Disable Unnecessary Services:

Disabling or removing unused programs and services from your host is the most effective way to limit threats originating from a remote host. Use your distributions package management tools to scan the list of installed packages, then remove those that are unnecessary.

• Many of the services running from **inetd** are legacy programs, which are hardly ever required, yet typically enabled by default. The file **/etc/inetd.conf** is used to specify which services are offered. Disable all services that you do not want to provide by commenting them out using the # character in the first column of the line.

• The **/etc/rc*.d** or **/etc/rc.d/rc*** directories contains shell scripts that control the execution of network and system services during runlevels. Rename or otherwise disable any that are not required or remove the package entirely. Red Hat users can use **/sbin/chkconfig --list** to list which services run in which runlevel, and **/sbin/chkconfig --del <name>** to disable a service.

If you don't understand what a particular service does, disable it until you find out. Use **netstat** and **ps** to confirm they have not been started after a reboot. Use **/bin/netstat -a -p --inet** to determine which are available and the process ID associated with them. A port scanner should also be used to get a view of what remote hosts see.

Checking Package Integrity:

The **md5sum** command is used to compute a 128-bit fingerprint that is strongly dependant upon the contents of the file to which it is applied. It can be used to compare against a previously-generated sum to determine whether the file has changed. It is commonly used to ensure the integrity of updated packages distributed by a vendor:

```
# md5sum package-name
995d4f40cda13eacd2beaf35c1c4d5c2  package-name
```

The string of numbers can then be compared against the MD5 checksum published by the packager. While it does not take into account the possibility that the same person that may have modified a package also may have modified the published checksum, it is especially useful for establishing a great deal of assurance in the integrity of a package before installing it.

Install and Configure OpenSSH:

OpenSSH is a replacement for **telnet** and **ftp** that eliminates eavesdropping, connection hijacking, and encrypts all communication between hosts. One of the most indepensible free security tools in existence.

• Install the OpenSSH and OpenSSL Packages:

```
openssh-<current-version>.rpm
openssh-server-<current-version>.rpm
openssh-clients-<current-version>.rpm
openssl-<current-version>.rpm
```

• Generate Public/Private Key Pair:

OpenSSH uses public key cryptography to provide secure authorization. Generating the public key, which is shared with remote systems, and the private key which is kept on the local system, is done first to configure OpenSSH.

```
orion$ ssh-keygen
Generating RSA keys:  ...ooooooO....oooooooO
Key generation complete.
Enter file in which to save the key (/home/dave/.ssh/identity):
Created directory '/home/dave/.ssh'.
Enter passphrase (empty for no passphrase): <passphrase>
Enter same passphrase again: <passphrase>
Your identification has been saved in /home/dave/.ssh/identity.
Your public key has been saved in /home/dave/.ssh/identity.pub.
The key fingerprint is:
ac:42:11:c8:0d:b6:7e:b4:06:6a:a3:a7:e8:2c:b0:12 dave@orion
```

• Copy Public Key to Remote Host:

```
host2$ mkdir -m 700 ~dave/.ssh
host2$ cp /mnt/floppy/identity.pub ~dave/.ssh/authorized_keys
```

• Log in to Remote Host:

The SSH client (**/usr/bin/ssh**) is a drop-in replacement for **rlogin** and **rsh**. It can be used to securely login to a remote host:

```
orion$ ssh host2
Enter passphrase for RSA key 'dave@orion': <passphrase>
Last login: Sat Aug 15 17:13:01 2000 from orion
No mail.
host2$
```

• Copy Files to Remote Host:

The OpenSSH package also includes **scp**, a secure and improved replacement for **rcp**. This allows you to securely copy files over a network.

```
orion$ scp /tmp/file.tar.gz host2:/tmp
Enter passphrase for RSA key 'dave@orion:
file.tar.gz  100% |*************************| 98304 00:00
```

It is also possible to encapsulate ordinarily insecure protocols such as IMAP and POP within SSH to prevent transmitting clear text passwords to your mail server. Additionally, the **rsync** incremental file transfer utility can use SSH to securely synchronize two hosts, backup data to a log server securely, or even securely connect two subnets across the Internet, effectively creating a virtual private network.

 http://www.guardiandigital.com

Apache Security:

• Limit Apache to listen only on local interface by configuring **/etc/httpd/conf/httpd.conf** to read:

```
Listen 127.0.0.1:80
```

• Use the following to disable access to the entire filesystem by default, unless explicitly permitted. This will disable printing of indexes if no **index.html** exists, server-side includes, and following symbolic links. Disabling symlinks may impact performance for large sites.

```
<Directory />
 Options None
 AllowOverride None
 Order deny,allow
 Deny from all
</Directory>
```

• Use the following to control access to the server from limited addresses in **/etc/httpd/conf/access.conf** to read:

```
<Directory /home/httpd/html>
 # Deny all accesses by default
 Order deny,allow
 # Allow access to local machine
 Allow from 127.0.0.1
 # Allow access to entire local network
 Allow from 192.168.1.
 # Allow access to single remote host
 Allow from 192.168.5.3
 # Deny from everyone else
 Deny from all
</Directory>
```

• Use the following to require password authentication when attempting to access a specific directory in **/etc/httpd/conf/access.conf**:

```
<Directory /home/httpd/html/protected>
 Order Deny,Allow
 Deny from All
 Allow from 192.168.1.11
 AuthName "Private Information"
 AuthType Basic
 AuthUserFile /etc/httpd/conf/private-users
 AuthGroupFile /etc/httpd/conf/private-groups
 require group <group-name>
</Directory>
```

Create the **private-groups** file using the following format:

```
group-name: user1 user2 user...
```

Create password entries for each user in the above list:

```
# htpasswd -cm /etc/httpd/conf/private-users user1
New password: <password>
Re-type new password: <password>
Adding password for user user1
```

Be sure to restart apache and test it. This will result in the enabling of double reverse lookups to verify the identity of the remote host. Remove the **-c** option to **htpasswd** after the first user has been added. Be sure the password file you create is not located within the **DocumentRoot** to prevent it from being downloaded.

Configuring TCP Wrappers:

Frequently used to monitor and control access to services listed in **/etc/inetd.conf**. The **in.ftpd** service might be wrapped using:

```
ftp stream tcp nowait root /usr/sbin/tcpd  in.ftpd -l -L -i -o
```

Before the **in.telnetd** daemon is spawned, **tcpd** first determines if the source is a permitted host. Connection attempts are sent to **syslogd**. All services should be disabled by default in **/etc/hosts.deny** using the following:

```
ALL: ALL
```

To send an email to the admin and report failed connection attempt:

```
ALL: ALL: /bin/mail \
   -s "%s connection attempt from %c" admin@mydom.com
```

Enable specific services in **/etc/hosts.allow** using the service name followed by the host:

```
sshd: magneto.mydom.com, juggernaut.mydom.com
in.ftpd: 192.168.1.
```

Trailing period indicates entire network should be permitted. Use **tcpdchk** to verify your access files. A syslog entry will be created for failed attempts. Access control is performed in the following order:

• Access will be granted when a daemon/client pair matches an entry in the **/etc/hosts.allow** file.

• Otherwise, access will be denied when a daemon/client pair matches an entry in the **/etc/hosts.deny** file.

• Otherwise, access will be granted.

A non-existing access control file is treated as if it were an empty file. Thus, access control will be turned off if no access control files are present!

Using RPM and dpkg:

The **/bin/rpm** program on Red Hat and derivitives and the **/usr/bin/dpkg** on Debian and derivitives are used to control the management of packages.

• Remove a package

```
# rpm -e <package-name>
# dpkg -r <package-name>
```

• List contents of entire package

```
# rpm -qvl <package-name.rpm>
# dpkg -c <package-name.deb>
```

• List all installed packages with info about each

```
# rpm -qvia
# dpkg -l
```

• List contents of a package

```
# rpm -qvpl <package-name.rpm>
# dpkg -c <package-name.deb>
```

• Print information about a package

```
# rpm -qpi <package-name.rpm>
# dpkg -I <package-name.deb>
```

• Verify package characteristics (basic integrity check)

```
# rpm -Va
# debsums -a
```

• Determine to which package a file belongs

```
# rpm -qf </path/to/file>
# dpkg -S </path/to/file>
```

• Install new package

```
# rpm -Uvh <package-name.rpm>
# dpkg -i <package-name.deb>
```

Configuring Syslog:

The **syslogd** is responsible for capturing logging information generated by system processes. The **klogd** is responsible for capturing logging information generated by the kernel. System logs provide the primary indication of a potential problem.

• Fine-tune the default **/etc/syslog.conf** to send log information to specific files for easier analysis.

```
# Monitor authentication attempts
auth.*;authpriv.*                       /var/log/authlog

# Monitor all kernel messages
kern.*                                  /var/log/kernlog

# Monitor all warning and error messages
*.warn;*.err                            /var/log/syslog

# Send a copy to remote loghost.  Configure syslogd init
# script to run with -r -s domain.com options on log
# server. Ensure a high level of security on the log
# server!
*.info                                  @loghost
auth.*;authpriv.*                       @loghost
```

• Restrict access to log directory and syslog files for normal users using:

```
# chmod 751 /var/log /etc/logrotate.d
# chmod 640 /etc/syslog.conf /etc/logrotate.conf
# chmod 640 /var/log/*log
```

Install and Configure Tripwire:

Tripwire is a program that monitors file integrity by maintaining a database of cryptographic signatures for programs and configuration files installed on the system, and reports changes in any of these files.

A database of checksums and other characteristics for the files listed in the configuration file is created. Each subsequent run compares any differences to the reference database, and the administrator is notified.

The greatest level of assurance that can be provided occurs if Tripwire is run immediately after Linux has been installed and security updates applied, and before it is connected to a network.

A text configuration file, called a policy file, is used to define the characteristics for each file that are tracked. Your level of paranoia determines the frequency in which the integrity of the files are checked. Administration requires constant attention to the system changes, and can be time-consuming if used for many systems. Available in unsupported commercial binary for Red Hat and similar.

```
# Create policy file from text file
/usr/TSS/bin/twadmin -m P policy.txt

# Initialize database according to policy file
/usr/TSS/bin/tripwire –init

# Print database
/usr/TSS/bin/twprint -m d

# Generate daily report file
/usr/TSS/bin/tripwire -m c -t 1 -M

# Update database according to policy file and report file
/usr/TSS/bin/tripwire --update --polfile policy/tw.pol \
   --twrfile report/<hostname>-<date>.twr
```

DNS Security:

• Zone transfers should only be permitted by master name servers to update the zone (domain) information in their slave servers. Failure to do so may result in IP numbers and hostnames being revealed to unauthorized users. Restrict queries to only public domains. Suitable for name servers with both public and private zones.

```
// Allow transfer only to our slave name server. Allow queries
// only by hosts in the 192.168.1.0 network.
zone "mydomain.com" {
    type master;
    file "master/db.mydomain.com";
    allow-transfer { 192.168.1.6; };
    allow-query { 192.168.1.0/24; };
};
```

• Deny and log queries for our version number except from the local host. The ability to determine the bind version enables an attacker to find the corresponding exploit for that version.

```
// Disable the ability to determine the version of BIND running
zone "bind" chaos {
    type master;
    file "master/bind";
    allow-query { localhost; };
};
```

The **./master/bind** file should then contain:

```
$TTL 1d
@     CHAOS   SOA     localhost. root.localhost. (
                1         ; serial
               3H         ; refresh
              15M         ; retry
               1W         ; expire
               1D )       ; minimum
               NS         localhost.
```

• Control which interfaces **named** listens on. Restricting the interfaces on which **named** runs can limit the exposure to only the necessary networks.

```
listen-on { 192.168.1.1; };
```

• Use Access Control Lists to classify groups of hosts with differing degrees of trust. The "internal" ACL label might be used to describe internal hosts that are permitted a greater degree of access to the information than other hosts might be. Before it can be used it must be defined:

```
acl "internal" {
       { 192.168.1.0/24; 192.168.2.11; };
};
```

It can then be used in "zone" statements or the main "options" statement:

```
zone "inside.mynet.com" {
       type master;
       file "master/inside.mynet.com";
       allow-query { "internal"; };
};
```

• Configure BIND to run as a normal user. Once BIND has been started, it has the ability to relinquish its privileges, and run as a user with limited abilities instead of **root**.

```
# useradd -M -r -d /var/named -s /bin/false named
# groupadd -r named
```

This account should be used for nothing other than running the name server. Ensure the zone files are readable by the **named** user. It is then necessary to modify the default **named** init script, typically found in **/etc/rc.d/init.d/named** on Red Hat or **/etc/init.d/named** on Debian:

```
/usr/sbin/named -u named -g named
```

It is also possible to run **named** in a "chroot jail" which helps to restrict the damage that can be done should **named** be subverted.

Critical System Files:

File/Directory	Perms	Description
/var/log	751	Directory containing all log files
/var/log/messages	644	System messages
/etc/crontab	600	System-wide crontab file
/etc/syslog.conf	640	Syslog daemon configuration file
/etc/logrotate.conf	640	Controls rotating of system log files
/var/log/wtmp	660	Who is logged in now. Use **who** to view
/var/log/lastlog	640	Who has logged in before. Use **last** to view
/etc/ftpusers	600	List of users that *cannot* FTP
/etc/passwd	644	List of the system's user accounts
/etc/shadow	600	Contains encrypted account passwords
/etc/pam.d	750	PAM configuration files
/etc/hosts.allow	600	Access control file
/etc/hosts.deny	600	Access control file
/etc/lilo.conf	600	Boot loader configuration file
/etc/securetty	600	TTY interfaces that allow root logins
/etc/shutdown.allow	400	Users permitted to ctrl-alt-del
/etc/security	700	System access security policy files
/etc/rc.d/init.d	750	Program start-up files on Red Hat systems
/etc/init.d	750	Program start-up files on Debian systems
/etc/sysconfig	751	System and network config files on Red Hat
/etc/inetd.conf	600	Internet SuperServer configuration file
/etc/cron.allow	400	List of users permitted to use **cron**
/etc/cron.deny	400	List of users denied access to **cron**
/etc/ssh	750	Secure Shell configuration files
/etc/sysctl.conf	400	Contains kernel tunable options on recent Red Hat