# UNIX Administration Course

**Copyright 1999 by Ian Mapleson BSc.**

**Version 1.0**

mapleson@gamers.org
Tel: (+44) (0)1772 893297
Fax: (+44) (0)1772 892913
WWW: http://www.futuretech.vuurwerk.nl/

**Detailed Notes for Day 3 (Part 4)**

## UNIX Fundamentals: Security and Access Control.

### General Security.

Any computer system must be secure, whether it's connected to the Internet or not. Some issues may be irrelevant for Intranets (isolated networks which may or may not use Internet-style technologies), but security is still important for any internal network, if only to protect against employee grievances or accidental damage. Crucially, a system should not be expanded to include external network connections until internal security has been dealt with, and individual systems should not be added to a network until they have been properly configured (unless the changes are of a type which cannot be made until the system is physically connected).

However, security is not an issue which can ever be finalised; one must constantly maintain an up-to-date understanding of relevant issues and monitor the system using the various available tools such as 'last' (display recent logins; there are many other available tools and commands).

In older UNIX variants, security mostly involved configuring the contents of various system/service setup files. Today, many UNIX OSs offer the admin a GUI-frontend security manager to deal with security issues in a more structured way. In the case of SGI's IRIX, version 6.5 has such a GUI tool, but 6.2 does not. The GUI tool is really just a convenient way of gathering together all the relevant issues concerning security in a form that is easier to deal with (ie. less need to look through man pages, online books, etc.) The security issues themselves are still the same.

UNIX systems have a number of built-in security features which offer a reasonably acceptable level of security without the need to install any additional software. UNIX gives users a great deal of flexibility in how they manage and share their files and data; such convenience may be incompatible with an ideal site security policy, so decisions often have to be taken about how secure a system is going to be - the more secure a system is, the less flexible for users it becomes.

Older versions of any UNIX variant will always be less secure than newer ones. If possible, an admin should always try and use the latest version in order to obtain the best possible default security. For example, versions of IRIX as old as 5.3 (circa 1994) had some areas of subtle system functionality rather open by default (eg. some feature or service turned on), whereas versions later than 6.0 turned off the features to improve the security of a default installation - UNIX vendors began making these changes in order to comply with the more rigorous standards demanded by the Internet age.

Standard UNIX security features include:

```
1. File ownership,
2. File permissions,
3. System activity monitoring tools, eg. who, ps, log files,
4. Encryption-based, password-protected user accounts,
5. An encryption program (crypt) which any user can exploit.
```

**Figure 60. Standard UNIX security features.**

All except the last item above have already been discussed in previous lectures.

The ’crypt’ command can be used by the admin and users to encrypt data, using an encryption key supplied as an argument. Crypt employs an encryption schema based on similar ideas used in the German ’Enigma’ machine in WWII, although crypt’s implementation of the mathematical equivalent is much more complex, like having a much bigger and more sophisticated Enigma machine. Crypt is a satisfactorily secure program; the man page says, "Methods of attack on such machines are known, but not widely; moreover the amount of work required is likely to be large."

However, since crypt requires the key to be supplied as an argument, commands such as ps could be used by others to observe the command in operation, and hence the key. This is crypt’s only weakness. See the crypt man page for full details on how crypt is used.

**Responsibility.**

Though an admin has to implement security policies and monitor the system, ordinary users are no less responsible for ensuring system security in those areas where they have influence and can make a difference. Besides managing their passwords carefully, users should control the availability of their data using appropriate read, write and execute file permissions, and be aware of the security issues surrounding areas such as accessing the Internet.

Security is not just software and system files though. Physical aspects of the system are also important and should be noted by users as well as the admin.

Thus:

- Any item not secured with a lock, cable, etc. can be removed by anyone who has physical access.

- Backups should be securely stored.

- Consider the use of video surveillance equipment and some form of metal-key/key-card/numeric-code entry system for important areas.

- Account passwords enable actions performed on the system to be traced. All accounts should have passwords. Badly chosen passwords, and old passwords, can compromise security. An admin should consider using password-cracking software to ensure that poorly chosen passwords are not in use.

- Group permissions for files should be set appropriately (user, group, others).

- Guest accounts can be used anonymously; if a guest account is necessary, the tasks which can be carried out when logged in as guest should be restricted. Having open guest accounts on multiple systems which do not have common ordinary accounts is unwise - it allows users to anonymously exchange data between such systems when their normal accounts would not allow them to do so. Accounts such as guest can be useful, but they should be used with care, especially if they are left with no password.

- Unused accounts should be locked out, or backed up and removed.

- If a staff member leaves the organisation, passwords should be changed to ensure such former users do not retain access.

- Sensitive data should not be kept on systems with more open access such as anonymous ftp and modem dialup accounts.

- Use of the su command amongst users should be discouraged. Its use may be legitimate, but it encourages lax security (ordinary users have to exchange passwords in order to use su). Monitor the /var/adm/sulog file for any suspicious use of su.

- Ensure that key files owned by a user are writeable only by that user, thus preventing ’trojan horse’ attacks. This also applies to root-owned files/dirs, eg. /, /bin, /usr/bin, /etc, /var, and so on. Use find and other tools to locate directories that are globally writeable - if such a directory is a user’s home directory, consider contacting the user for further details as to why their home directory has been left so open. For added security, use an account-creation schema which sets users’ home directories to not be readable by groups or others by default.

- Instruct users not to leave logged-in terminals unattended. The xlock command is available to secure an unattended workstation but its use for long periods may be regarded as inconsiderate by other users who are not able to use the terminal, leading to the temptation of rebooting the machine, perhaps causing the logged-in user to lose data.

- Only vendor-supplied software should be fully trusted. Commercial 3rd-party software should be ok as long as one has confidence in the supplier, but shareware or freeware software must be treated with care, especially if such software is in the form of precompiled ready-to-run binaries (precompiled non-vendor software might contain malicious code). Software distributed in source code form is safer, but caution is still required, especially if executables have to be owned by root and installed using the set-UID feature in order to run. Set-UID and set-GID programs have legitimate uses, but because they are potentially harmful, their presence on a system should be minimised. The find command can be used to locate such files, while older file system types (eg. EFS) can be searched with commands such as ncheck.

- Network hardware can be physically tapped to eavesdrop on network traffic. If security must be particularly tight, keep important network hardware secure (eg. locked cupboard) and regularly check other network items (cables, etc.) for any sign of attack. Consider using specially secure areas for certain hardware items, and make it easy to examine cabling if possible (keep an up-to-date printed map to aid checks). Fibre-optic cables are harder to interfere with, eg. FDDI. Consider using video surveillance technologies in such situations.

- Espionage and sabotage are issues which some admins may have to be aware of, especially where commercially sensitive or government/police-related work data is being manipulated. Simple example: could someone *see* a monitor screen through a window using a telescope? What about RF radiation? Remote scanners can pickup stray monitor emissions, so consider appropriate RF shielding (Faraday Cage). What about insecure phone lines? Could someone, even an ordinary user, attach a modem to a system and dial out, or allow someone else to dial in?

- Keep up-to-date with security issues; monitor security-related sites such as www.rootshell.com, UKERNA, JANET, CERT, etc. [7]. Follow any extra advice given in vendor-specific security FAQ files (usually posted to relevant ’announce’ or ’misc’ newsgroups, eg. comp.sys.sgi.misc). Most UNIX vendors also have an anonymous ftp site from which customers can obtain security patches and other related information. Consider joining any specialised mailing lists that may be available.

- If necessary tasks are beyond one’s experience and capabilities, consider employing a vendor-recommended external security consultancy team.

- Exploit any special features of the UNIX system being used, eg. at night, an Indy’s digital camera could be used to send single frames twice a second across the network to a remote system for subsequent compression, time-stamping and recording. NB: this is a real example which SGI once helped a customer to do in order to catch some memory thieves.

**Figure 61. Aspects of a system relevant to security.**

Since basic security on UNIX systems relies primarily on login accounts, passwords, file ownership and file permissions, proper administration and adequate education of users is normally sufficient to provide adequate security for most sites. Lapses in security are usually caused by human error, or improper use of system security features. Extra security actions such as commercial security-related software are not worth considering if even basic features are not used or are compromised via incompetence.

An admin can alter the way in which failed login attempts are dealt with by configuring the /etc/default/login file. There are many possibilities and options - see the ’login’ reference page for details (man login). For example, an effective way to enhance security is to make repeated guessing of account passwords an increasingly slow process by penalising further login attempts with ever increasing delays between login failures. Note that GUI-based login systems may not support features such as this, though one can always deactivate them via an appropriate chkconfig command.

Most UNIX vendors offer the use of hardware-level PROM passwords to provide an extra level of security, ie. a password is required from a users who attempts to gain access to any low-level hardware PROM-based ’Command Monitor’, giving greater control over who can carry out admin-level actions. While PROM passwords cannot prevent physical theft (eg. someone stealing a disk and accessing its data by installing it as an option drive on another system), they do limit the ability of malicious users to boot a system using their own program or device (a common flaw with Mac systems), or otherwise harm the system at its lowest level. If the PROM password has been forgotten, the root user can reset it. If both are lost, then one will usually have to resort to setting a special jumper on the system motherboard, or temporarily removing the PROM chip altogether (the loss of power to the chip resets the password).

## Shadow Passwords

If the /etc/passwd file can be read by users, then there is scope for users to take a copy away to be brute-force tested with password-cracking software. The solution is to use a shadow password file called /etc/shadow - this is a copy of the ordinary password file (/etc/passwd) which cannot be accessed by non-root users. When in use, the password fields in /etc/passwd are replaced with an ’x’. All the usual password-related programs work in the same way as before, though shadow passwords are dealt with in a different way for systems using NIS (this is because NIS keeps all password data for ordinary users in a different file called /etc/passwd.nis). Users won’t notice any difference when shadow passwords are in use, except that they won’t be able to see the encrypted form of their password anymore.

The use of shadow passwords is activated simply by running the ’pwconv’ program (see the man page for details). Shadow passwords are in effect as soon as this command has been executed.

## Password Ageing.

An admin can force passwords to age automatically, ensuring that users must set a new password at desired intervals, or no earlier than a certain interval, or even immediately. The passwd command is used to control the various available options. Note that NIS does not support password ageing.

## Choosing Passwords.

Words from the dictionary should not be used, nor should obvious items such as film characters and titles, names of relatives, car number plates, etc. Passwords should include obscure characters, digits and punctuation marks. Consider using and mixing words from other languages, eg. Finnish, Russian, etc.

An admin should not use the same root password for more than one system, unless there is good reason.

When a new account is created, a password should be set there and then. If the user is not immediately present, a default password such as ’password’ might be used in the expectation that the user will login in immediately and change it to something more suitable. An admin should lockout the account if the password isn’t changed after some duration: replace the password entry for the user concerned in the /etc/passwd file with anything that contains at least one character that is not used by the encryption schema, eg. ’*’.

Modern UNIX systems often include a minimum password length and may insist on certain rules about what a password can be, eg. at least one digit.

## Network Security.

As with other areas of security, GUI tools may be available for controlling network-related security issues, especially those concerning the Internet. Since GUI tools may vary between different UNIX OSs, this discussion deals mainly with the command line tools and related files.

Reminder: there is little point in tightening network security if local security has not yet been dealt with, or is lax.

Apart from the /etc/passwd file, the other important files which control network behaviour are:

```
     /etc/hosts.equiv          A list of trusted hosts.
```

```
.rhosts                   A list of hosts that are allowed
                          access to a specific user account.
```

**Figure 62. Files relevant to network behaviour.**

These three files determine whether a host will accept an access request from programs such as rlogin, rcp, rsh, or rdist. Both hosts.equiv and .rhosts have reference pages (use ’man hosts.equiv’ and ’man rhosts’).

Suppose a user on host A attempts to access a remote host B. As long as the hosts.equiv file on B contains the host name of A, and B’s /etc/passwd lists A’s user ID as a valid account, then no further checks occur and the access is granted (all successful logins are recorded in /var/adm/SYSLOG). The hosts.equiv file used by the Ve24 Indys contains the following:

```
localhost
yoda.comp.uclan.ac.uk
akira.comp.uclan.ac.uk
ash.comp.uclan.ac.uk
cameron.comp.uclan.ac.uk
chan.comp.uclan.ac.uk
conan.comp.uclan.ac.uk
gibson.comp.uclan.ac.uk
indiana.comp.uclan.ac.uk
leon.comp.uclan.ac.uk
merlin.comp.uclan.ac.uk
nikita.comp.uclan.ac.uk
ridley.comp.uclan.ac.uk
sevrin.comp.uclan.ac.uk
solo.comp.uclan.ac.uk
spock.comp.uclan.ac.uk
stanley.comp.uclan.ac.uk
warlock.comp.uclan.ac.uk
wolfen.comp.uclan.ac.uk
woo.comp.uclan.ac.uk
milamber.comp.uclan.ac.uk
```

**Figure 63. hosts.equiv files used by Ve24 Indys.**

Thus, once logged into one of the Indys, a user can rlogin directly to any of the other Indys without having to enter their password again, and can execute rsh commands, etc. A staff member logged into Yoda can login into any of the Ve24 Indys too (students cannot do this).

The hosts.equiv files on Yoda and Milamber are completely different, containing only references to each other as needed. Yoda’s hosts.equiv file contains:

```
localhost
milamber.comp.uclan.ac.uk
```

**Figure 64. hosts.equiv file for yoda.**

Thus, Yoda trusts Milamber. However, Milamber’s hosts.equiv only contains:

```
localhost
```

**Figure 65. hosts.equiv file for milamber.**

ie. Milamber doesn’t trust Yoda, the rationale being that even if Yoda’s root security is compromised, logging in to Milamber as root is blocked. Hence, even if a hack attack damaged the server and Ve24 clients, I would still have at least one fully functional secure machine with which to tackle the problem upon its discovery.

Users can extend the functionality of hosts.equiv by using a .rhosts file in their home directory, enabling or disabling access based on host names, group names and specific user account names.

The root login only uses the /.rhosts file if one is present - /etc/hosts.equiv is ignored.

NOTE: an entry for root in /.rhosts on a local system allows root users on a remote system to gain local root access. Thus, including the root name in /.rhosts is unwise. Instead, file transfers can be more securely dealt with using ftp via a guest account, or through an NFS-mounted directory. An admin should be very selective as to the entries included in root's .rhosts file.

A user's .rhosts file must be owned by either the user or root. If it is owned by anyone else, or if the file permissions are such that it is writeable by someone else, then the system ignores the contents of the user's .rhosts file by default.

An admin may decide it's better to bar the use of .rhosts files completely, perhaps because an external network of unknown security status is connected. The .rhosts files can be barred by adding a -l option to the rshd line in /etc/inetd.conf (use 'man rshd' for further details).

Thus, the relationship between the 20 different machines which form the SGI network I run is as follows:

- All the Indys in Ve24 trust each other, as well as Yoda and Milamber.
- Yoda only trusts Milamber.
- Milamber doesn't trust any system.

With respect to choosing root passwords, I decided to use the following configuration:

- All Ve24 systems have the same root password and the same PROM password.
- Yoda and Milamber have their own separate passwords, distinct from all others.

This design has two deliberate consequences:

- Ordinary users have flexible access between the Indys in Ve24,
- If the root account of any of the Ve24 Indys is compromised, the unauthorised user will not be able to gain access to Yoda or Milamber as root. *However*, the use of NFS compromises such a schema since, for example, a root user on a Ve24 Indy could easily alter any files in /home, /var/mail, /usr/share and /mapleson.

With respect to the use of identical root and PROM passwords on the Ve24 machines: because Internet access (via a proxy server) has recently been setup for users, I will probably change the schema in order to hinder brute force attacks.

## The /etc/passwd File and NIS.

The NIS service enables users to login to a client by including the following entry as the last line in the client's /etc/passwd file:

```
+::0:0:::
```

**Figure 66. Additional line in /etc/passwd enabling NIS.**

For simplicity, a + on its own can be used. I prefer to use the longer version so that if I want to make changes, the fields to change are immediately visible.

If a user logs on with an account ID which is not listed in the /etc/passwd file as a local account, then such an entry at the end of the file instructs the system to try and get the account information from the NIS server, ie. Yoda. Since Yoda and Milamber do not include this extra line in /etc/passwd, students cannot login to them with their own ID anyway, no matter the contents of .rhosts and hosts.equiv.

## inetd and inetd.conf

inetd is the 'Internet Super-server'. inetd listens for requests for network services, executing the appropriate program for each request.

inetd is started on bootup by the /etc/init.d/network script (called by the /etc/rc2.d/S30network link via the init process). It reads its configuration information from /etc/inetd.conf.

By using a super-daemon in this way, a single daemon is able to invoke other daemons when necessary, reducing system load and using resources such as memory more efficiently.

The /etc/inetd.conf file controls how various network services are configured, eg. logging options, debugging modes, service restrictions, the use of the bootp protocol for remote OS installation, etc. An admin can control services and logging behaviour by customising this file. A reference page is available with complete information ('man inetd').

Services communicate using 'port' numbers, rather like separate channels on a CB radio. Blocking the use of certain port numbers is a simple way of preventing a particular service from being used. Network/Internet services and their associated port numbers are contained in the /etc/services database. An admin can use the 'fuser' command to identify which processes are currently using a particular port, eg. to see the current use of TCP port 25:

```
fuser 25/tcp
```

On Yoda, an output similar to the following would be given:

```
yoda # fuser 25/tcp
25/tcp:      855o
yoda # ps -ef | grep 855 | grep -v grep
root   855     1  0   Apr 27 ?       5:01 /usr/lib/sendmail -bd -q15m
```

**Figure 67. Typical output from fuser.**

Insert (a quick example of typical information hunting): an admin wants to do the same on the ftp port, but can't remember the port number. Solution: use grep to find the port number from /etc/services:

```
yoda 25# grep ftp /etc/services
ftp-data        20/tcp
ftp             21/tcp
tftp            69/udp
sftp            115/tcp
yoda 26# fuser 21/tcp
21/tcp:      255o
yoda 28# ps -ef | grep 255 | grep -v grep
   root   255     1  0   Apr 27 ?       0:04 /usr/etc/inetd
 senslm   857   255  0   Apr 27 ?      11:44 fam
   root 11582   255  1 09:49:57 pts/1   0:01 rlogind
```

An important aspect of the inetd.conf file is the user name field which determines which user ID each process runs under. Changing this field to a less privileged ID (eg. nobody) enables system service processes to be given lower access permissions than root, which may be useful for further enhancing security. Notice that services such as http (the WWW) are normally already set to run as nobody. Proxy servers should also run as nobody, otherwise http requests may be able to retrieve files such as /etc/passwd (however, some systems may have the nobody user defined so that it cannot run programs, so another user may have to be used - an admin can make one up).

Another common modification made to inetd.conf in order to improve security is to restrict the use of the finger command, eg. with -S to prevent login status, home directory and shell information from being given out. Or more commonly the -f option is used which forces any finger request to just return the contents of a file, eg. yoda's entry for the finger service looks like this:

```
finger stream tcp nowait guest /usr/etc/fingerd fingerd -f /etc/fingerd.message
```

**Figure 68. Blocking the use of finger in the /etc/inetd.conf file.**

Thus, any remote user who executes a finger request to yoda is given a brief message [3].

If changes are made to the inetd.conf file, then inetd must be notified of the changes, either by rebooting the system or via the following command (which doesn't require a reboot afterwards):

```
killall -HUP inetd
```

**Figure 69. Instructing inetd to restart itself (using killall).**

In general, a local trusted network is less likely to require a highly restricted set of services, ie. modifying inetd.conf becomes more important when connecting to external networks, especially the Internet. Thus, an admin should be aware that creating a very secure inetd.conf file on an isolated network or Intranet may be unduly harsh on ordinary users.

## X11 Windows Network Access

The X Windows system is a window system available for a wide variety of different computer platforms which use bitmap displays [8]. Its development is managed by the X Consortium, Inc. On SGI IRIX systems, the X Windows server daemon is called 'Xsgi' and conforms to Release 6 of the X11 standard (X11R6).

The X server, Xsgi, manages the flow of user/application input and output requests to/from client programs using a number of interprocess communication links. The xdm daemon acts as the display manager. Usually, user programs are running on the same host as the X server, but X Windows also supports the display of client programs which are actually running on remote hosts, even systems using completely different OSs and hardware platforms, ie. X is network-transparent.

The X man page says:

```
"X supports overlapping hierarchical subwindows and text and
graphics operations, on both monochrome and color displays."
```

One unique side effect of this is that access to application mouse menus is *independent* of application focus, requiring only a single mouse click for such actions. For example, suppose two application windows are visible on screen:

- a jot editor session containing an unsaved file (eg. /etc/passwd.nis),
- a shell window which is partially obscuring the jot window.

With the shell window selected, the admin is about to run /var/yp/ypmake to reparse the password database file, but realises the file isn't saved. Moving the mouse over the partially hidden jot window, the admin holds down the right mouse button: this brings up jot's right-button menu (which may or may not be partly ontop of the shell window even though the jot window is at the back) from which the admin clicks on 'Save'; the menu disappears, the file is saved, but the shell window is still on top of the jot window, ie. their relative front/back positions haven't changed during the operation.

The ability of X to process screen events independently of which application window is currently in focus is a surprisingly useful time-saving feature. Every time a user does an action like this, at least one extraneous mouse click is prevented; this can be shown by comparing to MS Windows interfaces:

- Under Win95 and Win98, trying to access an application's right-button menu when the application's window is currently not in focus requires at least two extraneous mouse clicks: the first click brings the application in focus (ie. to the front), the second brings up the menu, and a third (perhaps more if the original application window is now completely hidden) brings the original application window back to the front and in focus. Thus, X is at least 66% more efficient for carrying out this action compared to Win95/Win98.
- Under WindowsNT, attempting the same action requires at least one extraneous mouse click: the first click brings the application in focus and reveals the menu, and a second (perhaps more, etc.) brings the original application window back to the front and in focus. Thus, X is at least 50% more efficient for carrying out this action compared to NT.

The same effect can be seen when accessing middle-mouse menus or actions under X, eg. text can be highlighted and pasted to an application with the middle-mouse button even when that application is not in focus and not at the front. This is a classic example of how much more advanced X is over Microsoft's GUI interface technologies, even though X is now quite old. X also works in a way which links to graphics libraries such as OpenGL.

Note that most UNIX-based hardware platforms use video frame buffer configurations which allow a large number of windows to be present without causing colour map swapping or other side effects, ie. the ability to have multiple overlapping windows is a feature supported in hardware, eg. Indigo2 [6].

X is a widely used system, with emulators available for systems which don't normally use X, eg.

Windows Exceed for PCs.

Under the X Window System, users can run programs transparently on remote hosts that are part of the local network, and can even run applications on remote hosts across the Internet with the windows displayed locally if all the various necessary access permissions have been correctly set at both ends. An ’X Display Variable’ is used to denote which host the application should attempt to display its windows on. Thus, assuming a connection with a remote host to which one had authorised telnet access (eg. haarlem.vuurwerk.nl), from a local host whose domain name is properly visible on the Internet (eg. thunder.uclan.ac.uk), then the local display of applications running on the remote host is enabled with a command such as:

```
    haarlem% set DISPLAY = thunder.uclan.ac.uk:0.0
```

I’ve successfully used this method while at Heriot Watt to run an xedit editor on a remote system in England but with the xedit window itself displayed on the monitor attached to the system I was physically using in Scotland.

The kind of inter-system access made possible by X has nothing to do with login accouns, passwords, etc. and is instead controlled via the X protocols. The ’X’ man page has full details, but note: the man page for X is quite large.

A user can utilise the xhost command to control access to their X display. eg. ’xhost -’ bars access from all users, while ’xhost +harry’ gives X access to the user harry.

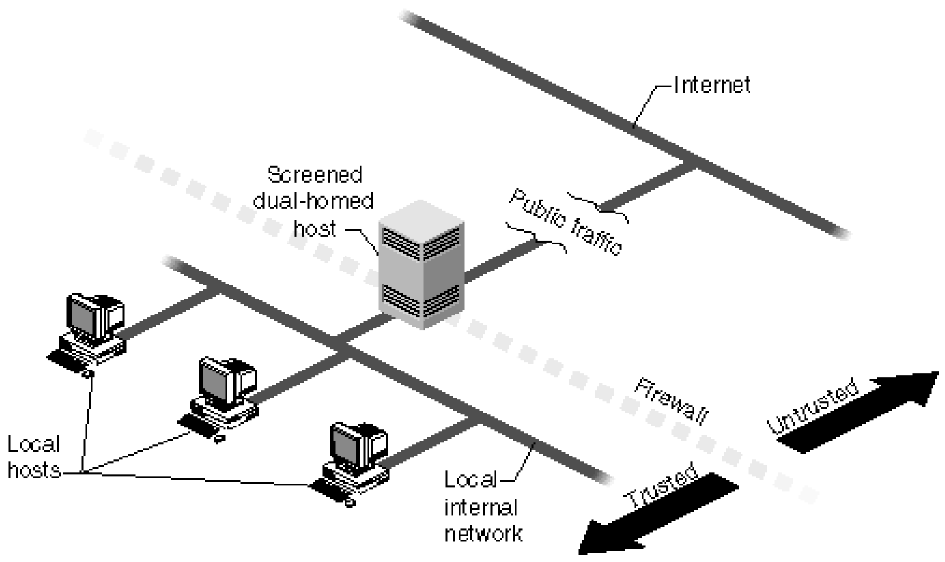
Note that system-level commands and files which relate to xhost and X in general are stored in /var/X11/xdm.

## Firewalls [4].

A firewall is a means by which a local network of trusted hosts can be connected to an external untrusted network, such as the Internet, in a more secure manner than would otherwise be the case. ’Firewall’ is a conceptual idea which refers to a combination of hardware and software steps taken to setup a desired level of security; although an admin can setup a firewall via basic steps with as-supplied tools, all modern systems have commercial packages available to aid in the task of setting up a firewall environment, eg. Gauntlet for IRIX systems.

As with other security measures, there is a tradeoff between ease of monitoring/administration, the degree of security required, and the wishes/needs of users. A drawback of firewalls is when a user has a legitimate need to access packets which are filtered out - an alternative is to have each host on the local network configured according to a strict security regime.

The simplest form of a firewall is a host with more than one network interface, called a dual-homed host [9]. Such hosts effectively exist on two networks at once. By configuring such a host in an appropriate manner, it acts as a controllable obstruction between the local and external network, eg. the Internet.



A firewall does not affect the communications between hosts on an internal network; only the way in which the internal network interacts with the external connection is affected. Also, the presence of a firewall should not be used as an excuse for having less restrictive security measures on the internal network.

One might at first think that Yoda could be described as a firewall, but it is not, for a variety of reasons. Ideally, a firewall host should be treated thus:

- no ordinary user accounts (root admin only, with a different password),
- as few services as possible (the more services are permitted, the greater is the chance of a security hole; newer, less-tested software is more likely to be at risk) and definitely no NIS or NFS,
- constantly monitored for access attempts and unusual changes in files, directories and software (commands: w, ps, ’versions changed’, etc.),
- log files regularly checked (and not stored on the firewall host!),
- no unnecessary applications,
- no anonymous ftp!

Yoda breaks several of these guidelines, so it cannot be regarded as a firewall, even though a range of significant security measures are in place. Ideally, an extra host should be used, eg. an Indy (additional Ethernet card required to provide the second Ethernet port), or a further server such as Challenge S. A simple system like Indy is sufficient though, or other UNIX system such as an HP, Sun, Dec, etc. - a Linux PC should not be used though since Linux has too many security holes in its present form. [1]

Services can be restricted by making changes to files such as /etc/inetd.conf, /etc/services, and others. Monitoring can be aided via the use of free security-related packages such as COPS - this package can also check for bad file permission settings, poorly chosen passwords, system setup file integrity, root security settings, and many other things. COPS can be downloaded from:

```
ftp://ftp.cert.org/pub/tools/cops
```

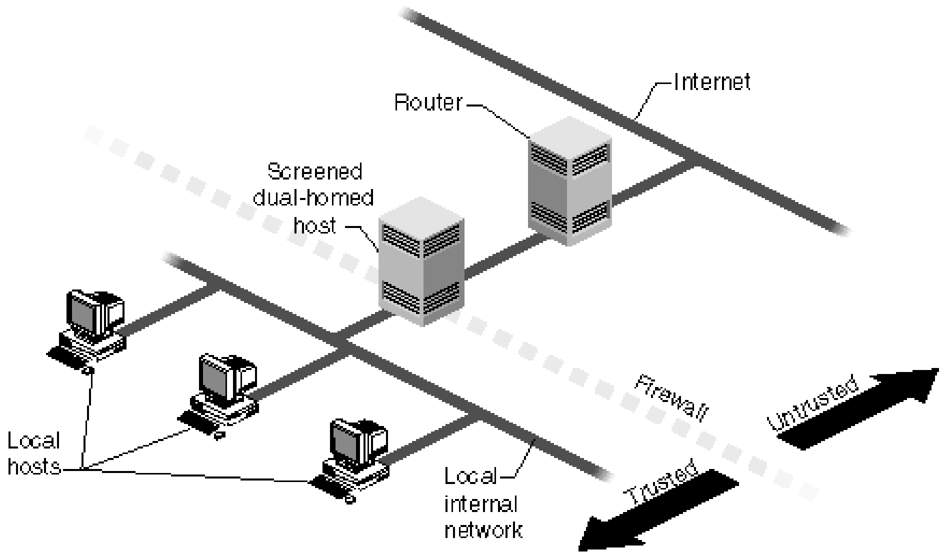
Monitoring a firewall host is also a prime candidate for using scripts to automate the monitoring process.

Other free tools include Tripwire, a file and directory integrity checker:

```
ftp://ftp.cert.org/pub/tools/tripwire
```

With Tripwire, files are monitored and compared to information stored in a database. If files change when they're supposed to remain static according to the database, the differences are logged and flagged for attention. If used regularly, eg. via cron, action can be taken immediately if something happens such as a hacking attempt.

Firewall environments often include a router - a high speed packet filtering machine installed either privately or by the ISP providing the external connection. Usually, a router is installed inbetween a dual-homed host and the outside world [9]. This is how yoda is connected, via a router whose address is 193.61.250.33, then through a second router at 193.61.250.65 before finally reaching the JANET gateway at Manchester.



Routers are not very flexible (eg. no support for application-level access restriction systems such as proxy servers), but their packet-filtering abilities do provide a degree of security, eg. the router at 193.61.250.33 only accepts packets on the 193.61.250.* address space.

However, because routers can block packet types, ports, etc. it is possible to be overly restrictive with their use, eg. yoda cannot receive USENET packets because they're blocked by the router. In such a scenario, users must resort to using WWW-based news services (eg. DejaNews) which are obviously less secure than running and managing a locally controlled USENET server, as well as being more wasteful of network resources.

Accessing sites on the web poses similar security problems to downloading and using Internet-sourced software, ie. the source is untrusted, unless vendor-verified with checksums, etc. When a user accesses a site and attempts to retrieves data, what happens next cannot be predicted, eg. a malicious executable program could be downloaded (this is unlikely to damage root-owned files, but users could lose data if they’re not careful). Users should be educated on these issues, eg. turning off Java script features and disallowing cookies if necessary.

If web access is of particular concern with regard to security, one solution is to restrict web access to just a limited number of internal hosts.

**Anonymous ftp.**

An anonymous FTP account allows a site to make information available to anyone, while still maintaining control over access issues. Users can login to an anonymous FTP account as ’anonymous’ or ’ftp’. The ’chroot’ command is used to put the user in the home directory for anonymous ftp access (~ftp), preventing access to other parts of the filesystem. A firewall host should definitely not have an anonymous FTP account. A site should not provide such a service unless absolutely necessary, but if it does then an understanding of how the anonymous FTP access system works is essential to ensuring site security, eg. preventing outside agents from using the site as a transfer point for pirated software. How an anon FTP account is used should be regularly monitored.

Details of how to setup an anon FTP account can usually be found in a vendor’s online information; for IRIX, the relevant source is the section entitled, "Setting Up an Anonymous FTP Account" in chapter three of the, "IRIX Admin: Networking and Mail" guide.

## UNIX Fundamentals: Internet access: files and services. Email.

For most users, the Internet means the World Wide Web (’http’ service), but this is just one service out of many, and was in fact a very late addition to the Internet as a whole. Before the advent of the web, Internet users were familiar with and used a wide range of services, including:

```
● telnet (interactive login sessions on remote hosts),
● ftp    (file/data transfer using continuous connections),
● tftp   (file/data transfer using temporary connections)
● NNTP   (Internet newsgroups, ie. USENET)
● SMTP   (email)
● gopher (remote host data searching and retrieval system)
● archie (another data-retrieval system)
● finger (probe remote site for user/account information)
● DNS    (Domain Name Service)
```

Exactly which services users can use is a decision best made by consultation, though some users may have a genuine need for particular services, eg. many public database systems on sites such as NASA are accessed by telnet only.

Disallowing a service automatically improves security, but the main drawback will always be a less flexible system from a user’s point of view, ie. a balance must be struck between the need for security and the needs of users. However, such discussions may be irrelevant if existing site policies already state what is permitted, eg. UCLAN’s campus network has no USENET service, so users exploit suitable external services such as DejaNews [2].

For the majority of admins, the most important Internet service which should be appropriately configured with respect to security is the web, especially considering today’s prevalence of Java, Java Script, and browser cookie files. It is all too easy for a modern web user to give out a surprising amount of information about the system they’re using without ever knowing it. Features such as cookies and Java allow a browser to send a substantial amount of information to a remote host about the user’s environment (machine type, OS, browser type and version, etc.); there are sites on the web which an admin can use to test how secure a user’s browser environment is - the site will display as much information as it can extract using all methods, so if such sites can only report very little or nothing in return, then that is a sign of good security with respect to user-side web issues.

There are many good web server software systems available, eg. Apache. Some even come free, or are designed for local Intranet use on each host. However, for enhanced security, a site should use a professional suite of web server software such as Netscape Enterprise Server; these packages come with more advanced control mechanisms and security management features, the configuration of which is controlled by GUI-based front-end servers, eg. Netscape Administration Server. Similarly, lightweight proxy servers are available, but a site should a professional solution, eg. Netscape Proxy Server. The GUI administration of web server software makes it much easier for an admin to configure security issues such as access and service restrictions, permitted data types, blocked sites, logging settings, etc.

Example: after the proxy server on the SGI network was installed, I noticed that users of the campus-wide PC network were using Yoda as a proxy server, which would give them a faster service than the University’s proxy server. A proxy server which is accessible in this way is said to be ’open’. Since all accesses from the campus PCs appear in the web logs as if they originate from the Novix security system (ie. there is no indication of individual workstation or user), any illegal activity would be untraceable. Thus, I decided to prevent campus PCs from using Yoda as a proxy. The mechanism employed to achieve this was the ipfilterd program, which I had heard of before but not used.

ipfilterd is a network packet-filtering daemon which screens all incoming IP packets based on source/destination IP address, physical network interface, IP protocol number, source/destination TCP/UDP port number, required service type (eg. ftp, telnet, etc.) or a combination of these. Up to 1000 filters can be used. To improve efficiency, a configurable memory caching mechanism is used to retain recently decided filter verdicts for a specified duration.

ipfilterd operates by using a searchable database of packet-filtering clauses stored in the /etc/ipfilterd.conf file. Each incoming packet is compared with the filters in the file one at a time until a match is found; if no match occurs, the packet is rejected by default. Since filtering is a line-by-line database search process, the order in which filters are listed is important, eg. a reject clause to exclude a particular source IP address from Ethernet port ec0 would have no effect if an accept clause was earlier in the file that accepted all IP data from ec0, ie. in this case, the reject should be listed before the accept. IP addresses may be specified in hex, dot format (eg. 193.61.255.4 - see the man page for ’inet’), host name or fully-qualified host name.

With IRIX 6.2, ipfilterd is not installed by default. After consulting with SGI to identify the appropriate source CD, the software was installed, /etc/ipfilterd.conf defined, and the system activated with:

```
    chkconfig -f ipfilterd on
    reboot
```

Since there was no ipfilterd on/off flag file in /etc/config by default, the -f forces the creation of such a file with the given state.

Filters in the /etc/ipfilterd.conf file consist of a keyword and an expression denoting the type of filter to be used; available keywords are:

- `accept       Accept all packets matching this filter`
- `reject       Discard all packets matching this filter (silently)`
- `grab         Grab all packets matching this filter`
- `define       Define a new macro`

ipfilterd supports macros, with no limit to the number of macros used.

Yoda’s /etc/ipfilterd.conf file looks like this:

```
#
# ipfilterd.conf
# $Revision: 1.3 $
#
# Configuration file for ipfilterd(1M) IP layer packet filtering.
# Lines that begin with # are comments and are ignored.
# Lines begin with a keyword, followed either by a macro definition or
# by an optional interface filter, which may be followed by a protocol filter.
# Both macros and filters use SGI’s netsnoop(1M) filter syntax.
#
# The currently supported keywords are:
# accept        : accept all packets matching this filter
# reject        : silently discard packets matching this filter
# define        : define a new macro to add to the standard netsnoop macros
#
# See the ipfilterd(1M) man page for examples of filters and macros.
#
# The network administrator may find the following macros useful:
#
define ip.netAsrc (src&0xff000000)=$1
define ip.netAdst (dst&0xff000000)=$1
define ip.netBsrc (src&0xffff0000)=$1
define ip.netBdst (dst&0xffff0000)=$1
define ip.netCsrc (src&0xffffff00)=$1
define ip.netCdst (dst&0xffffff00)=$1
define ip.notnetAsrc not((src&0xff000000)=$1)
define ip.notnetAdst not((dst&0xff000000)=$1)
define ip.notnetBsrc not((src&0xffff0000)=$1)
define ip.notnetBdst not((dst&0xffff0000)=$1)
define ip.notnetCsrc not((src&0xffffff00)=$1)
define ip.notnetCdst not((dst&0xffffff00)=$1)
#
# Additional macros:
#
# Filters follow:
#
accept -i ec0
reject -i ec3 ip.src 193.61.255.21 ip.dst 193.61.250.34
reject -i ec3 ip.src 193.61.255.22 ip.dst 193.61.250.34
accept -i ec3
```

Any packet coming from an SGI network machine is immediately accepted (traffic on the ec0 network interface). The web logs contained two different source IP addresses for accesses coming from the campus PC network. These are rejected first if detected; a final accept clause is then included so that all other types of packet are accepted.

The current contents of Yoda’s ipfilterd.conf file does mean that campus PC users will not be able to access Yoda as a web server either, ie. requests to www.comp.uclan.ac.uk by legitimate users will be blocked too. Thus, the above contents of the file are experimental. Further refinement is required so that accesses to Yoda’s web pages are accepted, while requests which try to use Yoda as a proxy to access non-UCLAN sites are rejected. This can be done by using the ipfilterd-expression equivalent of the following if/then C-style statement:

```
if ((source IP is campus PC) and (destination IP is not Yoda)) then
   reject packet;
```

Using ipfilterd has system resource implications. Filter verdicts stored in the ipfilterd cache by the kernel take up memory; if the cache size is increased, more memory is used. A longer cache and/or a larger number of filters means a greater processing overhead before each packet is dealt with. Thus, for busy networks, a faster processor may be required to handle the extra load, and perhaps more RAM if an admin increases the ipfilterd kernel cache size. In order to monitor such issues and make decisions about resource implications as a result of using ipfilterd, the daemon can be executed with the -d option which causes extra logging information about each filter to be added to /var/adm/SYSLOG, ie. an /etc/config/ipfilterd.options file should be created, containing ’-d’.

As well as using programs like ’top’ and ’ps’ to monitor CPU loading and memory usage, log files should be monitored to ensure they do not become too large, wasting disk space (the same applies to any kind of log file). System logs are ’rotated’ automatically to prevent this from happening, but other logs created by 3rd-party software usually are not; such log files are not normally stored in /var/adm either. For example, the proxy server logs are in this directory:

```
/var/netscape/suitespot/proxy-sysname-proxy/logs
```

If an admin wishes to retain the contents of older system logs such as /var/adm/oSYSLOG, then the log file could be copied to a safe location at regular intervals, eg. once per night (the old log file could then be emptied to save space).

A wise policy would be to create scripts which process the logs, summarising the data in a more intuitive form. General shell script methods and programs such as grep can be used for this.

The above is just one example of the typical type of problem and its consequences that admins come up against when managing a system:

- The first problem was how to give SGI network users Internet access, the solution to which was a proxy server. Unfortunately, this allowed campus-PC users to exploit Yoda as an open proxy, so ipfilterd was then employed to prevent such unauthorised use.

Thus, as stated in the introduction, managing system security is an ongoing, dynamic process.

Another example problem: in 1998, I noticed that some students were not using the SGIs (or not asking if they could) because they thought the machines were turned off, ie. the monitor power-saving feature would blank out the screen after some duration. I decided to alter the way the Ve24 Indys behaved so that monitor power-saving would be deactivated during the day, but would still happen overnight.

The solution I found was to modify the /var/X11/xdm/Xlogin file. This file contains a section controlling monitor power-saving using the xset command, which normally looks like this:

```
#if [ -x /usr/bin/X11/xset ] ; then
#    /usr/bin/X11/xset s 600 3600
#fi
```

If these lines are uncommented (the hash symbols removed), a system whose monitor supports power-saving will tell the monitor to power down after ten minutes of unuse, after the last user logs out. With the lines still commented out, modern SGI monitors use power-saving by default anyway.

I created two new files in /var/X11/xdm:

```
-rwxr-xr-x    1 root    sys    1358 Oct 28  1998 Xlogin.powersaveoff*
-rwxr-xr-x    1 root    sys    1361 Oct 28  1998 Xlogin.powersaveon*
```

They are identical except for the the section concerning power-saving. Xlogin.powersaveoff contains:

```
if [ -x /usr/bin/X11/xset ] ; then
    /usr/bin/X11/xset s 0 0
fi
```

while Xlogin.powersaveon contains:

```
#if [ -x /usr/bin/X11/xset ] ; then
#    /usr/bin/X11/xset s 0 0
#fi
```

The two ’0’ parameters supplied to xset in the Xlogin.powersaveoff file have a special effect (see the xset man page for full details): the monitor is instructed to disable *all* power-saving features.

The cron system is used to switch between the two files when no one is present: every night at 9pm and every morning at 8am, followed by a reboot after the copy operation is complete. The entries from the file /var/spool/cron/crontabs/cron on any of the Ve24 Indys are thus:

```
# Alternate monitor power-saving. Turn it on at 9pm. Turn it off at 8am.
0  21 *  *  *  /bin/cp /var/X11/xdm/Xlogin.powersaveon /var/X11/xdm/Xlogin && init 6&
#
0  8  *  *  *  /bin/cp /var/X11/xdm/Xlogin.powersaveoff /var/X11/xdm/Xlogin && init 6&
```

Hence, during the day, the SGI monitors are always on with the login logo/prompt visible - students can see the Indys are active and available for use; during the night, the monitors turn themselves off due to the new xset settings. The times at which the Xlogin changes are made were chosen so as to occur when other cron jobs would not be running. Students use the Indys each day without ever noticing the change, unless they happen to be around at the right time to see the peculiar sight of 18 Indys all rebooting at once.

## Static Routes.

A simple way to enable packets from clients to be forwarded through an external connection is via the use of a ’static route’. A file called /etc/init.d/network.local is created with a simple script that adds a routing definition to the current routing database, thus enabling packets to be forwarded to their destination. To ensure the script is executed on bootup or shutdown, extra links are added to the /etc/rc0.d and /etc/rc2.d directories (the following commands need only be executed once as root):

```
ln -s /etc/init.d/network.local /etc/rc0.d/K39network
ln -s /etc/init.d/network.local /etc/rc2.d/S31network
```

Yoda once had a modem link to ’Demon Internet’ for Internet access. A static route was used to allow SGI network clients to access the Internet via the link. The contents of /etc/init.d/network.local (supplied by SGI) was:

```
#!/sbin/sh
#Tag 0x00000f00
IS_ON=/sbin/chkconfig
case "$1" in
  ’start’)
        if $IS_ON network; then
                /usr/etc/route add default 193.61.252.1 1
        fi ;;

  ’stop’)
        /usr/etc/route delete default 193.61.252.1 ;;

  *)
        echo "usage: $0 {start|stop}"
        ;;
esac
```

Note the use of chkconfig to ensure that a static route is only installed on bootup if the network is defined as active.

The other main files for controlling Internet access are /etc/services and /etc/inetd.conf. These were discussed earlier.

**Internet Access Policy.**

Those sites which choose to allow Internet access will probably want to minimise the degree to which someone outside the site can access internal services. For example, users may be able to telnet to remote hosts from a company workstation, but should the user be able to successfully telnet to that workstation from home in order to continue working? Such an ability would obviously be very useful to users, and indeed administrators, but there are security implications which may be prohibitive.

For example, students who have accounts on the SGI network cannot login to Yoda because the /etc/passwd file contains /dev/null as their default shell, ie. they can’t login because their account ’presence’ on Yoda itself does not have a valid shell - another cunning use of /dev/null. The /etc/passwd.nis file has the main user account database, so users can logon to the machines in Ve24 as desired. Thus, with the use of /dev/null in the password file’s shell field, students cannot login to Yoda via telnet from outside UCLAN. Staff accounts on the SGI network do not have /dev/null in the shell field, so staff can indeed login to Yoda via telnet from a remote host.

Ideally, I’d like students to be able to telnet to a Ve24 machine from a remote host, but this is not yet possible for reasons explained in Appendix A (detailed notes for Day 2 Part 1).

There are a number of Internet sites which are useful sources of information on Internet issues, some relating to specific areas such as newsgroups. In fact, USENET is an excellent source of information and advice on dealing with system management, partly because of preprepared FAQ files, but also because of the many experts who read and post to the newsgroups. Even if site policy means users can’t access USENET, an admin should exploit the service to obtain relevant admin information.

A list of some useful reference sites are given in Appendix C.

**Example Questions:**

1. The positions of the ’accept ec0’ and ’reject’ lines in /etc/ipfilterd.conf could be swapped around without affecting the filtering logic. So why is the ec0 line listed first? The ’netstat -i’ command (executed on Yoda) may be useful here.

2. What would an appropriate ipfilterd.conf filter (or filters) look like which blocked unauthorised use of Yoda as a proxy to connect to an external site but still allowed access to Yoda’s own web pages via www.comp.uclan.ac.uk? Hint: the netsnoop command may be useful.

**Course summary.**

This course has focused on what an admin needs to know in order to run a UNIX system. SGI systems running IRIX 6.2 have been used as an example UNIX platform, with occasional mention of IRIX 6.5 as an example of how OSs evolve.

Admins are, of course, ordinary users too, though they often do not use the same set of applications that other users do. Though an admin needs to know things an ordinary user does not, occasionally users should be made aware of certain issues, eg. web browser cookie files, choosing appropriate passwords etc.

Like any modern OS, UNIX has a vast range of features and services. This course has not by any means covered them all (that would be impossible to do in just three days, or even thirty). Instead, the basic things a typical admin needs to know have been introduced, especially the techniques used to *find* information when needed, and how to exploit the useful features of UNIX for daily administration.

Whatever flavour of UNIX an admin has to manage, a great many issues are always the same, eg. security, Internet concepts, etc. Thus, an admin should consider purchasing relevant reference books to aid in the learning process. When writing shell scripts, knowledge of the C programming language is useful; since UNIX is the OS being used, a C programming book (mentioned earlier) which any admin will find particularly useful is:

```
"C Programming in a UNIX Environment"

Judy Kay & Bob Kummerfeld, Addison Wesley Publishing, 1989.
ISBN: 0 201 12912 4
```

For further information on UNIX or related issues, read/post to relevant newsgroups using DejaNews; example newsgroups are given in Appendix D.

---

**Background Notes:**

1. UNIX OSs like IRIX can be purchased in a form that passes the US Department of Defence’s Trusted-B1 security regulations (eg. ’Trusted IRIX’), whereas Linux doesn’t come anywhere near such rigorous security standards as yet. The only UNIX OS (and in fact the only OS of any kind) which passes *all* of the US DoD’s toughest security regulations is Unicos, made by Cray Research (a subsidiary of SGI). Unicos and IRIX will be merged sometime in the future, creating the first widely available commercial UNIX OS that is

extremely secure - essential for fields such as banking, local and national government, military, police (and other emergency/crime services), health, research, telecoms, etc.

## References:

2. DejaNews USENET Newsgroups, Reading/Posting service:

```
http://www.dejanews.com/
```

4. "Firewalls: Where there's smoke...", Network Week, Vol4, No. 12, 2nd December 1998, pp. 33 to 37.

5. Gauntlet 3.2 for IRIX Internet Firewall Software:

```
http://www.sgi.com/solutions/internet/products/gauntlet/
```

6. Framebuffer and Clipping Planes, Indigo2 Technical Report, SGI, 1994:

```
http://www.futuretech.vuurwerk.nl/i2sec4.html#4.3
http://www.futuretech.vuurwerk.nl/i2sec5.html#5.6.3
```

7. Useful security-related web sites:

```
UKERNA:     http://www.ukerna.ac.uk/
JANET:      http://www.ja.net/
CERT:       http://www.cert.org/
RootShell:  http://www.rootshell.com/
2600:       http://www.2600.com/mindex.html
```

8. "About the X Window System", part of X11.org:

```
http://www.X11.org/wm/index.shtml
```

9. Images are from the online book, "IRIX Admin: Backup, Security, and Accounting.", Chapter 5.

## Appendix B:

3. Contents of /etc/fingerd.message:

```
Sorry, the finger service is not available from this host.

However, thankyou for your interest in the Department of
Computing at the University of Central Lancashire.

For more information, please see:

  http://www.uclan.ac.uk/
  http://www.uclan.ac.uk/facs/destech/compute/comphom.htm

Or contact Ian Mapleson at mapleson@gamers.org

Regards,
Ian.

Senior Technician,
Department of Computing,
University of Central Lancashire,
Preston,
England,
PR1 2HE.

mapleson@gamers.org
Tel: (+44 -0) 1772 893297
Fax: (+44 -0) 1772 892913

Doom Help Service (DHS):   http://doomgate.gamers.org/dhs/
SGI/Future Technology/N64: http://sgi.webguide.nl/
BSc Dissertation (Doom):   http://doomgate.gamers.org/dhs/diss/
```

## Appendix C:

Example web sites useful to administrators:

| | |
|---|---|
| **AltaVista:** | **http://altavista.digital.com/cgi-bin/query?pg=aq** |
| **Webcrawler:** | **http://webcrawler.com/** |
| **Lycos:** | **http://www.lycos.com/** |
| **Yahoo:** | **http://www.yahoo.com/** |
| **DejaNews:** | **http://www.dejanews.com/** |
| **SGI Support:** | **http://www.sgi.com/support/** |
| **SGI Tech/Advice Center:** | **http://www.futuretech.vuurwerk.nl/sgi.html** |
| **X Windows:** | **http://www.x11.org/** |
| **Linux Home Page:** | **http://www.linux.org/** |
| **UNIXHelp for Users:** | **http://unixhelp.ed.ac.uk/** |
| **Hacker Security Update:** | **http://www.securityupdate.com/** |
| **UnixVsNT:** | **http://www.unix-vs-nt.org/** |
| **RootShell:** | **http://www.rootshell.com/** |
| **UNIX System Admin (SunOS):** | **http://sunos-wks.acs.ohio-state.edu/sysadm_course/html/sysadm-** |

## Appendix D:

Example newsgroups useful to administrators:

```
comp.security.unix
comp.unix.admin
comp.sys.sgi.admin
comp.unix.admin
comp.sys.sun.admin
comp.sys.next.sysadmin
comp.unix.aix
comp.unix.cray
comp.unix.misc
comp.unix.questions
comp.unix.shell
comp.unix.solaris
comp.unix.ultrix
comp.unix.wizards
comp.unix.xenix.misc
comp.sources.unix
comp.unix.bsd.misc
comp.unix.sco.misc
comp.unix.unixware.misc
comp.sys.hp.hpux
comp.unix.sys5.misc
comp.infosystems.www.misc
comp.sys.dec
```