Linux Hardening

A guide to secure your Linux Operating System

**Removing Unnecessary Software Packages (RPMs)**

A very important step in securing a Linux system is to determine the primary function or role of the Linux server. You should have a detailed knowledge of what is on your system. Otherwise you will have a difficult time to understand what needs to be secured and hence securing your Linux systems proactively won't be that effective. Therefore, it is very critical to look at the default list of software packages and remove unneeded packages or packages that don't comply with your security policy. If you do that you will have less packages to update and to maintain when security alerts and patches are released. For example, you should not have Apache or Samba installed on your system if you don't use them. Also, it is a good practice not to have development packages, desktop software packages (e.g. X Server) etc. installed on production servers. Other packages like FTP and Telnet daemons should not be installed as well unless there is a justified business reason for it (SSH/SCP/SFTP should be used instead).  
  
*One of the first action items should be to create a Linux image that only contains RPMs needed by the applications, and needed for maintenance and troubleshooting purposes. A good approach is to start with a minimum list of RPMs and then add packages as needed. It may be time-consuming but worth the efforts.*   
  
To get a list of all installed RPMs you can use the following command:

rpm -qa

If you want to know more about a particular RPM, run:

rpm -qi <package\_name>

To check for and report potential conflicts and dependencies for deleting a RPM, run:

rpm -e **--test** <package\_name>

**Patching Linux Systems**

Building an infrastructure for patch management is another very important step to proactively secure Linux production environments. It is recommended to have a written security policy and procedure to handle Linux security updates and issues.   
  
For example, a security policy should detail the timeframe for assessment, testing, and rollout of patches. Network related security vulnerabilities should get the highest priority and should be addressed immediately within a short timeframe.   
  
For example, a security procedure should detail the process for assesment, testing, and rollout of patches. The assessment phase should occur within a testing lab, and initial rollout should occur on development systems first.   
  
A separate security log should detail what Linux security notices have been received, when patches have been researched and assessed, when patches have been applied etc.   
  
For Red Hat systems I recommend [Red Hat Network (RHN)](http://www.redhat.com/software/rhn/) for patch management. In fact, for secure environments you may have to consider Red Hat's Satellite solution. For more information, see [Red Hat Network Architectural Overview](http://www.redhat.com/software/rhn/architecture/).

**Detecting Listening Network Ports**

One of the most important tasks is to detect and close network ports that are not needed.   
  
To get a list of listening network ports (TCP and UDP sockets), you can run the following command:

# netstat -tulp

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 \*:auth \*:\* LISTEN 2328/xinetd

tcp 0 0 localhost.localdomain:smtp \*:\* LISTEN 2360/sendmail: acce

tcp 0 0 \*:ssh \*:\* LISTEN 2317/sshd

From the output you can see that xinetd, sendmail, and sshd are listening.   
  
On all newer Red Hat Linux distributions sendmail is configured to listen for local connections only. Sendmail should not listen for incoming network connections unless the server is a mail or relay server. Running a port scan from another server will confirm that (make sure that you have permissions to probe a machine):

# nmap -sTU <remote\_host>

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-12-10 22:51 CST

Interesting ports on jupitor (172.16.0.1):

(The 3131 ports scanned but not shown below are in state: closed)

PORT STATE SERVICE

22/tcp open ssh

113/tcp open auth

Nmap run completed -- 1 IP address (1 host up) scanned in 221.669 seconds

#

Note that the above nmap command can take a while. If you remove the UDP port scan (without the option "-U"), then nmap will finish the port scan immediately. If you run it on the local machine it will also complete very fast. Also note that nmap might not show all listening network sockets if a firewall is being used to block ports.   
From the output above you can see that the xinetd daemon is listening on port auth (port 113) for IDENT (for more information on this service, see below). You can also see that sendmail is not listening for remote incoming network connections, see also [Securing Sendmail](http://www.puschitz.com/SecuringLinux.shtml#SecuringSendmail).   
  
Another method to list all of the TCP and UDP sockets to which programs are listening is lsof:

# lsof -i -n | egrep 'COMMAND|LISTEN|UDP'

COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME

sshd 2317 root 3u IPv6 6579 TCP \*:ssh (LISTEN)

xinetd 2328 root 5u IPv4 6698 TCP \*:auth (LISTEN)

sendmail 2360 root 3u IPv4 6729 TCP 127.0.0.1:smtp (LISTEN)

#

**Closing Network Ports and Disabling Runlevel System Services**

One of the most important tasks is to remove any network services from the system startup process that are not needed.   
  
On Red Hat systems you can list all services which are started at bootup using the following command:

chkconfig --list |grep on

You will notice that there are quite a lot of services enabled on your system. But many runlevel services (Stand-Alone Services) are not network related services like kudzu which is responsible for detecting and configuring new and/or changed hardware on your system. This service is only run during the boot process. Ensure not to disable runlevel services that are needed by the system to run smoothly.   
  
Here are examples of Red Hat Runlevel System Services which you may or may not want to enable:

gpm needed if you want to use the mouse at the console

kudzu important for detecting new hardware

syslog important for syslog services

netfs needed only if there are NFS shares that should be mounted at boot time

network important for starting network interfaces (e.g. eth0, eth1, bonding,...)

random used for the system entropy pool

atd needed if the at(1) service is used instead of cron

apmd Advanced Power Management (APM) daemon is used for laptops and some desktops

isdn needed if ISDN is being used

iptables needed if Netfilter (iptables) Firewall is being used

ip6tables needed if ip6tables Firewall is being used

pcmcia not needed on servers - needed for laptops

irqbalance important for distributing interrupts across all CPUs

sendmail needed if Sendmail is used - Procmail should be used which is more secure

autofs needed if automounter is used - production applications should not be dependent on automounter

sshd important for logins via SSH

portmap needed if e.g. NFS is being used

nfslock needed if NFS shares are mounted

nfs needed if server runs the NFS server

mdmonitor needed only if software RAID is being used

crond important for running cron jobs

xinetd needed if xinetd services are being used, see /etc/xinetd.d/ for list of services

cups needed if CUPS is used for the printing system

rhnsd needed if server should connect to RHN to check for software updates etc.

sysstat needed to reset system statistics logs

audit needed only if Linux Audit Subsystem (LAuS) should run for collecting system call audit records

psacct needed only if kernel process accounting information is needed

smartd important for monitoring disk problems if hard disks support SMART technology

netdump important if kernel oops data and memory dumps should be sent to a Netdump server for server crashes

The start/stop scripts of all runlevel services can be found in the /etc/init.d directory. For example, if you don't know what the atd service does, go to /etc/init.d and open the file atd. And in the script look for lines that start programs. In the atd script the "daemon /usr/sbin/atd" line starts the binary atd. Now having the name of the program that is started by this service, you can check the online pages of atd by running man atd. This will help you to find out more about a system service.   
  
To permanently disable e.g. the runlevel service nfs, run:

chkconfig nfs off

To immediately disable the runlevel service nfs, run:

/etc/init.d/nfs stop

**Closing Network Ports and Disabling Xinetd Services**

The xinetd daemon is a replacement for inetd, the internet services daemon. It monitors the ports for all network services configured in /etc/xinetd.d, and starts the services in response to incoming connections.   
  
To check if xinetd is enabled and running, execute:

# chkconfig --list xinetd

xinetd 0:off 1:off 2:off 3:**on** 4:**on** 5:**on** 6:off

# /etc/init.d/xinetd status

xinetd (pid 2619) is running...

#

If xinetd is active, it is important to check which Unix services are active and controlled by xinetd. The following command shows all services configured in /etc/xinetd.d and wheter xinetd monitors the ports for these services:

# chkconfig --list | awk '/xinetd based services/,/""/'

xinetd based services:

krb5-telnet: off

rsync: off

eklogin: off

gssftp: off

klogin: off

chargen-udp: off

kshell: off

auth: on

chargen: off

daytime-udp: off

daytime: off

echo-udp: off

echo: off

services: off

time: off

time-udp: off

cups-lpd: off

#

To get a list of only active services for which xinetd monitors the ports, you could run:

# chkconfig --list | awk '/xinetd based services/,/""/' | grep -v off

xinetd based services:

auth: on

#

In the above example you can see that the telnet-server RPM is not installed on the system. If the Telnet Server package telnet-server would be installed, it would show up on the list whether it's active or not.   
  
Here is an example how to disable a service. Assuming the telnet service is active, run the following commands to disable it and to see how the telnet service entries are being updated:

# chkconfig --list telnet

telnet on

# cat /etc/xinetd.d/telnet | grep disable

disable = no

# chkconfig telnet off

# chkconfig --list telnet

telnet off

# cat /etc/xinetd.d/telnet | grep disable

disable = yes

#

For the telnet service it would be better to remove the package from the system since SSH should be used instead:

# rpm -e telnet-server

It is important to investigate all active xinetd services and to disable them if they are not needed.   
  
Here is an example how to find out what a service does. Assuming you don't know what the auth service does which is listed as active in the list above, run the following commands:

# grep " server" /etc/xinetd.d/auth

server = /usr/sbin/in.authd

server\_args = -t60 --xerror --os -E

# man in.auth

No manual entry for in.auth

# rpm -qf /usr/sbin/in.authd

authd-1.4.1-1.rhel3

# rpm -qi authd-1.4.1-1.rhel3 | awk '/Description/,/""/'

Description :

authd is a small and fast RFC 1413 ident protocol daemon

with both xinetd server and interactive modes that

supports IPv6 and IPv4 as well as the more popular features

of pidentd.

# rpm -ql authd-1.4.1-1.rhel3

/etc/ident.key

/etc/xinetd.d/auth

/usr/sbin/in.authd

/usr/share/doc/authd-1.4.1

/usr/share/doc/authd-1.4.1/COPYING

/usr/share/doc/authd-1.4.1/README.html

/usr/share/doc/authd-1.4.1/rfc1413.txt

/usr/share/locale/ja/LC\_MESSAGES/authd.mo

#

This example shows what can be done if there exists no online manuals for the binary in.authd that is started by xinetd. The steps above should be helpful for finding out more about services.   
  
The  auth service (aka IDENT, see RFC 1413) allows remote daemons to query information about users establishing TCP connections on the local server. In a trusted environment it helps a server to identify who is trying to use it. For example, it can provide vital information for troubleshooting and who has done what. IDENT requests are needed by some applications like IRC. However, IDENT can be a security risk.   
  
To disable the auth service, run the following command:

# chkconfig auth off

The xinetd daemon is quite flexible and has many features. Here are just a few functionalities of Xinetd:  
- Acces control for TCP, UDP, and RPC services  
- Acess limitations based on time  
- Provides mechanisms to prevent DoS attacks

**Reviewing Inittab and Boot Scripts**

The inittab file /etc/inittab also describes which processes are started at bootup and during normal operation. For example, Oracle uses it to start cluster services at bootup. Therefore, it is recommended to ensure that all entries in /etc/inittab are legitimate in your environment.   
  
I would at least remove the CTRL-ALT-DELETE trap entry to prevent accidental reboots:

# sed -i 's/ca::ctrlaltdel:/#ca::ctrlaltdel:/g'

The default runlevel should be set to 3 since in my opinion X11 (X Windows System) should not be running on a production server. In fact, it shouldn't even be installed.

# grep ':initdefault' /etc/inittab

id:**3**:initdefault:

#

And depending on your environment you might want to comment out the UPS entries as well.   
  
To have changes in /etc/inittab become effective immediately, you can run:

# init q

The /etc/rc.local script is used for commands or startup scripts which are pertinent only to a specific server. (/etc/rc.local is a link to /etc/rc.d/rc.local).   
Ensure that all startup scripts in /etc/rc.d/rc.local are legitimate.

**Restricting System Access from Servers and Networks**

Usually a firewall is used to protect a server from other servers and networks. However, in some cases you may also want to protect a server within a network by using a TCP Wrapper.   
  
The Xinetd super server that comes with most Linux distributions includes a built-in TCP wrapper. It can be used to explicitly define network services to accept incoming connections from specified servers and networks. The TCP wrappers implements access control through the use of two files, /etc/hosts.allowand /etc/hosts.deny. Note that the hosts.allow file takes precedence over the hosts.deny file. And you may want to change the permissions on the two configuration files since they are both world readable.   
  
A recommended security-strategy is to block all incoming requests by default, but allow specific hosts or networks to connect. This is the strategy I will describe here.   
  
To deny everything by default, add the following line to /etc/hosts.deny:

ALL: ALL

To accept incoming SSH connections from e.g. nodes rac1cluster, rac2cluster and rac3cluster, add the following line to /etc/hosts.allow:

sshd: rac1cluster rac2cluster rac3cluster

To accept incoming SSH connections from all servers from a specific network, add the name of the subnet to /etc/hosts.allow. For example:

sshd: rac1cluster rac2cluster rac3cluster .subnet.example.com

To accept incoming portmap connections from IP address 192.168.0.1 and subnet 192.168.5, add the following line to /etc/hosts.allow:

portmap: 192.168.0.1 192.168.5.

To accept connections from all servers on subnet .subnet.example.com but not from server cracker.subnet.example.com, you could add the following line to /etc/hosts.allow:

ALL: .subnet.example.com EXCEPT cracker.subnet.example.com

Here are other examples that show some features of TCP wrapper:   
  
If you just want to restrict ssh connections without configuring or using /etc/hosts.deny, you can add the following entries to /etc/hosts.allow:

sshd: rac1cluster rac2cluster rac3cluster

sshd: ALL: DENY

The version of TCP wrapper that comes with Red Hat also supports the extended options documented in the hosts\_options(5) man page. Here is an example how an additional program can be spawned in e.g. the /etc/hosts.allow file:

sshd: ALL : spawn echo "Login from %c to %s" | mail -s "Login Info for %s" log@loghost

For information on the % expansions, see "man 5 hosts\_access".   
  
The TCP wrapper is quite flexible. And xinetd provides its own set of host-based and time-based access control functions. You can even tell xinetd to limit the rate of incoming connections. I recommend reading various documentations about the Xinetd super daemon on the Internet.

**Securing SSH**

Many network services like telnet, rlogin, and rsh are vulnerable to eavesdropping which is one of several reasons why SSH should be used instead. Red Hat's default configuration for SSH meets the security requirements for many environments. However, there are a few parameters in /etc/ssh/sshd\_config that you may want to change on RHEL and other Linux systems.   
  
The chapter [Restricting System Access from Servers and Networks](http://www.puschitz.com/SecuringLinux.shtml#RestrictingSystemAccessFromServersAndNetworks) shows how direct logins can be disabled for shared and system accounts including root. But it's prudent to disable direct root logins at the SSH level as well.

PermitRootLogin no

Also ensure to have privilege separation enabled where the daemon is split into two parts. With privilege separation a small part of the code runs as root and the rest of the code runs in a chroot jail environment. Note that on older RHEL systems this feature can break some functionality, for example see [Preventing Accidental Denial of Service](http://www.puschitz.com/SecuringLinux.shtml#PreventingAccidentalDenialOfService).

UsePrivilegeSeparation yes

Since SSH protocol version 1 is not as secure you may want to limit the protocol to version 2 only:

Protocol 2

You may also want to prevent SSH from setting up TCP port and X11 forwarding if you don't need it:

AllowTcpForwarding no

X11Forwarding no

Ensure the StrictModes directive is enabled which checks file permissions and ownerships of some important files in the user's home directory like ~/.ssh, ~/.ssh/authorized\_keys etc. If any checks fail, the user won't be able to login.

StrictModes yes

Ensure that all host-based authentications are disabled. These methods should be avoided as primary authentication.

IgnoreRhosts yes

HostbasedAuthentication no

RhostsRSAAuthentication no

Disable sftp if it's not needed:

#Subsystem sftp /usr/lib/misc/sftp-server

After changing any directives make sure to restart the sshd daemon:

/etc/init.d/sshd restart

**Securing Postfix**

Postfix is a replacement for Sendmail which has several security advantages over Sendmail. Postfix consists of several small programs that perform their own small task. And almost all programs run in a chroot jail. These are just a few examples why Postfix is recommended over Sendmail. For more information on chroot jail, see [Using Chroot Securely](http://www.linuxsecurity.com/content/view/117632/49/).   
  
Linux servers that are not dedicated mail or relay servers should not accept external emails. However, it is important for production servers to send local emails to a relay server.   
  
Before you continue on a Red Hat system, make sure Postfix is activated using the following command:

# alternatives --set mta /usr/sbin/sendmail.postfix

The following parameters in /etc/postfix/main.cf should be set to ensure that Postfix accepts only local emails for delivery:

mydestination = $myhostname, localhost.$mydomain, localhost

inet\_interfaces = localhost

The parameter mydestination lists all domains to receive emails for.  
The parameter inet\_interfaces specifies the network to liston on.   
  
Once you've configured Postfix, restart the mail system with the following command:

# /etc/init.d/postfix restart

To verify whether Postfix is still listening for incoming network request, you can run one of the following commands from another node:

# nmap -sT -p 25 <remode\_node>

# telnet <remote\_node> 25

Don't run these commands on the local host since Postfix is supposed to accept connections from the local node.   
  
If you believe that I should cover other parameter(s) or if you think that other parameter(s) should explicitly be set/changed for local mail delivery, please drop me an email.

**Securing Sendmail**

This article focuses on security issues that pertain to most Linux servers in a production environment. Therefore, securing a mail or relay server is out of scope for this article since not all Linux servers in a production environment are mail or relay servers. However, Sendmail or Postfix is usually required for local mail delivery. Note that it is recommended to use Postfix over Sendmail for various security reasons, see [Securing Postfix](http://www.puschitz.com/SecuringLinux.shtml#SecuringPostfix) for more information.   
  
On newer Linux systems Sendmail is configured to run in the background for local mail delivery and not to accept incoming network connections. If your server is not a mail or relay server, then it is important that Sendmail is not accepting incoming network connections from any host other than the local server.   
  
The default sendmail.cf configuration file on RedHat does not allow Sendmail to accept incoming network connections. The following setting in /etc/mail/sendmail.cf tells Sendmail not to accept incoming network connections from servers other than the local node:

O DaemonPortOptions=Port=smtp,Addr=127.0.0.1, Name=MTA

If that's not the case on your system, you can change it by setting or uncommenting the DAEMON\_OPTIONSparameter in the /etc/mail/sendmail.mc file. Uncomment the DAEMON\_OPTIONS line in/etc/mail/sendmail.mc to read:

DAEMON\_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl

Then run:

# mv /etc/mail/sendmail.cf /etc/mail/sendmail.cf.old

# m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf

# /etc/init.d/sendmail restart

To verify whether Sendmail is still listening for incoming network request, you can run one of the following commands from another node (make sure that you have permissions to probe a machine):

# nmap -sT -p 25 <remode\_node>

# telnet <remote\_node> 25

Don't run these commands on the local host since Sendmail is supposed to accept connections from the local node.

**Securing NFS**

**General**  
  
NFS (Network File System) allows servers to share files over a network. But like all network services using NFS involves risks.   
  
Here are some basic rules:  
- NFS should not be enabled if not needed.  
- If you must use NFS, use TCP wrapper to restrict remote access.  
- Make sure you export to only those machines that you really need to.  
- Use fully qualified domain names to diminish spoofing attempts.  
- Export only directories you need to export.  
- Export read-only wherever possible.  
- Use NFS over TCP.  
  
If you don't have shared directories to export, ensure that the NFS service is NOT enabled and running:

# service nfs status

rpc.mountd is stopped

nfsd is stopped

rpc.rquotad is stopped

# chkconfig --list nfs

nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off

#

You probably don't need the portmap service as well which is used by NFS (the portmap daemon registers rpc-based services for services like NFS, NIS, etc.):

# service portmap status

portmap is stopped

# chkconfig --list portmap

portmap 0:off 1:off 2:off 3:off 4:off 5:off 6:off

#

**Enabling and Starting NFS Server**  
  
If you must use NFS, it can be activated using the following commands on Red Hat Linux:

chkconfig portmap on

chkconfig nfs on

service portmap start

service nfs start

The "portmap" service starts the portmap daemon.   
Depending on the Linux OS distribution and version, the "nfs" service starts the rpc.rquotad, nfsd, lockd, rpciod, rpc.mountd and rpc.idmapd daemons on Red Hat Fedora Core 3 which is the OS that Red Hat Advanced Server 4 is based on. On Red Hat Advanced Server 3, the NFS service startsrpc.mountd, nfsd and rpc.rquotad.   
  
To probe the portmapper for all registered NFS related RPC programs, you can run rpcinfo. On Red Hat Advanced Server 3, the output will look like this:

# rpcinfo -p <server>

program vers proto port

100000 2 tcp 111 portmapper

100000 2 udp 111 portmapper

100011 1 udp 607 rquotad

100011 2 udp 607 rquotad

100011 1 tcp 610 rquotad

100011 2 tcp 610 rquotad

100003 2 udp 2049 nfs

100003 3 udp 2049 nfs

100003 2 tcp 2049 nfs

100003 3 tcp 2049 nfs

100005 1 udp 623 mountd

100005 1 tcp 626 mountd

100005 2 udp 623 mountd

100005 2 tcp 626 mountd

100005 3 udp 623 mountd

100005 3 tcp 626 mountd

#

**Restricting Incoming NFS Requests**  
  
As I showed at [Restricting System Access from Servers and Networks](http://www.puschitz.com/SecuringLinux.shtml#RestrictingSystemAccessFromServersAndNetworks) a recommended security-strategy is to block all incoming requests by default, but allow specific hosts or networks to connect. This is the strategy I will use here.   
  
The portmap program and some of the NFS programs include a built-in TCP wrapper. To verify if a program includes a TCP wrapper, you can run the following commands:

# strings /sbin/portmap | egrep "hosts.deny|hosts.allow|libwrap"

hosts\_allow\_table

hosts\_deny\_table

**/etc/hosts.allow**

**/etc/hosts.deny**

# strings /usr/sbin/rpc.rquotad | egrep "hosts.deny|hosts.allow|libwrap"

**libwrap.so.0**

# ldd /usr/sbin/rpc.rquotad | grep libwrap

**libwrap.so.0** => /usr/lib/libwrap.so.0 (0x00874000)

#

If hosts.deny and hosts.allow are displayed, or if libwrap is displayed, then the program includes a built-in TCP wrapper. If none of these strings are displayed, then adding the program name to /etc/hosts.deny and /etc/hosts.allow will most probably have no effect.   
  
To block all incoming requests by default, add the following line to /etc/hosts.deny if you have not done so yet:

ALL: ALL

Verify from a remote server that portmapper does not list any registered RPC programs:

# rpcinfo -p <server>

No remote programs registered.

#

To allow NFS mounts from trusted servers and networks only, you have to configure the servers and networks for portmap and various NFS related programs in /etc/hosts.allow. To find out which NFS programs are invoked when you start the NFS server, you can check the /etc/init.d/nfs service script. Then run the "strings <program> | egrep "hosts.deny|hosts.allow|libwrap"" command I described above.   
  
On Red Hat Advanced Server 3, the /usr/sbin/rpc.rquotad program includes a built-in TCP wrapper. On Red Hat Fedora Core 3, /usr/sbin/rpc.mountd now also includes a built-in TCP wrapper.   
  
To allow NFS requests from e.g. servers rac1pub.example.com, rac2pub.example.com, rac3pub.example.com and from the .subnet.puschitz.com network, the configuration in /etc/hosts.allow would look like as follows:

portmap: rac1pub.example.com rac2pub.example.com rac3pub.example.com .subnet.puschitz.com

rpc.mountd: rac1pub.example.com rac2pub.example.com rac3pub.example.com .subnet.puschitz.com

rpc.rquotad: rac1pub.example.com rac2pub.example.com rac3pub.example.com .subnet.puschitz.com

Make sure to test NFS access thoroughly when using TCP wrappers. If a TCP wrapper has been added to another NFS related program, please drop me a note.   
  
For portmapper you can now test access from trusted servers or networks using the rpcinfo command:

# rpcinfo -p <server>

program vers proto port

100000 2 tcp 111 portmapper

100000 2 udp 111 portmapper

100011 1 udp 607 rquotad

100011 2 udp 607 rquotad

100011 1 tcp 610 rquotad

100011 2 tcp 610 rquotad

100003 2 udp 2049 nfs

100003 3 udp 2049 nfs

100003 2 tcp 2049 nfs

100003 3 tcp 2049 nfs

100005 1 udp 623 mountd

100005 1 tcp 626 mountd

100005 2 udp 623 mountd

100005 2 tcp 626 mountd

100005 3 udp 623 mountd

100005 3 tcp 626 mountd

#

If you run it from an "untrusted" server or network, you should get the following output:

# rpcinfo -p <server>

No remote programs registered.

#

**Exporting NFS File Systems**  
  
To allow a client access to a filesystem or directory, the /etc/exports serves as the access control list.   
  
To give the network "subnet.example.com" read-only access to /pub, the entries in /etc/exports would look like as follows:

/pub \*.subnet.example.com(**ro**,sync)

It is very important NOT to give write access to NFS clients if not absolutely needed! Entries in /etc/exports are exported read-only ("ro" option) by default.   
  
To allow servers rac1pub, rac2pub and rac3pub read-write access to the /data/OracleArch directory, the entries in /etc/exports would look like as follows:

/data/OracleArch rac1pub.example.com(rw,sync) rac2pub.example.com(rw,sync) rac3pub.example.com(rw,sync)

Note that options MUST NOT be separated from hostnames or networks with whitespace(s).  
*And use fully qualified domain names to diminish spoofing attempts.*   
  
All entries in /etc/exports are exported with the root\_squash option ('root squashing') by default. This means that a root user on a client machine does not have root privileges (root access) to root-owned files on exported NFS filesystems/directories. It is not recommended to turn 'root squashing" off using the no\_root\_squash option!   
  
After you've made all your entries in /etc/exports, you can export all filesystems/directories using the following command:

# exportfs -a

To unexport all shared filesystems/directories, run:

# exportfs -ua

To see all shared filesystems/directories, run:

# showmount -e localhost

Export list for localhost:

/pub \*.subnet.example.com

/data/OracleArch rac3pub.example.com,rac2pub.example.com,rac1pub.example.com

**Using NFS over TCP**  
  
If you need NFS, it is recommended to use NFS over TCP since NFS over UDP is not very secure. All 2.4 and 2.6 kernels support NFS over TCP on the client side. Server support for TCP appears in later 2.4 kernels, and in all 2.6 kernels.   
  
To verify whether your server supports NFS over TCP, use the wire-test command (/usr/sbin/wire-test is part of the am-utils package). If your server supports NFS over TCP, the output looks like this:

# wire-test localhost

Network 1: wire="172.16.1.0" (netnumber=172.16.1).

Network 2: wire="172.16.1.1" (netnumber=172.16.1).

My IP address is 0xac100101.

NFS Version and protocol tests to host "localhost"...

testing vers=2, proto="udp" -> found version 2.

testing vers=3, proto="udp" -> found version 3.

testing vers=2, **proto="tcp" -> found version 2.**

testing vers=3, **proto="tcp" -> found version 3.**

#

If the server does not support NFS over TCP, the output will look like this:

# wire-test localhost

Network 1: wire="172.16.1.0" (netnumber=172.16.1).

Network 2: wire="172.16.1.1" (netnumber=172.16.1).

My IP address is 0xac100101.

NFS Version and protocol tests to host "localhost"...

testing vers=2, proto="udp" -> found version 2.

testing vers=3, proto="udp" -> found version 3.

testing vers=2, **proto="tcp" -> failed!**

testing vers=3, **proto="tcp" -> failed!**

#

To mount a shared directory using NFS over TCP, use the "proto=tcp" mount option:

# mount **-o proto=tcp** <nfs\_server\_name>:/pub /usr/local/pub

Make sure the target directory, in this example /usr/local/pub, exists on the client.   
  
You can verify the NFS over TCP mount using the mount command:

# mount

...

nfsserver:/pub on /usr/local/pub type nfs (rw,**proto=tcp**,addr=172.16.10.8)

...

To have the shared directory mounted on the client at boot time, use the /etc/fstab file.  
For the above example, the /etc/fstab entry could look like this:

nfsserver:/pub /usr/local/pub **nfs** rsize=8192,wsize=8192,timeo=14,intr,**tcp** 0 0

**Kernel Tunable Security Parameters**

The following list shows tunable kernel parameters you can use to secure your Linux server against attacks.   
  
For each tunable kernel parameters I will show the entry that needs to be added to the/etc/sysctl.conf configuration file to make the change permanent after reboots. To activate the configured kernel parameters immediately at runtime, use:

# sysctl -p

**Enable TCP SYN Cookie Protection**  
  
A "SYN Attack" is a denial of service attack that consumes all the resources on a machine. Any server that is connected to a network is potentially subject to this attack.   
  
To enable TCP SYN Cookie Protection, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.tcp\_syncookies = 1

**Disable IP Source Routing**  
  
Source Routing is used to specify a path or route through the network from source to destination. This feature can be used by network people for diagnosing problems. However, if an intruder was able to send a source routed packet into the network, then he could intercept the replies and your server might not know that it's not communicating with a trusted server.   
  
To disable Source Route Verification, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.conf.all.accept\_source\_route = 0

**Disable ICMP Redirect Acceptance**  
  
ICMP redirects are used by routers to tell the server that there is a better path to other networks than the one chosen by the server. However, an intruder could potentially use ICMP redirect packets to alter the hosts's routing table by causing traffic to use a path you didn't intend.   
  
To disable ICMP Redirect Acceptance, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.conf.all.accept\_redirects = 0

**Enable IP Spoofing Protection**  
  
IP spoofing is a technique where an intruder sends out packets which claim to be from another host by manipulating the source address. IP spoofing is very often used for denial of service attacks. For more information on IP Spoofing, I recommend the article [IP Spoofing: Understanding the basics](http://www.linuxexposed.com/internal.php?op=modload&name=News&file=article&sid=550).   
  
To enable IP Spoofing Protection, turn on Source Address Verification. Edit the /etc/sysctl.conf file and add the following line:

net.ipv4.conf.all.rp\_filter = 1

**Enable Ignoring to ICMP Requests**  
  
If you want or need Linux to ignore ping requests, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.icmp\_echo\_ignore\_all = 1

This cannot be done in many environments.   
  
**Enable Ignoring Broadcasts Request**  
  
If you want or need Linux to ignore broadcast requests, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.icmp\_echo\_ignore\_broadcasts = 1

**Enable Logging of Spoofed Packets, Source Routed Packets, Redirect Packets**  
  
To turn on logging for Spoofed Packets, Source Routed Packets, and Redirect Packets, edit the/etc/sysctl.conf file and add the following line:

net.ipv4.conf.all.log\_martians = 1

**References for Kernel Tunable Parameters**  
  
[Network Security with /proc/sys/net/ipv4](http://www.linuxsecurity.com/content/view/111337/65/)   
[IP Spoofing: Understanding the basics](http://www.linuxexposed.com/internal.php?op=modload&name=News&file=article&sid=550) 

**Checking File Permissions and Ownership**

**Default umask**  
  
The umask (user file-creation mode mask) command is a shell built-in command which determines the default file permissions for newly created files. This can be overwritten by system calls but many programs and utilities make use of umask.   
  
By default, Red Hat sets umask to 022 or 002 which is fine. If the name of the user account and the group account is the same and the UID is 100 or larger, then umask is set to 002, otherwise it's set to 022, see /etc/bashrc for bash shells.

$ id

uid=509(test) gid=510(test) groups=100(users),510(test) context=user\_u:system\_r:unconfined\_t

$ umask

0002

$

# id

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel) context=root:system\_r:unconfined\_t

# umask

0022

#

Here is an example how umask works:

$ umask 000

$ touch file1

$ ls -l file1

**-rw-rw-rw-** 1 oracle oinstall 0 Dec 26 19:24 file1

$ umask 002

$ touch file2

$ ls -l file2

**-rw-rw-r--** 1 oracle oinstall 0 Dec 26 19:24 file2

$ umask 022

$ touch file3

$ ls -l file3

**-rw-r--r--** 1 oracle oinstall 0 Dec 26 19:25 file3

$

For the bash shell you can find the setting of umask in /etc/bashrc. The /etc/bashrc file is for system-wide aliases and functions and is invoked by ~/.bashrc.   
  
**SUID/SGID Files**  
  
When the SUID (set user ID) or SGID (set group ID) bits are set on an executable, it executes with the UID or GID of the owner of the executable rather than that of the person executing it. This means that e.g. all executables that have the SUID bit set and are owned by root are executed with the UID of root. A good example is the passwd command that allows ordinary users to update the password field in the /etc/shadow file which is owned by root.   
  
But SUID/SGID bits can be misused when the SUID/SGID executable has a security hole. Therefore, you might want to search the entire system for SUID/SGID executables and document it. For example, ensure that code developers don't set SUID/SGID bits on their programs if it's not an absolute requirement. Very often you can use workarounds like removing just the executable bit for world/others. However, a better approach is to change the design of the software if possible.   
  
To search the entire system for SUID or SGID files, you can run the following command:

find / -path /proc -prune -o -type f -perm +6000 -ls

The -prune option in this example is used to skip the /proc filesystem.   
  
**World-Writable Files**  
  
World-writable files are a security risk since it allows anyone to modify them. Additionally, world-writable directories allow anyone to add or delete files.   
  
To locate world-writable files and directories, you can use the following command:

find / -path /proc -prune -o -perm -2 ! -type l -ls

The "! -type l" parameter skips all symbolic links since symbolic links are always world-writable. However, this is not a problem as long as the target of the link is not world-writable, which is checked by the above find command.   
  
World-Writable directories with sticky bit such as the /tmp directory do not allow anyone except the owner of a file to delete or modify it in this directory. The sticky bit makes files stick to the user who created it and it prevents other users from deleting and renaming the files. Therefore, depending on the purpose of the directory world-writable directories with sticky are usually not an issue. An example is the /tmp directory:

$ ls -ld /tmp

drwxrwxrw**t** 18 root root 16384 Dec 23 22:20 /tmp

The "t" mode, which denotes the sticky bit, allows files to be deleted and renamed only if the user is the owner of this file or the owner of the directory.   
  
**Unowned Files**  
  
Files not owned by any user or group might not necessarily be a security problem in itself. However, unowned files could pose a security problem in the future. For example, if a new user is created and the new users happens to get the same UID as the unowned files have, then this new user will automatically become the owner of these files.   
  
To locate files not owned by any user or group, use the following command:

find / -path /proc -prune -o -nouser -o -nogroup

**Checking Accounts**

**Checking for Unlocked Accounts**  
  
It is important that all system and vendor accounts that are not used for logins are locked.   
  
To get a list of unlocked accounts on your system, you can check for accounts that do NOT have an encrypted password string starting with "!" or "\*" in the /etc/shadow file. If you lock an account using passwd -l, it will put a '!!' in front of the encrypted password, effectively disabling the password. If you lock an account using usermod -L, it will put a '!' in front of the encrypted password. Many system and shared accounts are usually locked by default by having a '\*' or '!!' in the password field which renders the encrypted password into an invalid string.   
  
Hence, to get a list of all unlocked (encryptable) accounts, run:

# egrep -v '.\*:\\*|:\!' /etc/shadow | awk -F: '{print $1}'

Also make sure all accounts have a 'x' in the password field in /etc/passwd. The following command lists all accounts that do not have a 'x' in the password field:

# grep -v ':x:' /etc/passwd

A 'x' in the password fields means that the password has been shadowed, i.e. the encrypted password has to be looked up in the /etc/shadow file. If the password field in /etc/passwd is empty, then the system will not lookup the shadow file and it will not prompt the user for a password at the login prompt. 

**Checking for Unused Accounts**  
  
All system or vendor accounts that are not being used by users, applications, by the system or by daemons should be removed from the system. You can use the following command to find out if there are any files owned by a specific account:

# find / -path /proc -prune -o -user <account> -ls

The -prune option in this example is used to skip the /proc filesystem.   
  
If you are sure that an account can be deleted, you can remove the account using the following command:

# userdel -r <account>

Without the "-r" option userdel will not delete the user's home directory and mail spool (/var/spool/mail/<user>). Note that many system accounts have no home directory.   
  
**Single User Mode Password for root**  
  
Some admins suggest to add the following line to the /etc/inittab file to ensure that a root password is required for Single User Mode logons:

~~:S:wait:/sbin/sulogin

*This works but can be circumvented easily!* At the GRUB or LILO prompt you can tell the boot loader to alternate the init program by using the boot params "init=/bin/bash". This will place you at a root shell prompt without a password.

**Enabling Password Aging**

In general I do not recommend that the system enforces password expiration for system and shared accounts. This could lead to outages if an application's account expires:

# su oracle -c id

You are required to change your password immediately (password aged)

Changing password for test

(current) UNIX password:

Rather a corporate policy should govern password changes for system and shared accounts. But for individual user accounts the system should expire the passwords automatically. The following example shows how password expiration can be setup for individual user accounts.   
  
The following files and parameters in the table are used when a new account is created with the useraddcommand. These settings are recorded for each user account in the /etc/shadow file. Therefore, make sure to configure the following parameters before you create any user accounts using the useraddcommand: 

|  |  |  |  |
| --- | --- | --- | --- |
| /etc/login.defs | PASS\_MAX\_DAYS | 60 | Maximum number of days a password is valid. |
| /etc/login.defs | PASS\_MIN\_DAYS | 7 | Minimum number of days before a user can change the password since the last change. |
| /etc/login.defs | PASS\_MIN\_LEN | n/a | This parameter does not work. It is superseded by the PAM module "pam\_cracklib". See [Enforcing Stronger Passwords](http://www.puschitz.com/SecuringLinux.shtml#EnforcingStrongerPasswords) for more information. |
| /etc/login.defs | PASS\_WARN\_AGE | 7 | Number of days when the password change reminder starts. |
| /etc/default/useradd | INACTIVE | 14 | Number of days after password expiration that account is disabled. |
| /etc/default/useradd | EXPIRE |  | Account expiration date in the format YYYY-MM-DD. |

Ensure that the above parameters are changed in the /etc/login.defs and /etc/default/useraddfiles.   
  
When a user account is created using the useradd command, the parameters listed in the above table are recorded in the /etc/shadow file in the following fields:

<username>:<password>:<date>:PASS\_MIN\_DAYS:PASS\_MAX\_DAYS:PASS\_WARN\_AGE:INACTIVE:EXPIRE:

To create a new user account you can execute the following command:

useradd -c "Test User" -g users test

The -g option specifies the primary group for this account:

# id test

uid=509(test) gid=100(users) groups=100(users)

The settings in /etc/login.defs and /etc/default/useradd are recorded for the test user in the /etc/shadow file as follows:

# grep test /etc/shadow

test:!!:12742:7:60:7:14::

You can change the password aging any time using the chage command.   
  
*To disable password aging for system and shared accounts, you can run the following chage command:*

# chage -M 99999 <system\_account\_name>

To get password expiration information:

# chage -l <system\_account\_name>

For example:

# chage -l test

Minimum: 7

Maximum: 60

Warning: 7

Inactive: 14

Last Change: Jan 11, 2005

Password Expires: Mar 12, 2005

Password Inactive: Mar 26, 2005

Account Expires: Never

**Restricting Use of Previous Passwords**

The pam\_unix module parameter remember can be used to configure the number of previous passwords that cannot be reused. And the pam\_cracklib module parameter difok can be used to specify the number of characters hat must be different between the old and the new password.   
  
In the following example I will show how to tell the system that a password cannot be reused for at least 6 months and that at least 3 characters must be different between the old and new password.   
  
Remember that in the chapter [Enabling Password Aging](http://www.puschitz.com/SecuringLinux.shtml#EnablingPasswordAging) we set PASS\_MIN\_DAYS to 7, which specifies the minimum number of days allowed between password changes. Hence, if we tell pam\_unix to remember 26 passwords, then the previously used passwords cannot be reused for at least 6 months (26\*7 days).   
  
Here is an example. Edit the /etc/pam.d/system-auth file and add/change the following pam\_crackliband pam\_unix arguments:

auth required /lib/security/$ISA/pam\_env.so

auth sufficient /lib/security/$ISA/pam\_unix.so likeauth nullok

auth required /lib/security/$ISA/pam\_deny.so

account required /lib/security/$ISA/pam\_unix.so

account sufficient /lib/security/$ISA/pam\_succeed\_if.so uid < 100 quiet

account required /lib/security/$ISA/pam\_permit.so

password requisite /lib/security/$ISA/pam\_cracklib.so retry=3 minlen=8 lcredit=-1 ucredit=-1 dcredit=-1 ocredit=-1 difok=3

password sufficient /lib/security/$ISA/pam\_unix.so nullok use\_authtok md5 shadow remember=26

password required /lib/security/$ISA/pam\_deny.so

session required /lib/security/$ISA/pam\_limits.so

session required /lib/security/$ISA/pam\_unix.so

NOTE:  
If the /etc/security/opasswd doesn't exist, create the file.

# ls -l /etc/security/opasswd

-rw------- 1 root root 0 Dec 8 06:54 /etc/security/opasswd

**Locking User Accounts After Too Many Login Failures**

I do not recommend that the system automatically locks system and shared accounts after too many failed login or su attempts. This could lead to outages if the application's account gets locked due to too many login failures like in this example:

# su oracle -c id

su: incorrect password

#

This could be an easy target for a denial of service attack. *At*[*Restricting Direct Login Access for System and Shared Accounts*](http://www.puschitz.com/SecuringLinux.shtml#RestrictingDirectLoginAccessForSystemAndSharedAccounts)*I will show how to disable direct logins for system or shared accounts.*   
  
In the following example I will show how to lock only individual user accounts after too many failed su or login attempts.   
  
Add the following two lines highlighted in blue to the /etc/pam.d/system-auth file as shown below:

auth required /lib/security/$ISA/pam\_env.so

auth required /lib/security/$ISA/pam\_tally.so onerr=fail no\_magic\_root

auth sufficient /lib/security/$ISA/pam\_unix.so likeauth nullok

auth required /lib/security/$ISA/pam\_deny.so

account required /lib/security/$ISA/pam\_unix.so

account required /lib/security/$ISA/pam\_tally.so per\_user deny=5 no\_magic\_root reset

account sufficient /lib/security/$ISA/pam\_succeed\_if.so uid < 100 quiet

account required /lib/security/$ISA/pam\_permit.so

password requisite /lib/security/$ISA/pam\_cracklib.so retry=3

password sufficient /lib/security/$ISA/pam\_unix.so nullok use\_authtok md5 shadow

password required /lib/security/$ISA/pam\_deny.so

session required /lib/security/$ISA/pam\_limits.so

session required /lib/security/$ISA/pam\_unix.so

The first added line counts failed login and failed su attempts for each user. The default location for attempted accesses is recorded in /var/log/faillog.   
  
The second added line specifies to lock accounts automatically after 5 failed login or su attempts (deny=5). The counter will be reset to 0 (reset) on successful entry if deny=n was not exceeded. But you don't want system or shared accounts to be locked after too many login failures (denial of service attack). To exempt system and shared accounts from the deny=n parameter, I added the per\_user parameter to the module. The per\_user parameter instructs the module NOT to use the deny=n limit for accounts where the maximum number of login failures is set explicitly. For example:

# faillog -u oracle -m -1

# faillog -u oracle

Username Failures **Maximum** Latest

oracle 0 -1 Fri Dec 10 23:57:55 -0600 2005 on unknown

The faillog command with the option "-m -1" has the effect of not placing a limit on the number of failed logins. To instruct the module to activate the deny=n limit for this account again, run:

# faillog -u oracle -m 0

By default, the maximum number of login failures for each account is set to 0 which instructs pam\_tallyto use the deny=n parameter. The faillog manual page on my Red Hat system says that selecting maximum number of login failures of 0 will deactivate deny=n for the account. The PAM documentation, however, says that per\_user will only work if the .fail\_max field contains a non-zero value. After testing both values setting it to -1 worked. Maybe because it's read as a high unsigned value?   
  
To see failed login attempts, run:

# faillog

To unlock an account after too many login failures, run:

# faillog -u <user> -r

Make sure to test these changes thoroughly on your system using e.g. ssh and su, and make sure rootdoes not get locked!   
  
To lock/unlock accounts manually, you can run one of the following commands:

# passwd -l <user>

# usermod -L <user>

# passwd -u <user>

# usermod -U <user>

**NOTE:**  
  
Since the /var/log/faillog is owned by root and only root can write to the /var/log/faillog file, xscreensaver and vlock won't work correctly. Each time xscreensaver or vlock is executed as a non-root user, you won't be able to do an unlock since these programs can't write to /var/log/faillog. I don't have a good solution for that. I can only think of setting the SUID bits on these programs.

**Preventing Accidental Denial of Service**

**General**  
  
Linux allows you to set limits on the amount of system resources that users and groups can use. This is also very handy if bugs in programs accidentally use up too much resources, slow down the machine, or even render the system unusable. I've seen systems where incorrect settings have allowed programs to use up too much resources which made the server unresponsible for new connections or local logins (e.g. a program uses up all file handles on the system). This could become a security issue if someone is allowed to use up all resources and causes a denial of service attack. Depending on your environment you may want to review resource limits for user accounts and groups.   
  
**Example for Restricting System Resources**  
  
The following example shows a practical use of setting or restricting system resources for an Oracle user account. For a list of system resource settings, see /etc/security/limits.conf. It would be a good idea to review the default settings of system resource.   
  
Most shells like Bash provide control over various resources like the maximum allowable number of open file descriptors or the maximum number of processes available to a user. To see all shell limits, run:

ulimit -a

For more information on ulimit for the Bash shell, see man bash and search for ulimit.   
  
**Important Note:**  
Setting "hard" and "soft" limits in the following examples might not work properly when you login to oracle using a SSH session. It should work if you log in as root and su to oracle. Resource limits should also work if the application is started automatically during the boot process. But if you experience the problem that the changed resource limits in /etc/security/limits.conf are not applied when logging in through SSH, then you may have to try to set UsePrivilegeSeparation in /etc/ssh/sshd\_config to "no" and restart the SSH daemon by executing /etc/init.d/sshd restart. Unfortunately, privilege separation does not work properly with PAM on some Linux distributions. But also note that turning off privilege separation is not really recommended since it's a valuable security feature that has already prevented exploitation of SSH vulnerabilities.   
  
For example, to change the number of file handles or open files that the Oracle user can use, you have to edit the file /etc/security/limits.conf as root and make the following changes or add the following lines, respectively:

oracle soft nofile 4096

oracle hard nofile 63536

The "soft limit" in the first line defines the number of file handles or open files that the Oracle user will have after login. If the Oracle user gets error messages about running out of file handles, then the Oracle user can increase the number of file handles like in this example up to 63536 ("hard limit") by running the following command:

ulimit -n 63536

You can set the "soft" and "hard" limits higher if necessary. *Note that I do not recommend to set the "hard" limit for nofile for the oracle user equal to /proc/sys/fs/file-max. If you do that and the oracle user uses up all the file handles, then the whole system will be out of file handles. This could mean that you won't be able to initiate new remote logins any more since the system won't be able to open any PAM modules which are required for performing a login.*   
  
You also need to ensure that pam\_limits is configured in the file /etc/pam.d/system-auth, or in/etc/pam.d/sshd (for SSH), /etc/pam.d/su (for su), or /etc/pam.d/login (local logins and telnet) if you don't want to enable it for all logins, or if /etc/pam.d/system-auth does not exist like on SUSE. This is the PAM module that will read the /etc/security/limits.conf file. The entry should read like:

session required /lib/security/pam\_limits.so

Here are the two "session" entries I have in my /etc/pam.d/system-auth file:

session required /lib/security/pam\_limits.so

session required /lib/security/pam\_unix.so

Now login to the oracle account again since the changes will become effective for new login sessions only.

$ su - oracle

$ ulimit -n

4096

$

*Note that the ulimit options are different for other shells.*   
  
The default limit for oracle is now 4096 and the oracle user can increase the number of file handles up to 63536:

$ su - oracle

$ ulimit -n

4096

$ ulimit -n 63536

$ ulimit -n

63536

$

To make this change permanent, add "ulimit -n 63536" (for Bash) to the ~oracle/.bash\_profile file which is the user startup file for the Bash shell on Red Hat Linux (to verify your shell run: echo $SHELL). To do this you could simply copy/paste the following commands for the oracle's Bash shell:

su - oracle

cat >> ~oracle/.bash\_profile << EOF

ulimit -n 63536

EOF

**Miscellaneous**

**Host-Based Linux Monitoring and Intrusion Detection**  
  
Before you put a server into production or better, before you put a server on the network, you should have an integrity checker installed on your system that lets you check if unauthorized changes have been made. In this way, if an intruder compromises your system you will know what changed on your server. You should also have an Intrusion Detection Software (IDS) solution in place that alarms you about intrusions, as well as Intrusion Prevention software.   
  
It is outside the scope of this article to cover Linux Monitoring and Intrusion Detection solutions. There are lots of interesting articles out there to read and there are several good products available on the market including open source solutions. For a powerful open source network-intrusion prevention and detection system, see [Securing your system with Snort](http://www.redhat.com/magazine/013nov05/features/snort/).   
  
**Connect Accounting Utilities**  
  
Here is a list of commands you can use to get data about user logins:   
  
who         Shows a listing of currently logged-in users.   
w           Shows who is logged on and what they are doing.   
last        Shows a list of last logged-in users, including login time, logout time, login IP address, etc.   
lastb       Same as last, except that by default it shows a log of the file /var/log/btmp, which contains all the bad login attempts.   
lastlog     This command reports data maintained in /var/log/lastlog, which is a record of the last time a user logged in.   
ac          Prints out the connect time in hours on a per-user basis or daily basis etc. This command reads /var/log/wtmp.   
dump-utmp   Converts raw data from /var/run/utmp or /var/log/wtmp into ASCII-parsable format.   
  
Also check the /var/log/messages file.   
  
**Other**  
  
The following items may not necessarily be Linux security related but should be configured correctly on all audited Linux systems: 

* Resolver (/etc/hosts, /etc/resolv.conf, /etc/nsswitch.conf)
* NTP (/etc/ntp.conf)