Daemon (Usr)

     SUNWntpr       NTP, (Root)
     SUNWntpu       NTP, (Usr)

eliminating

    540/tcp    open        uucp
    21/tcp     open        ftp
    23/tcp     open        ftp
                                  # The ntp daemon was not actually listening.
```

X Font Server

The X font server:

```
SUNWxwfs        X Window System Font server

7100/tcp   open        font-service
```

YP, NIS+, NFS, Samba, PPP, HTTP and DHCP

More Services: YP/NIS, NIS+, NFS, Samba, PPP, Web, DHCP

```
SUNWypr         NIS Server for Solaris (root)
SUNWypu         NIS Server for Solaris (usr)


SUNWnisr        Network Information System, (Root)
SUNWnisu        Network Information System, (Usr)


SUNWnfscr       Network File System (NFS) client support (Root)
    SUNWvolu        Volume Management, (Usr)
        SUNWvolg         Volume Management Graphical User Interface
SUNWnfscu       Network File System (NFS) client support (Usr)
SUNWnfscx       Network File System (NFS) client support (Root) (64-bit)
SUNWnfssr       Network File System (NFS) server support (Root)
SUNWnfssu       Network File System (NFS) server support (Usr)
SUNWnfssx       Network File System (NFS) server support (Root) (64-bit)


SUNWsmbac       samba - A Windows SMB/CIFS fileserver for UNIX (client)
SUNWsmbar       samba - A Windows SMB/CIFS fileserver for UNIX (Root)
SUNWsmbau       samba - A Windows SMB/CIFS fileserver for UNIX (Usr)


SUNWpppd        Solaris PPP Device Drivers
SUNWpppdr       Solaris PPP configuration files
SUNWpppdt       Solaris PPP Tunneling
SUNWpppdu       Solaris PPP daemon and utilities
SUNWpppdx       Solaris PPP Device Drivers (64-bit)
SUNWpppg        GNU utilities for PPP


SUNWapchr       Apache Web Server (root)
SUNWapchu       Apache Web Server (usr)
SUNWapchd       Apache Web Server (usr)


SUNWtcatr       Tomcat Servlet/JSP Container (root)
SUNWtcatu       Tomcat Servlet/JSP Container


SUNWdhcm        DHCP Manager
SUNWdhcsb       Binary File Format Data Module for BOOTP/DHCP Services
SUNWdhcsr       BOOTP/DHCP Server Services, (Root)
SUNWdhcsu       BOOTP/DHCP Server Services, (Usr)
```

What's Left?

So what's left?

```
22/tcp     open       ssh
25/tcp     open       smtp
111/tcp    open       sunrpc
587/tcp    open       submission
6112/tcp   open       dtspc
7100/tcp   open       font-service
32771/tcp  open       sometimes-rpc5
32772/tcp  open       sometimes-rpc7
32773/tcp  open       sometimes-rpc9
32774/tcp  open       sometimes-rpc11
32775/tcp  open       sometimes-rpc13
32776/tcp  open       sometimes-rpc15
32777/tcp  open       sometimes-rpc17
32778/tcp  open       sometimes-rpc19
```

We have removed the X-font-server package and no XFS-type daemon is running, but the machine is still listening on 7100!?!? Aha! The font-server entry in `inetd.conf` is still there… Comment out.

Sendmail

Next up, Sendmail: we want to be able to send email but not receive it so rename and `chmod 400` the sendmail init script in `/etc/init.d` and remove `S88sendmail` from `rc2.d`:

```
mv sendmail __sendmail.orig_install.no_start
```

and eliminated

```
25/tcp open smtp
```

Hmmm, port 587 also closed, I wonder what did that? Google [`port 587 solaris`]: <BLOCKQUOTE> If you are using sendmail to receive messages (on port 25), you should turn off the Message Submission port (587/tcp). In /etc/mail/sendmail.cf, comment out

```
O DaemonPortOptions=Port=587, Name=MSA, M=E
```

with a hashmark </BLOCKQUOTE> So we got one for free!

CDE (`/usr/dt`)

There is a load of RPC-related stuff to get rid of. CDE uses RPC and its crap anyway… There's also `dtspc` on `6112`:

```
fgrep dtspc /var/sadm/install/contents
```

yields

```
/usr/dt/bin/dtspcd f none 0555 root bin 27376 27505 1016069365 SUNWdtdmn
/usr/dt/config/dtspcdenv f none 0444 root bin 1183 21978 1016069365 SUNWdtdmn
/usr/dt/share/man/man1m/dtspcd.1m f none 0444 root bin 5230 44603 1016073808 SUNWdtma
/usr/dt/share/man/man4/dtspcdenv.4 f none 0444 root bin 5316 53514 1016073844 SUNWdtma
```

so remove

```
SUNWdtma     CDE man pages
SUNWdtdmn    CDE daemons
    SUNWscgui    Solaris Smart Card Administration GUI
    SUNWjmfp     Java Media Framework Player
    SUNWpdas     PDA Synchronization for Solaris
    SUNWdtjxt    Java Extensions
        SUNWdtdst    CDE Desktop Applications
            SUNWpmowu  Power Management OW Utilities, (Usr)
            SUNWdtnsc     Netscape Componentization Support for CDE
            SUNWpmowm  Power Management OW Utilities Man Pages
    SUNWdthe     CDE HELP RUNTIME
    SUNWdtezt      Solaris Desktop Extensions Applications
        SUNWdtmaz       Desktop Power Pack man pages
    SUNWdtim   Solaris CDE Image Viewer
    SUNWdtwm  CDE DESKTOP WINDOW MANAGER
    SUNWdtab  CDE DTBUILDER
    SUNWdthev    CDE HELP VOLUMES
    SUNWdthez   Desktop Power Pack Help Volumes
    SUNWdtdem       CDE DEMOS
```

Solaris Management Tools

Noticed this on a restart:

```
Shutting down Solaris Management Console server on port 898.
```

so

```
pkginfo | grep -i management
```

so

```
SUNWmcex       Solaris Management Console 2.1 (Examples)
SUNWmcdev      Solaris Management Console 2.1 (Development Kit)
SUNWmc         Solaris Management Console 2.1 (Server Components)
    SUNWwbmc       Solaris Management Console 2.1 (WBEM Components)
        SUNWmga   Solaris Management Applications
            SUNWdclnt    Solaris Diskless Client Management Application
            SUNWlvmg     Solaris Volume Management Application
            SUNWrmui     Resource Management User Interface Components
            SUNWpmgr     Solaris Patch Management Applications
    SUNWlvma    Solaris Volume Management APIs
    SUNWmga     Solaris Management Applications
SUNWmcc        Solaris Management Console 2.1 (Client Components)
SUNWmccom      Solaris Management Console 2.1 (Common Components)
```

then `init s`, `init 3` had no effect! Try a reboot — eliminated:

```
898/tcp    open        unknown
```

Solaris Web Management Tools

A `netstat -a | grep LIST` shows up

```
*.5987               *.*                0      0 49152      0 LISTEN
```

Google [`port 5987 solaris`] <BLOCKQUOTE> The CIM Object Manager listens for remote method invocation (RMI) connections on RMI port 5987 and now listens for XML/HTTP connections on HTTP port 5988. (In the Solaris 8 software release and updates of the Solaris 8 release, the CIM Object Manager listened for XML/HTTP connections on default HTTP port 80.) </BLOCKQUOTE> Solaris documentation indicates that CIM is part of WBEM.

```
fgrep -i WBEM /var/sadm/install/contents
```

yields up

```
SUNWmgapp  WBEM Management Applications
    SUNWrmwbu     Resource Management WBEM Instrumentation (usr)
    SUNWrmwbx Resource Management WBEM Instrumentation (64-bit

SUNWwbcou       WBEM Services (usr)
```

Restart; eliminated!

Nearly There!

Aside from SSH, only RPC-related stuff left:

```
22/tcp     open        ssh
111/tcp    open        sunrpc
32771/tcp  open        sometimes-rpc5
32772/tcp  open        sometimes-rpc7
32773/tcp  open        sometimes-rpc9
32774/tcp  open        sometimes-rpc11
```

The Last Few RPC-Related Daemons

What processes are running?

```
ps -ef
```

reveals

```
rpc.ttdbserverd
snmpXdmid
dmispd
```

Of ToolTalk Google says: <BLOCKQUOTE> CDE ToolTalk database server (ttdbserver) allows remote attackers to overwrite arbitrary memory locations with a zero, and possibly gain </BLOCKQUOTE> and we know about SNMP…

```
fgrep -i tooltalk /var/sadm/install/contents
```

gives candidates:

```
SUNWtltk  ToolTalk runtime
    SUNWolrte    OPEN LOOK toolkits runtime environment
    SUNWtltkd    ToolTalk developer support
    SUNWtltkm    ToolTalk manual pages
    SUNWdtct     UTF-8 Code Conversion Tool
    SUNWxwdem    X Window System demo programs
    SUNWlpmsg    LP Alerts

 # ...and for SNMP :

 SUNWsacom Solstice Enterprise Agents 1.0.3 files for root file system
     SUNWmipr   Mobile-IP (Root)
         SUNWmipu    Mobile-IP (Usr)
```

Inetd and RPC: Finally

A final consideration of `inetd` and RPC — we don't need the former since SSH runs independently and we don't want the latter, so:

```
/etc/init.d/inetsvc stop    (to stop inetd)
/etc/init.d/rpc stop

cd /etc/init.d
mv inetsvc __inetsvc.orig_install.no_start
mv rpc __rpc.orig_install.no_start
chmod 400 __inetsvc.orig_install.no_start
chmod 400 __rpc.orig_install.no_start
```

syslogd

That leaves no TCP listeners and one UDP listener:

```
  *.syslog                              Idle
```

The `man` page for `syslogd` indicates that the *default* behaviour is to listen.

```
/etc/init.d/syslog:            # ...and /etc/rc?.d/S??syslog

    ## /usr/sbin/syslogd >/dev/msglog 2>&1 &
    /usr/sbin/syslogd -t >/dev/msglog 2>&1 &


/etc/default/syslogd:

    LOG_FROM_REMOTE=NO         # ...from "YES"
```

and restart.

25. Example: Hardening a Linux Installation

...

```
  http://www.debian.org/doc/manuals/securing-debian-howto/
  http://tldp.org/LDP/solrhe/
```

...

26. Example: Finding an Intruder's Back Door

This section outlines how I found a backdoor on a Solaris box that I was attempting to secure for "a customer" — too late as it turned out. The notes below are a little rough — they are the notes I took at the time.

```
netstat -a | grep LIST

      *.sunrpc             *.*                 0      0     0      0 LISTEN
      *.32771              *.*                 0      0     0      0 LISTEN
      *.fs                 *.*                 0      0     0      0 LISTEN
      *.printer            *.*                 0      0     0      0 LISTEN
      *.32772              *.*                 0      0     0      0 LISTEN
      *.32773              *.*                 0      0     0      0 LISTEN
      *.22370              *.*                 0      0     0      0 LISTEN
      *.22102              *.*                 0      0     0      0 LISTEN
      *.32775              *.*                 0      0     0      0 LISTEN
      *.32776              *.*                 0      0     0      0 LISTEN
      *.32795              *.*                 0      0     0      0 LISTEN
      *.32816              *.*                 0      0     0      0 LISTEN
      *.32838              *.*                 0      0     0      0 LISTEN
      *.6000               *.*                 0      0     0      0 LISTEN
      *.32918              *.*                 0      0     0      0 LISTEN
      *.22                 *.*                 0      0     0      0 LISTEN
      *.6010               *.*                 0      0     0      0 LISTEN
bash-2.05b#
```

(Of course, if `netstat` had been trojanned, then the port that I was suspicious of above, 22102, would not have shown up — but using `nmap` to scan the machine *would* have found it.)

```
lsof_4.72/lsof_4.72_src/lsof | grep 22102
sshd2      285    root    3u  inet 0x30000315108   0t0    TCP *:22102 (LISTEN)



ls -l /usr/bin/sshd2
-rwxr-xr-x   1 root     root      264424 Mar 28  2002 /usr/bin/sshd2



bash-2.05b# lsof_4.72/lsof_4.72_src/lsof | grep sshd2
sshd2      285    root  cwd   VDIR          32,0       1536                      2 /
sshd2      285    root  txt   VREG          32,0     264424                 439751 /usr/bin/sshd2█
sshd2      285    root  txt   VREG          32,0    1126216                 141590 /usr/lib/libc.so.1█
sshd2      285    root  txt   VREG          32,0      17256                  90258 /usr/platform/sun4u/l
sshd2      285    root  txt   VREG          32,0     838700                 141177 /usr/lib/libnsl.so.1█
sshd2      285    root  txt   VREG          32,0      19876                 141106 /usr/lib/libmp.so.2█
sshd2      285    root  txt   VREG          32,0      14908                 141120 /usr/lib/libsec.so.1█
sshd2      285    root  txt   VREG          32,0      56988                 141121 /usr/lib/libsocket.so
sshd2      285    root  txt   VREG          32,0       5328                 141075 /usr/lib/libdl.so.1█
sshd2      285    root  txt   VREG          32,0     234132                 141118 /usr/lib/ld.so.1█
sshd2      285    root    0r  VCHR          13,2        0t0                 479352 /devices/pseudo/mm@0:
sshd2      285    root    1w  VCHR          13,2        0t0                 479352 /devices/pseudo/mm@0:
sshd2      285    root    2w  VCHR          13,2       0t42                 479352 /devices/pseudo/mm@0:
sshd2      285    root    3u  inet 0x30000315108        0t0                    TCP *:22102 (LISTEN)█
ssh



ls -lR / | grep "Mar 28  2002"

drwxrwxrwx   3 root     root         512 Mar 28  2002 0nb
drwxrwxrwx   2 root     root         512 Mar 28  2002 backup
-rwxr--r--   1 root     root        4032 Mar 28  2002 cleaner                        *** GOLD! *********
-rwxr-xr-x   1 root     root        9852 Mar 28  2002 pg
-r-xr-xr-x   1 root     root       10456 Mar 28  2002 in.fingerd
-rw-r--r--   1 root     root        5254 Mar 28  2002 inetd.conf
-r-xr-sr-x   1 root     root       50712 Mar 28  2002 ldlibnet.so
-r-sr-xr-x   1 root     root       29292 Mar 28  2002 login
-r-xr-sr-x   1 root     root       50712 Mar 28  2002 netstat
-r-sr-xr-x   1 root     root       18556 Mar 28  2002 su
-r-sr-xr-x   1 root     root       29292 Mar 28  2002 xlogin
-rw-rw-rw-   1 root     root          28 Mar 28  2002 ttyhash
-rw-r--r--   1 guest    staff        370 Mar 28  2002 CON.sixth.003.Z
-rw-r--r--   1 guest    staff        360 Mar 28  2002 CON.tenth.003.Z
-rwxrwxrwx   1 guest    staff        240 Mar 28  2002 SCRIPT.Z
-r-xr-xr-x   1 root     root      264424 Mar 28  2002 a.out
-rwxr--r--   1 root     sys         2544 Mar 28  2002 rc2
-rwxr--r--   1 root     sys         2001 Mar 28  2002 rc3
-r-sr-xr-x   1 root     root       29292 Mar 28  2002 xlogin
-rw-------   1 root     root         525 Mar 28  2002 ssh_host_key
-rw-r--r--   1 root     root         329 Mar 28  2002 ssh_host_key.pub
-rwxr-xr-x   1 root     root      264424 Mar 28  2002 sshd2
-rw-r--r--   1 root     root         461 Mar 28  2002 sshd_config
-r-xr-sr-x   1 root     root       50712 Mar 28  2002 ldlibnet.so
-rw-------   1 guest    staff          0 Mar 28  2002 wsconAAAmHaaym:0.0
bash-2.05b#



find / -name ldlibnet.so
```

```
/usr/lib/ldlibnet.so
/dev/rmt/0nb/backup/ldlibnet.so
```

27. Host-Based Intrusion Detection 0

Even a patched system, with wrapped services running a suitably configured packet filter, stands a significant chance of being intruded upon by an unwanted visitor.

27.1. Standard Host-Based Intrusion Detection

1. Keep *reliable* logs of what's going on on your system and read them! Log-scanning utilities and remote-logging are useful here.
2. Know what files have changed (e.g., `/bin/login`, `/bin/ps`...) so as to be able to detect an intrusion and effect a repair. To do this use a standard host-based intrusion-detection system (IDS) such as Tripwire which analyses files for signs or change. Other IDSs (e.g., Cheesewire[40]) monitor network and process activity in addition.

27.2. Rootkit Detection

1. An intruder who gains `root` access may install a *rootkit*, a set of system utilities and even kernel-related software designed both to exploit the machine/system, and to hide the presence of the exploitation and intrusion — such a rootkit can hide itself from standard IDSs: even running an IDS and all it depends on from read-only media is not good enough: *all* applications, including of course IDSs, rely on the integrity of the kernel, in particular the integrity of the system-calls. Any rootkit which changes the system-calls automatically comprises any non-kernel related IDS which is running under the OS.

We introduce methods of detecting rootkits below. For more on rootkits and how to detect them, see *Unix Security: More Stuff*[41].

27.3. Boot from a CD — "Its the only way to be sure"

The only sure-fire way of discovering a rootkit which changes the kernel (the running OS) in order to hide itself is to periodically boot the host from a CD-ROM containing an IDS and (minimal) OS and check the system disk(s) for intrusion, then boot back from disk as usual. This is hardly convenient for 24/7/265 servers, but it works.

28. Intrusion Detection I: Logging

To quote Mick Bauer:

Whatever else you do to secure a Linux system, it must have comprehensive, accurate, and carefully watched logs...they provide valuable early warning signs of system abuse. Third, after all else fails...system compromise..., logs can provide us with crucial forensic data.

[40] `../_cheesewire`
[41] `unix_security_k.tex`

Unix and Linux log messages come from all sort of places: the kernel itself, running daemons and services, the authentication subsystem, the email service, from the boot sequence... For the most part these messages are stored in files under

```
/var/log/               # all unices?
/var/adm/               # Solaris uses this too.
```

28.1. Syslog: `klogd` and `syslogd`

Almost every unix-like OS comes with Syslog. Sometimes this service is implemented by one daemon, `syslogd`; on Linux the service is implemented by two, `syslogd` and `klogd` — messages from the kernel are treated separately.

(Linux also separates out boot messages via the `bootlogd` service.)

28.1.1. Syslog Startup

Syslog is started at boot time in the usual way, by one or more `init` scripts:

```
/etc/init.d/sysklogd    #
/etc/init.d/klogd       # ...Linux

/etc/init.d/syslog      # ...Solaris
```

28.1.2. Syslog Configuration

Syslog is configured by one file, `/etc/syslog.conf`. A simple example, based on that which comes installed on a Debian box is given below. Some documentation is provided by the comments within; for more see the manpage: syslog.conf(5). N.B. *The whitespace between columns consists of TABS.*

```
# For more information see syslog.conf(5) manpage.
#
# -- All logs, split two ways (syslog contains everything, except for
#    authorisation-related messages which might contain passwords) :
#
auth,authpriv.*                        /var/log/auth.log
*.*;auth,authpriv.none                -/var/log/syslog


    # ...the "-" means don't sync, i.e., buffer output:  this is for busy
    #    log files, but can lead to missing or inconsistent messages.


# -- Log by facility (messages can be split by facility:  auth, auth-priv,
#    cron, daemon, kern, lpr, mail, mark, news, syslog, user, uucp and
#    local{0-7}) :
#
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


# -- Log by priority (each message has a priority:  debug, info, notice,
#    warning, err, crit, alert, emerg) : "debug", usually commented out,
#    grabs all (*) messages of priority (=) debug (except
#    authorisation-related messages);  "messages" gets all low priority
#    messages (info, notice, warn, except debug), except those from cron
#    and daemon :

#
##*.=debug;\
#        auth,authpriv.none      -/var/log/debug
#
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none         -/var/log/messages


# -- Emergencies are sent to everybody logged in (all screens/ptys).
#
*.emerg                          *


# -- The above are pretty standard;  these are more sys-admin-personal:
#    send messages to a virtual console (tty8 --- CTRL-ALT-F8 or ttysnoop);
#    send messages to /dev/xconsole (the pipe used by the xconsole utilitye) :
#
daemon,mail.*;\
        news.=crit;news.=err;news.=notice;\
        *.=debug;*.=info;\
        *.=notice;*.=warn        /dev/tty8
#
daemon.*;mail.*;\
        news.crit;news.err;news.notice;\
        *.=debug;*.=info;\
        *.=notice;*.=warn        |/dev/xconsole
```

```
# -- Copy logs to a couple of remote servers:
#
*.info                           @130.88.200.230
```

Logs should be as per the last few lines of the `syslog.conf` file, above.

N.B. The whitespace between the fields in `syslog.conf` *is made up of tabs, not spaces*. After creating the entries it is necessary to restart the Syslog daemon.

29. Securing Logs

An intruder will likely attempt to hide or delete evidence of their presence. First on their list will be to delete log messages which show the means by which they obtained access (e.g., a buffer overrun visible in `daemon.log`).

29.0.1. Append-Only Logging

ext2/3 attributes (chattr, lsattr); LIDS; SELinux. . .

29.0.2. Remote Logging

Syslog log daemons are able to copy messages to (and accept messages from) remote daemons — usually on UPD port 514. So it is easy to keep a secure copy of logs away from an intruder.

The `man` pages for `syslog` or `syslogd` describe how to set up remote logging. In short, ensure the daemon is started with the relevant switches: on Linux

```
/sbin/syslogd -r
```

enables *reception* and logging of events from the network; on Solaris no special switches are apparently required. To ensure messages are sent to a remote host, entries such as those below are required in `/etc/syslog.conf`:

```
auth.notice                                              @myhost.umist.ac.uk
*.info                                                   @myhost.umist.ac.uk
```

30. Log Monitoring Utilities

Manually watching these logs is frankly unintesting and time-consuming. Many utilities exist which automatically scan logs for signatures of suspicious events. These include:

- LogWatch[42];
- LogCheck/LogSentry can be downloaded from Psionic[43].
- Swatch[44].

Each is easily installed and configured. Reports are periodically emailed to the system administrator.

Output from log-watching utilities should be sent to a remote machine to minimise the risk of an intruder tampering with the reports, for example, LogWatch can be configured to do this — see `/etc/log.d/logwatch.conf` for details.

[42] `http://www.logwatch.org`
[43] `http://www.psionic.com/products/logsentry.html`
[44] `http://www.oit.ucsb.edu/~eta/swatch/`

31. A Central Log Server — `syslog-ng`

Sysadmins are far too busy (lazy) to monitor logs properly; intruders unhelpfully edit logs to hide their presence. Both issues can be addressed by setting up a log server: such a machine collects system logs from many client machines via UDP remote-logging [Page 52].

31.1. An aside: NTP

For both forensic and legal reasons, date/timestamps of log messages must be accurate. To achieve this, all log clients and the log-server should use NTP (or an equivalent service) to keep system clocks set accurately.

31.2. Syslog NG

Syslog can be used to collect logs from client machines, but is ill-suited to the task — it's showing its age: it is not obvious, for example, how to separate logs from different machines into different directories; with dozens or even hundreds of clients this is a must. Syslog NG[45] is a prime candidate to use instead.

31.2.1. Installation

Syslog-NG can be installed from source via the classic `.configure`, `make`, `make install` sequence; Debian packages are available.

31.2.2. Startup Options

Options which affect the security of the server include:

```
-u <username>  }  after initialisation drop root privileges and run
-g <group>     }  as <username>:<group>

-C /chroot/path    after reading the config file, chroot to path given
```

Having Syslog-NG listening as `root` is a very bad idea, so set up a suitable username and group, for example:

```
/etc/group
    syslogng:x:77:

/etc/passwd
    syslogng:x:99:65534::/var/chroot/syslogng:/bin/false
```

[45] `http://www.balabit.com/products/syslog_ng/`

31.2.3. Configuration of `syslog-ng`

The configuration file `/etc/syslog-ng/syslog-ng.conf` is divided into `options`, `sources`, `destinations`, `filters` and `logs`.
`options` affecting server security include:

```
    options {

        # -- resolve address of message sender?  show names of all hosts by
        #    which the message has been handled?
        use_dns(no);
        chain_hostnames(no);
        ##dns_cache(yes);
            # ...syslog-ng blocks on DNS queries, so enabling DNS may lead
            #    to a DOS, hence set both of the above to "no"...

        # -- enable or disable directory creation for destination files
        create_dirs(yes);
            # ...macros in the config file (see below) mean that new dirs
            #    will be created as packets from new IP addresses are
            #    encountered --- this could be an issue if packets are spoofed,
            #    but appropriate firewalling should resolve this issue...

        # default owner, group, and permissions for log files
        # (defaults are 0, 0, 0600)
        owner(syslogng);
        group(adm);
        perm(0640);

        # default owner, group, and permissions for created directories
        # (defaults are 0, 0, 0700)
        dir_owner(syslogng);
        dir_group(syslogng);
        dir_perm(0755);
  };
```

Here are couple of likely sources for a central log-server. First, we'll want to log our own messages plus standard stuff from our own host OS:

```
    source s_local {
        # -- messages generated by Syslog-NG :
        internal();

        # -- standard Linux log source (location to which syslog() function
        #    sends messages by default) :
        unix-stream("/dev/log");

        # -- messages from the kernel :
        file("/proc/kmsg" log_prefix("kernel: "));
      };
```

Secondly, we want to listen for incoming UDP-based messages from our clients:

```
source s_udp {
    # -- receive remote UDP logging messages (equivalent to the "-r"
    #    syslogd flag)
    udp(ip(0.0.0.0) port(514));
  };
```

Here is a particularly simple treatment of our own local messages:

```
destination df_auth    { file("/var/log/auth.log"); };
destination df_syslog  { file("/var/log/syslog");   };

filter f_auth   {     facility(auth, authpriv); };
filter f_syslog { not facility(auth, authpriv); };

# -- auth,authpriv.*                 /var/log/auth.log
log {
    source(s_local);
    filter(f_auth);
    destination(df_auth);
  };

 # -- *.*;auth,authpriv.none          -/var/log/syslog
 log {
    source(s_local);
    filter(f_syslog);
    destination(df_syslog);
  };
```

Local boot-related messages:

```
# -- local7.*                                         /var/log/boot.log

destination df_boot     { file("/var/log/boot.log"); };
filter      f_boot      { facility(local7); };

log {
    source(s_local);
    filter(f_boot);
    destination(df_boot);
  };
```

We can use Syslog-NG macros to help us in dealing with messages from our clients. We don't want them all jumbled together. This way each client gets its own directory:

```
destination hosts {
    file("/var/log_hosts_by_facility/$HOST/$FACILITY.log"
         owner(root) group(root) perm(0600) dir_perm(0700)
         create_dirs(yes));
  };

log {
    source(s_udp);
    destination(hosts);
  };
```

31.3. Firewalling

To help prevent a DOS attack on the server, suitable firewalling is essential. Here is the most basic:

```
$IPTABLES -t filter -A INPUT -p udp -s server.umist.ac.uk --dport 514 -j ACCEPT
$IPTABLES -t filter -A INPUT -p udp -s host.umist.ac.uk --dport 514 -j ACCEPT

$IPTABLES -t filter -P INPUT  DROP
```

31.4. Chrooting Syslog-NG

Ideally all services on a server which listen on a publicly accessible port `chroot` should be `chrooted` for security reasons. Syslog-NG is no exception.

Syslog-NG has a built-in option to `chroot` itself after binding to port 514. This means that the `chroot` environment to be built can be greatly simplified (which means less for a potential intruder to play with if they can get into the jail, and less for us to maintain): we do not need a shell and its supporting libraries, or many of the files `syslog-ng` opens on startup, before it `chroots` (use `lsof` to see that there are quite a few of these).

Mick Bauer has written a HOW-TO[46]. The words below are based on that: this version is simplified — since all DNS look-ups are turned off in our configuration (to eliminate a possible DOS attack), none of `/etc/resolv.conf`, `/etc/nsswitch.conf` and `libnss*.so.*` are required.

31.4.1. Building the `chroot` Environment

1. First, make necessary directories:

```
$LOGJAIL/etc/syslog-ng
$LOGJAIL/dev/
$LOGJAIL/var/log
```

[46] `http://www.campin.net/syslog-ng/faq.html`

where e.g. `$LOGJAIL=/var/chroot/syslog-ng`.

2. Next, ensure that you have a suitable — unprivileged — user/group for Syslog-NG to run as, so add lines something like this

```
syslogng:x:99:65534::/var/chroot/syslogng:/bin/false
syslogng:x:77:
```

to `$LOGJAIL/etc/passwd` and `$LOGJAIL/etc/group`, respectively — these can be the *only* entries in these files.

3. Move the Syslog-NG configuration file, `/etc/syslog-ng/syslog-ng.conf` to the jail and turn its old location in to a symlink:

```
cd /etc/syslog-ng
mv syslog-ng.conf $LOGJAIL/etc/syslog-ng
ln -s $LOGJAIL/etc/syslog-ng/syslog-ng.conf /etc/syslog-ng/syslog-ng.conf
```

4. Create a couple of required devices (actually these are necessary only if required by the `destinations` in `syslog-ng.conf`):

```
cd $LOGJAIL/dev
mknod -m 0660 xconsole p
mknod -m 0660 tty10 c 4 10
chgrp syslogng ./xconsole ./tty10
```

(`xconsole` is a pipe and is used as a `destination` — see `syslog-ng.conf`; `tty10` is a character device with major and minor numbers 4 and 10, respectively: it seems traditional to send `syslog` output to `tty10` — you can watch it using `CTRL-ALT-F10` locally, or using `ttysnoop`.)

5. Copy `/etc/localtime` to `$LOGJAIL/etc`.

Simple!

31.4.2. Starting Syslog-NG: an `init` Script

The following simple `init` script is all that's necessary to start `syslog-ng` and have it `chroot` itself:

```
    /etc/init.d/syslog-ng

      #! /bin/sh

      case "$1" in
        start)
          /sbin/syslog-ng -C /var/chroot/syslogng -u syslogng -g syslogng || ex\
it 1
          ;;
        *)
          echo "Usage: /etc/init.d/syslog-ng start" >&2
          exit 1
          ;;
      esac

      exit 0
```

31.4.3. Stopping Syslog-NG

```
pkill syslog-ng
```

31.4.4. Restarting Syslog-NG — *Not* `kill -HUP`

The usual way to restart `syslog-ng` after a configuration change (or after a log rotation) is to `kill -HUP` the process. This will not work in the above jail as the *already*-`chroot`ed process will not be able to bind to port 514 (a privileged port, as there is not `root` user in the jail), not are many required files available, e.g., `/dev/kmesg` and a bunch of shared-object libraries (use `lsof` to see which).

Assuming this actually presented a problem, one could build a more complex jail in which one could "manually" start `syslog-ng` (and in which it would therefore be restartable), like this:

```
chroot /var/chroot/syslog-ng bash
/sbin/syslog-ng -u syslogng -g syslogng
    # ...that's /var/chroot/syslog-ng/sbin/syslog-ng now...
```

31.5. Log Rotation: `logrotate`

Since `syslog-ng` cannot be restarted within its jail (see above), we have to use the `copytruncate` option when using `logrotate`, for example:

```
  /var/chroot/syslogng/var/log_hosts_by_facility/130.88.*/*.log {
      rotate 28
      daily
      copytruncate
      missingok
      notifempty
      compress
}
```

32. Intrusion Detection II: File Change/Signarute Monitoring

In addition to reading and perhaps changin data stored on your machine, an intruder may attempt to hide there work from you by altering the behaviour of utilities such as `df`, `du`, `ls`, `ps`, `netstat`... This can be done by replacing the excutables/utilities themselves, or libraries (`.sos`) they use, or even altering the system-calls used.

File system integrity (i.e., unwanted alterations to executables and libraries) can be monitored by using MD5 checksums of files and/or using Tripwire or a similar utility. Alteration of system-calls (of the kernel — or more likely kernel-modules) is a different kettle of fish entiresly, but tools are available to detect this also — see the next section.

32.1. Using MD5 Checksums — Home-brewed Systems

Tripwire (below) is a powerful too, but can be a lot of work to configure and use. An alternative is simply to write or use a simple script to check the integrity of important files (and to do this regularly).

Relying on the size and/or datestamp of a file is not enough — these can be "faked". A much more reliable method is to determine the current MD5 checksum of a file and compare it to the value when the file was known to be clean (e.g., immediately after installation of the OS).

Linux comes with `/usr/bin/md5sum` which can be used for this; alternatively, use the Perl module `Digest::MD5`. There are several freely-available utilties which use MD5 checksums to ensure file integrity including AIDE (Advanced Intrusion Detection Environment) and Cheesewire[47].

32.2. Tripwire

Let me quote from the RPM: *Tripwire answers the fundamental question: "Is my system the same today as it was yesterday?" Tripwire creates a cryptographically-secured database of files and their characteristics based on the specifications of your configurable policy file. This database is then used to determine if any unauthorized changes have been made to your system.*

A report of changes to individual files is periodically emailed to the system administrator.

There are both commercial and open-source (freely downloadable) versions of Tripwire in existence. The open source versions may be downloaded from `www.tripwire.org`[48]. For Linux, binary RPMS are available; for Solaris one must compile and link the source code.

Configuration of Tripwire is not as simple as it might be — it is very easy to get many false-positives. However, in the near future, ISD may be able to supply a Perl script which will help with this process.

Installation and Configuration of Tripwire on RedHat Linux

Install the RPM; edit `/etc/tripwire/twpol.txt` as appropriate; run `/etc/tripwire/tripwire.sh`; run `tripwire --init`. (Remove `/etc/tripwire/twpol.txt`).

If you wish to subsequently change the policy, edit `/etc/tripwire/twpol.txt`; run `twadin --create-polfile twpol.txt` to create the encrypted version of the policy; then run `tripwire --init`.

Sending Tripwire Output to a Remote Machine

Tripwire should be configured to send its output to a remote machine to minimise the risk of an intruder tampering with its reports. Tripwire sends its reports to `stdout`, so simply configure Cron to send its output to a remote machine.

[47] `../_cheesewire`

[48] `http://www.tripwire.org`

33. Cheesewire

Cheesewire[49] has Tripwire-like functionality (monitoring MD5s checksums, inode values, etc of files); in addition Cheesewire monitors all network connections all root-owned processes and compares to known signatures; and uses LSOF to check signatures of all processes and network connections.

33.1. Remarks on IDS Configuration

A poorly installed and configured IDS will run on rewritable media (e.g., harddisk) on a host, using standard OS utilities and libraries:

Home-Brewed MD5-based Systems

A simple IDS might consist of a set of Perl scripts which compare files, and network and process activity, to known/expected values. A poor installation of the IDS would rely on `/usr/bin/perl`, the contents of `/usr/lib/perl` and several shared-object files from `/lib` and `/usr/lib`. The trojanning of any of these will obviously invalidate the output/results from the IDS.

The solution is to have the IDS use its own Perl installation and its own copies of any shared-objects it uses — and use them and them alone. (This is what Cheesewire does.)

Tripwire

Tripwire avoids the problem of using trojanned shared-object libraries — its a statically-linked binary. However, there is always the possibility that Tripwire itself could be trojanned.

Remote and Read-Only Media

The only way to be sure that an IDS and any system utilities or libraries it uses are not trojanned is to install the whole lot, including all OS libraries and utilities used, on read-only media, such as on a CD ROM, and mount it from there (under the running OS).

This still leaves the question of what to do with the IDS-related database (of file size, inode, checksum, etc.) — rewriting the CD ROM each time database updates are required is not a great option. (One can use a database mounted from remote media, but this opens its own can of worms.)

34. Detecting Kernel-Altering (e.g., LKM) Rootkits: `chkrootkit`, `kstat`

A more acceptable (and interesting) alternative to booting from CD-ROM is to analyse kernel-related memory directly. Such analysis can detect LKM (loadable kernel module) rootkits. `chkrootkit` and `kstat` are commonly used tool to help with this.

[49] `../_cheesewire`

`chkrootkit`

This utility takes a completely different approach of MD5 checksums and Tripwire: this looks for standard symptoms of exploitation by an intruder, such as alterations to the network configuration (change to promiscuous mode) and deletion of entries from `lastlog` and `wtmp`, and also detections many known rootkits.

`chkrootkit` can be downloaded from [50] — simply `make` and run.

Set if to run daily via `cron` and email output off-host.

Rookit Hunter (`rkhunter`)

Rootkit Hunter[51]...

More

For considerably more on kernel-altering rootkits, see *Unix Security: K*[52].

35. Basics

Utilities exist which monitor ports and, in response to given criteria, e.g., an apparent denial-of-service attack, can block access to them from particular IP addresses or domains, by dynamically configuring TCP Wrappers and/or IPTables (and other software). Perhaps the most notable is PortSentry, which until very recently could be freely downloaded[53] from Psionic. However Psionic were recently bought by Cisco and the download is not available and Cisco are not responding to my email, but all is not lost. I have a local mirror[54].

36. Portsentry and PSAD

36.1. Portsentry

Portsentry is an easy-to-build-and-install relatively simple application that is designed to detect portscans and other suspicious (multiple) connectsion and, if so configured, block them by calling an external command (see below).

Portsentry was originally written by Psionic[55] who were acquired by Cisco. It is now available via Sourceforge[56].

Specifically, PortSentry runs as a daemon on the protected host. When running, it listens to TPC/UDP ports. PortSentry is very easy to configure — the configuration files live in `/etc/portsentry`.

It can be argued that binding to a load of ports like this is *bad*.

[50] `http://www.chkrootkit.org/`
[51] `http://www.rootkit.nl`
[52] `unix_security_k.tex`
[53] `http://www.psionic.com/`
[54] `_mirror`
[55] `http://www.psionic.com`
[56] `http://sourceforge.net/projects/sentrytools/`

The method by which Portsentry blocks hosts is configurable via the `KILL__ROUTE` command (in `portsentry.conf`, for example:

```
# -- Linux running IP Tables :
KILL_ROUTE="/sbin/iptables -I INPUT -s $TARGET$ -j DROP"
#
# -- Generic Solaris :
# KILL_ROUTE="/usr/sbin/route add $TARGET$ 333.444.555.666 1"
```

36.2. PSAD

PSAD is available under the GPL from Cipherdyne[57].

From the Web page:

PSAD is a collection of three lightweight system daemons. . . that run on Linux machines and analyze IPTables log messages to detect port scans and other suspicious traffic.

PSAD incorporates many signatures from the Snort intrusion detection system to detect probes for various backdoor programs. . ., DDoS tools. . ., and advanced port scans. . . When combined with FWSnort, PSAD is capable of detecting approximately 75

37. Snort and SnortSAM

37.1. Snort

http://www.snort.org

/usr/local/sbin/snort -d -D -c /usr/local/src/snort-2.2.0/etc/snort.conf

/usr/local/src/snort-2.2.0/rules/*

/var/log/snort

37.2. SnortSAM

SnortSAM[58] is a plugin for Snort which facilitates the automated blocking of IP addresses on the following firewalls: Checkpoint Firewall-1, Cisco PIX (and router ACLs), IPFilter, (OpenBSD) PF, IPChains, IPTables. . .

Binary executables are available from the Web site for Windows, Linux and FreeBSD. Alternatively, download the source code for Snort, SnortSAM and also the SnortSAM patches for Snort, and follow the instructions in the `INSTALL` file from SnortSAM.

[57] `http://www.cipherdyne.com/psad/`
[58] `http://www.snortsam.net`

Suggestions and hints:

SnortSAM

There are two executables — `snortsam` and `snortsam-debug`. Use the latter first — messages are output indicating the Snort alerts being processed.

Snort

Don't use "`-A fast`" — alerts are then sent to `/var/log/snort/alerts` but *not* to SnortSAM (which listens on port 898).
To avail yourself of helpful (debugging) messages, use

```
 /usr/local/sbin/snort -d -c /usr/local/src/snort-2.2.0/etc/snort.conf
```

(no `-D`) with the following in `snort.conf`:

```
output alert_fwsam: 130.88.100.77
output alert_syslog: log_auth log_warn
```

When all is well:

```
/usr/local/sbin/snort -d -D -c /usr/local/src/snort-2.2.0/etc/snort.conf
```

38. Netfilter/IPTables PSD Patch

39. Linux Security-Related Kernel Stuff

. . . is beyond the scope of this introduction. **See this instead**[59]. (Such stuff includes: SELinux, LIDS, AppArmor, GRSecurity. . .)

About this page:

Produced from the SGML: `/home/isd/public_html/_unix_security/_reml_grp/unix_security_intro.reml`

On: 23/11/2006 at 13:47:59

Options: `reml2 -l long -o tex -p single`

[59] `../../~mc/_unix_security/unix_sec_kernel_stuff.html`