

## Hardening Vanilla Linux Systems

Tim Bell  
System Administrator, Trinity College  
bhat@trinity.unimelb.edu.au

It's 10:00pm...

... do you know what your computer's running?

---

---

Good security:  
being able to answer "Yes"

---

---

Goal of talk:  
To teach sysadmins to secure Linux systems.

## Seminar overview

- Security overview:
  - requirements assessment
  - plan, implement, verify and monitor
- Security prerequisites
- Hardening:
  - booting, kernel and core OS
  - services
  - filesystems
  - access control
  - maintenance
  - special requirements
- Difficulties
- Further reading

## Security overview

"A computer is secure if you can depend on it and its software to behave as you expect." -- Garfinkel and Spafford

Specifying expectations is critical.

## Security overview

Requirement assessment:

- what is the computer used for?
- what are you protecting against?
- threats to consider:
  - unauthorised access to data
  - unauthorised modification or deletion of data
  - denial of service
  - becoming an attack launchpad
- what level of security is required?
  - least priviledge
  - balance security with usability
  - balance security with cost of implementation
- security policy provides guidance

## Security steps

Plan:

- choose hardware and software to meet needs
- determine configuration
- plan deployment

Implement:

- install hardware
- install software
- ...including hardening

Verify, monitor and maintain:

- verify deployment meets requirements
- monitor on ongoing basis
- maintain system with security patches

## Security prerequisites

---

- physical location:
  - physical access => no security
- network:
  - firewall: first line of defence
  - no unauthorised or rogue hosts
- network services:
  - no insecure services:
    - telnet, ftp, rsh, NIS, NFS
  - secure central services:
    - authentication
- security policy
- management buy-in
- good knowledge of Linux => there is no magic recipe

## Hardening

---

Hardening is the "process of modifying a system to make it highly secure" (AEleen Frisch).

(Shouldn't it have been secure in the first place?)

Hardening is based on the idea of **least privilege**:

- no unneeded services
  - no unneeded access
- and **layers of security**:
- failure of one layer is not fatal

Hardening must start from known good state:

- fresh install
- no network until finished

## Booting

---

Aim: only boot specified image and with specified options

Configure BIOS:

- boot only from hard drive
- password protect settings

Boot loading:

- prevent lilo from prompting (or password protect)

(If they can boot with `init=/bin/sh`, or to another kernel, it's game over.)

## Kernel

---

Build a custom minimal kernel:

- enable process accounting
- disable unused features, eg:
  - networking protocols such as IPX, Appletalk
  - support for hardware not present

Principle of least privilege: don't give the kernel capabilities which are not required.

Also, bugs in code not compiled in don't matter.

## Core OS

---

Try to install an absolute minimum number of packages.

Safer to add as required than remove:

- packages may not remove cleanly
- a package may contain a trojan
- an unwanted package may be overlooked
- wastes less time

List installed packages with:

- `dpkg -l`
- `rpm -q`

## Core OS (cont.)

---

- enable shadow passwords
- enable md5 password hashes
- set good root password
- require root password to enter single user mode
- set up sudo for everyday use
- disable ctrl-alt-del if console not physically secure
- setup and tune logging
- use NTP so log timestamps are accurate (if you need them after intrusion)

## Services

---

- remove unneeded services; disabling not preferred
- run in chroot jails (if possible)
- avoid running as root
- use per-service users (e.g. web for apache)
- restrict access to services using
  - service's configuration
  - TCP wrappers
  - IP Tables
- apply process and file limits
- configure appropriate logging

## Identifying services

---

Many ways to find which services exist:

- look though boot scripts in:
  - /etc/init.d, /etc/rcN.d, /etc/rc.boot (Debian)
  - /etc/init.d, /etc/rc.d/rcN.d (RedHat)
- read internet superserver configuration:
  - /etc/inetd.conf
  - /etc/xinetd.d/
- check services started by `init` in `/etc/inittab`
- look in `crontab` files
- see what's listening on sockets:
  - `netstat -l`
  - `nmap`

Some services aren't network services.

## Services to remove or disable

---

Depends on requirements (obviously)

- network filesystems (nfsd, samba, netatalk)
- printing (lpd, lprng, ppr, cups)
- mail-related (sendmail, qmail, exim, postfix, pop, imap)
- remote access (telnetd, ftpd, tftpd, rshd, rlogind, talkd, fingerd)
- testing services (echo, daytime, chargin, discard, time)
- name resolution (named, djbdns)
- X (if being used as a server)
- build tools (gcc, make) and sources (kernel)

It's better if these are never installed in the first place.

## Filesystems

---

Apply least-privilege principle:

- file and directory ownership and permissions:
  - world-writable
  - world-readable
  - incorrect ownership
- special bits (setuid, setgid, sticky)
  - avoid setuid or setgid programs if possible
- mount options
  - noexec (except for `/usr`)
  - nodev (except for `/dev` (or `/`))
  - read-only

## Access control

---

Restrict remote access:

- server config (e.g. `access.conf`)
- TCP wrappers
- IP Tables
- password protection (over secure channel)

Restrict local access:

- require good user passwords
- regular password changing (pros and cons)
- group membership

Again, principle of least privilege applies.

## Ensuring the system stays 'hard'

---

- make backups
- keep up-to-date with patches
- read distribution security mailing list
- read BugTraq and (Aus)CERT if keen
- monitor:
  - nmap
  - netstat
  - tripwire
  - tiger

## Special requirements

---

### Servers:

- no user shell logins
- service monitoring (nagios)

### Workstations:

- don't give users root password
- provide sudo access to selected commands if necessary
- disable non-local X access
- applications: consider as services

### Labs:

- requirements for workstations also apply
- prevent use of removable media if possible
- disallow remote access from other lab computers

## Difficulties

---

- lack of time: "Anyone can install RedHat in just a few minutes! What's taking you so long?"
- security/usability conflicts: e.g. frequent password entry
- reluctant/obstinate users: "But I've always used telnet!"
- reduction in flexibility
- lack of management support

## Further reading

---

- Hardening Linux Systems series (AEleen Frisch):
  - [http://www.linux-mag.com/2002-09/guru\\_01.html](http://www.linux-mag.com/2002-09/guru_01.html)
  - [http://www.linux-mag.com/2002-10/guru\\_01.html](http://www.linux-mag.com/2002-10/guru_01.html)
  - [http://www.linux-mag.com/2002-11/guru\\_01.html](http://www.linux-mag.com/2002-11/guru_01.html)
  - [http://www.linux-mag.com/downloads/2002-09/guru/harden\\_list.htm](http://www.linux-mag.com/downloads/2002-09/guru/harden_list.htm)
- The Process of Hardening Linux (Chris Koutras, SANS):  
<http://www.sans.org/rr/linux/hardening.php>

## Further reading

---

- "Practical Unix and Internet Security", 2nd Edition, Simson Garfinkel, Gene Spafford
- "Real World Linux Security", 2nd Edition, Bob Toxen
- "Essential System Administration", 3rd Edition, AEleen Frisch
- "Linux Administration Handbook", Evi Nemeth, Garth Snyder, Trent R. Hein
- "The Practice of System and Network Administration", Thomas A. Limoncelli, Christine Hogan