Step 4.4.4. Install an alternative MTA

Several other mail transfer agents are available to replace sendmail. Two popular aternatives are Qmail (www.qmail.org) by David Bernstein, and Postfix (www.postfix.org) by Wietse Venema. Both of these MTAs were designed and written from the ground up with security and performance in mind. It is beyond the scope of this guide to give details on installing and configuring either of these alternatives, but a wealth of information is available on the Internet.

Step 4.4.5. Secure the POP and IMAP daemons

For mail servers that collect all incoming mail for an organization, a common means to deliver the mail to clients is for them to retrieve the mail using the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP). POP is the older and simpler of the two protocols, providing basic commands for authentication, retrieval and deletion of mail messages from the mail server. IMAP is more flexible and supports creating, deleting, and renaming mail folders (mailboxes), searching, selective retrieval of message attributes and more.

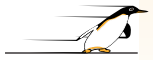
Step 4.4.5.1. Get the latest version of POP and IMAP daemons

Unfortunately, many POP and IMAP daemon implementations have been plagued with vulnerabilities that lead to remote root compromises of mail servers on many platforms. There are several well-known exploit programs available for cracking vulnerable Linux POP and IMAP daemons. Hopefully, most of the problems have been found and fixed, but it is very important to have the absolute latest version of the daemon program installed on the server.

Step 4.4.5.2. Control access to POP and IMAP with TCP wrappers

POP/IMAP is traditionally run out of `inetd`, so access control through TCP wrappers is easy and very important. Limit access to only those hosts that have a legitimate need for the service. For a mail hub that holds mail for the entire `example.org` domain and 128.184 network, and delivers it to clients with POP version 3 or IMAP, put this in `/etc/hosts.allow` (remember from Step 3.2.1 above that `/etc/hosts.deny` has only "`ALL: ALL`" for denial of all services by default):

```
ipop3d: .example.org 128.184.
imapd:  .example.org 128.184.
```



▲ **Step 4.4.5.3. Install an alternative POP or IMAP daemon**

There are several alternative POP daemons available. One of the most popular is Qpopper from Qualcomm, Inc. This version supports all POP3 extensions, APOP, and Kerberos V4. See Appendix for a URL to the Qpopper home page. As of this writing the current stable release is 2.53, and version 3.0 is in beta release. A popular IMAP daemon replacement is Cyrus IMAPD from the Carnegie Mellon Enterprise Electronic Mail Project. See Appendix A for more information on where to locate these packages.

▲ **Step 4.4.5.4. Install an SSL wrapper for secure POP/IMAP connections**

There are several third-party open source programs that wrap TCP services with the Secure Socket Layer (SSL) protocol to provide strong authentication and end-to-end encryption. For a list, see `http://www.openssl.org/related/apps.html`. Two applications that are useful for providing secure POP/IMAP connections are `stunnel` and `sslwrap` (see Appendix A for URLs to these packages). Both packages require the SSLeay or OpenSSL packages. See Step 4.9.5 for information on downloading and compiling OpenSSL.

STEP 4.5 PRINTING SERVICES

Red Hat Linux ships with the Berkeley line printer system. Over the last few years, a few buffer overflow exploits have been found in the `lpr` and `lprm` commands, both used on the client side of the connection. If you are running a version of Red Hat Linux earlier than 6.0, be sure to update to the latest version of the `lpr` package.

■ **Step 4.5.1. List allowed remote hosts in `/etc/hosts.lpd`**

Put the names of hosts allowed to use this print server in `/etc/hosts.lpd`. You can get the same effect by listing them in `/etc/hosts.equiv`, but that method has serious implications in conjunction with the BSD “r” programs, `rsh`, `rlogin`, etc. (See Step 4.2 for replacing the “r” programs with SSH).

■ **Step 4.5.2. Replace Berkeley lpr/lpd with LPRng**

A popular alternative to the Berkeley lpr/lpd system is LPRng. It is compliant with the RFC1179 requirements for network printing, gives a great deal of flexibility to the administrator for defining permissions for specific actions, and supports authentication with Kerberos and PGP. Caldera OpenLinux and Debian ship with LPRng already, administrators of those distributions can skip the installation step.

▲ **Step 4.5.2.1. Download and install LPRng**

Links to the latest stable version can be found at `http://www.astart.com/LPRng.html`. Download this version, and extract the files from the compressed tar file. Follow the instructions in the file INSTALL in the source directory to compile and install LPRng.

▲ **Step 4.5.2.2. Set remote hosts and/or networks that are allowed access**

The file `/etc/lpd.perms` controls access to specific hosts or networks in addition to controlling specific operations, for instance it is possible to allow or deny specific users from removing jobs from the print queue. The default `/etc/lpd.perms` installed by LPRng is well commented and the package comes with lots of documentation. See the manual page for `lpd.perms(5)` for more information on the permissions configuration file.

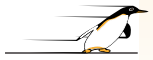
To restrict access to `lpd` to only those hosts in the `example.org` domain, or network `128.184`, place the following in `/etc/lpd.perms`:

```
REJECT SERVICE=X NOT REMOTEIP=128.184.0.0/255.255.0.0
REJECT SERVICE=X NOT REMOTEHOST=*.example.org
```

STEP 4.6 NETWORK FILE SYSTEM

NFS by its very design has some serious security problems. The RPC service depends on simple UID/GID and IP authorization for permissions, all of which are easily spoofed. Sun Microsystems, the developer of the NFS protocol, has enhanced it with Secure RPC that uses cryptographic authentication, but to date there is no Linux implementation of Secure RPC available.

Red Hat 6.0 has moved away from the “user-space” NFS implementation to a “kernel-space” implementation, `knfsd`. Just recently, security problems were discovered in the user-space NFS code. Users of earlier Red Hat versions and other distributions that use the user-space code should upgrade to the latest version. This is not to say that there are no problems with the kernel-space code. The version shipped with Red Hat 6.0 has a bug that makes it impossible for non-Linux systems to mount directories on Linux systems. See `ftp://ftp.kernel.org/pub/linux/devel/gcc/` for the latest version, which corrects this problem.



Step 4.6.1. Set access to RPC services in `/etc/hosts.allow`

RPC services are registered and accessed through the `portmap` daemon. The version of `portmap` supplied with all Linux distributions uses the TCP wrapper library to allow or deny access to RPC services, such as NFS. The only difference from other TCP-wrapped services is that requests from the local host are always authorized, regardless of what is in the `hosts.allow` or `hosts.deny` files, and `portmap` does not do hostname lookups, so `hosts.allow` must specify the hosts by their IP address, or network number/netmask.

For example, to allow NFS access to `client1.example.org` and `client2.example.org`, IP addresses `192.168.1.10` and `192.168.1.11` respectively, and the entire `128.184` network, put this in `/etc/hosts.allow`:

```
portmap: 192.168.1.10 192.168.1.11 128.184.0.0/255.255.0.0
```

Step 4.6.2. Limit exports to specific machines with specific permissions

The file `/etc/exports` controls which directories are exported for NFS mounting and the hosts that are allowed to mount them. The format is:

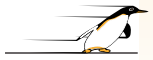
```
exported-dir     [host](options)
```

If you do not provide a host, the directory is exported to any host on the Internet, so always provide a host name, and remember that the hostname's IP address must match that given in `/etc/hosts.allow`.

Unless there is a compelling reason to do otherwise, export the directory read-only with the "`ro`" option. File permissions on the exported directory are determined by the UID/GID of the user on the remote host that mounts the directory. If the remote host is compromised, the attacker can emulate any UID/GID she wishes. Therefore, any file on a read-write exported NFS directory can be created, altered, or deleted. The administrator of an NFS server should not allow NFS access to hosts outside her administrative control except in read-only mode.

If at all possible, avoid allowing a NetWare server to mount any UNIX filesystem, Linux, Solaris, whatever. NetWare NFS services present significant security problems, such as root read-write privilege regardless of the export restrictions.

See the manual page for `exports(5)` for details on the options for `/etc/exports`.



STEP 4.7 SERVER MESSAGE BLOCK (SMB) SAMBA SERVER

The SMB protocol is the core of the Common Internet File System developed by Microsoft for file and printer sharing. The idea behind Samba is to make a UNIX server look exactly like any NT box to its clients in the network neighborhood. Setting up the Samba software itself is relatively simple, but there are a number of nuances to successfully integrating it into the office environment. Samba server administrators should review the extensive documentation that comes with the software, and the book "Samba: Integrating UNIX and Windows," Blair, SSC, Inc. ISBN 1-57831-006-7.

Step 4.7.1. Get the latest version of Samba

As of this writing, updated packages for Red Hat version 6.0 have been issued for Samba to correct security problems. Check the updates for your distribution, and make sure that you have installed Samba version 2.0.5a or later.

Step 4.7.2. Limit access to specific hosts

Edit the file `/etc/smb.conf` in the `[global]` section to set up the list of hosts that are allowed access to the Samba server and the interfaces that the Samba server will listen on:

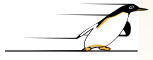
```
hosts allow = .example.org 128.184.
interfaces 192.168.0.1/24 127.0.0.1/32
bind interfaces only = true
```

Step 4.7.3. Use encrypted passwords

Before setting up Samba to use encrypted passwords, read `/usr/doc/samba-2.0.5a/docs/textdocs/ENCRYPTION.txt`.

Red Hat Linux 6.0 defaults to looking for Samba passwords in `/etc/smbpasswd`. Other distributions may use `/etc/samba.d/smbpasswd`, or a different path entirely. The path is set in the configuration file `/etc/smb.conf`. You can create a template `/etc/smbpasswd` with the following command:

```
[root]# mksmbpasswd.sh < /etc/passwd >/etc/smbpasswd
[root]# chmod 600 /etc/smbpasswd
```



Edit the template to remove entries for system accounts like bin, daemon, and ftp. The administrator should set default passwords for each of the accounts. If you wish to have your users set their own password, you will need to edit `/etc/smbpasswd` and put the string "NO PASSWORD" in the first eleven characters in the password field, leaving the remaining 21 "X" characters. Then enable null passwords with the line:

```
null passwords = true
```

in the `[global]` section of `/etc/smb.conf`, and restart the smb server. If you decide to use this method, give the users only a short period of time to reset their passwords. If any entries in `/etc/smbpasswd` contain the string "NO PASSWORD" after the deadline, then set passwords for the users yourself.

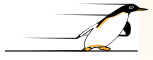
Step 4.7.4. Remove "guest" or anonymous shares

The default `/etc/smb.conf` that comes with Red Hat 6.0 only enables user-level shares of the home directories for each user on the local host. Other distributions may enable other publicly-readable guest shares. Before enabling Samba, carefully inspect the shares defined in `/etc/smb.conf` and disable any that are not absolutely necessary. For the remaining shares allow write access only when absolutely necessary. Consider setting write permissions for only those users that need the permission, not for any user connected to the service.

Step 4.7.5. Set default file creation masks

The default file creation mask makes files that are world-readable. Edit `/etc/smb.conf` and search for the following lines and change the masks to clear the "other" permission bits:

```
create mask = 0770
directory mask = 0750
```



STEP 4.8 CENTRAL SYSLOG HOST

As discussed in Step 2.8.3, a centralized logging host is important for organizations with multiple machines. They provide an additional line of defense in the preservation of information and evidence about system anomolies and break-ins. Because of this important security function, logging hosts should be as secure as possible. No other services, except perhaps SSH for remote administration, should be running, or even installed, on the machine. Only the absolute minimum set of system utilities should be installed. The host should have a large, fast disk dedicated to the `/var/log` directory for the collection of the syslog messages. A fast, high-capacity backup device is also advisable. Log rotation (see Step 2.8.4) should be turned off, or set to a long interval, and all log files should be retained for a long period of time.

As was mentioned in Step 2.8 above, there are several alternative syslog daemon implementations available that are more secure than the stock `sysklogd` provided on the Red Hat installation CD-ROM. See Appendix A under Step 2 for references to some of the alternatives.

Step 4.8.1. Configure syslogd to accept remote log messages

The default behavior of the syslog daemon in Red Hat Linux is not to accept remote log messages. This is contrary to the behavior of most BSD-style log daemons. To turn this feature on, edit `/etc/rc.d/init.d/syslog`, and add the "-r" option to the line that starts the syslog daemon:

```
start)   ...
  daemon syslogd -r
```

Then, restart the syslog service:

```
[root]# /etc/rc.d/init.d/syslog restart
```

Step 4.8.2. Configure log rotation

As discussed in Step 2.8.4, the `logrotate` program is designed to rotate, preserve, and delete log files after a certain period of time, or when the files reach a certain size. For a loghost, log rotation should be turned off (by deleting `/etc/cron.daily/logrotate`), or the logrotate configuration file should be edited to preserve the log files for a much longer period of time. For example, if your organization's security policy states that the logs must be kept for a year, edit `/etc/logrotate.conf` and change the first few lines to read:

```
# rotate log files monthly
monthly

# keep a years worth of backlogs
rotate 12
```

Red Hat Linux, and most other distributions, has a package called "`anonftp`" that sets up an anonymous FTP directory with the proper permissions for secure operation. To make sure, check in `/home/ftp`; the `bin` and `etc` directories should be owned by UID/GID root, mode 111 (execute-only). The `pub` directory should be owned by UID root, GID ftp, mode 02555 (set-GID, read-and-execute-only).

Step 4.9.1. Limit access with TCP wrappers

The FTP daemon is invoked through `inetd` and protected by TCP wrappers. If the FTP server is only meant to provide data to a limited set of machines, like your local domain or network, put the restriction in `/etc/hosts.allow`:

```
in.ftpd: .example.org 128.184.
```

A general anonymous FTP server will be accessible to the world:

```
in.ftpd: ALL
```

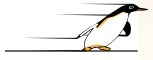
Note that TCP wrappers will log all connections, so that you can monitor them in the log file `/var/log/secure`. The files that are transferred are logged in `/var/log/xferlog`.

Step 4.9.2. Limit permitted operations in `/etc/ftpaccess`

The WU-FTP daemon supplied with Red Hat Linux allows fine-tuned control through the `/etc/ftpaccess` configuration file. See the manual page for `ftpaccess(5)` for more information. In this file, you can define special classes of users based on where they are connecting from, the number of simultaneous users, limit the operations allowed by classes of users, and much more.

If a writable directory is required (see below), anonymous users can be precluded from modifying the contents, regardless of the directory permissions, by putting the following into `/etc/ftpaccess`:

```
chmod          no  guest,anonymous
delete         no  guest,anonymous
overwrite      no  guest,anonymous
rename         no  guest,anonymous
```



Step 4.9.3. Protect incoming directory

In general, it is never a good idea to allow write access to an anonymous FTP directory, but sometimes it must be done. Traditionally this is called the "incoming" directory. The Washington University FTP daemon has a number of control features that will help to keep the incoming directory from turning into an illegal "warez" site.

First, create the incoming directory with write, but not read, access:

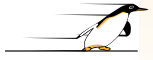
```
[root]# mkdir -m 333 /home/ftp/incoming
```

Then, edit `/etc/ftpaccess` and add these lines:

```
path-filter   anonymous   /etc/pathmsg  ^[-A-Za-z0-9._]*$ ^\.     ^-
upload        /home/ftp   /incoming     yes    root   ftp    0600  nodirs
noretrieve    /home/ftp/incoming/
```

The first line restricts upload file names to letters, numbers, hyphen, period, and underscore, and it restricts file names from starting with a period or hyphen (so the anonymous user can not create a file called "`...`" for instance). The second line says that files uploaded to the incoming directory are allowed, that files will have UID root, GID ftp, mode 0600, and that the user is not allowed to create subdirectories. The last line denies downloads from the incoming directory entirely, so once a file is written there, another anonymous user can't get it. Again, read the manual page for `ftpaccess(5)` for more information about these and other control mechanisms.

Regardless of the protections, the incoming directory should be reviewed daily, and all files stored there moved to another directory out of the anonymous directory tree. Write a cron job to check the directory each night, perform the move and notify the administrator about any files found there.



STEP 4.10 HYPERTEXT TRANSFER PROTOCOL (HTTP) SERVER

All major Linux distributions come with the Apache HTTP server software. Apache is designed for flexibility and has a wealth of features. Most security-related settings are in the main configuration files found in `/etc/httpd/conf` in Red Hat Linux. The file `httpd.conf` sets up basic operating parameters for the HTTP daemon; `access.conf` sets up basic access rules; and `srm.conf` sets up basic server configuration parameters like the document root, aliased directories, CGI script directories, and icons for indexes.

See the Apache manual included with Red Hat 6.0, `http://localhost/manual/index.html`, especially the Security Tips, `http://localhost/manual/misc/security_tips.html`, and the description of how sections work, `http://localhost/manual/sections.html`. The Apache manual is also available from the Apache Web site at `http://www.apache.org/docs/`.

Step 4.10.1. Set basic access to default deny

In `access.conf`, set the root directory access and options to:

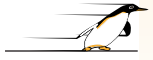
```
<Directory />
Options None
AllowOverride None
order deny,allow
deny from all
</Directory>
```

So by default, access to all directories and files on the server is denied.

Step 4.10.2. Selectively open access to specific directories

In the remainder of `access.conf`, specify only the access and options absolutely necessary. For an Intranet server, access should be allowed only to IP addresses on the internal network:

```
<Directory /home/httpd/html>
Options None
AllowOverride None
order deny,allow
deny from all
allow from 192.168.
</Directory>
```



Step 4.10.3. Selectively allow options on specific directories

Consider the implications of allowing any of the following Options before you set them in the `<Directory>` directive:

`ExecCGI`	Should only be allowed for CGI directories
`FollowSymLinks`	If users have write access to the HTML directories, they can set symbolic links to areas that contain sensitive data
`Includes`	Server side includes can be used to bypass default file access restrictions
`IncludesNOEXEC`	Safer version of Includes that disables the `#exec` statement and `#include` of CGI scripts
`Indexes`	The daemon will print a directory listing for any directory without an index file (index.html). This may expose the names of data files ordinarily hidden

Step 4.10.4. Selectively use `.htaccess` to override access control

Changing the `AllowOverride` directive from `None` to `All` tells Apache to look for a file named `.htaccess` in the directory holding the requested file, and to interpret the contents to override any of the default access and options directives for that directory. The advantage is that access and options can be set dynamically without editing the configuration file and restarting the server. The disadvantage is that security and options settings are no longer specified in a single place, but scattered around the directory structure. This can make it more difficult to audit the security policy for the Web site. In addition, there is a performance hit for every access, since the server has to look for, open and interpret the contents of the file for every request.

Step 4.10.5. Use password protection for sensitive data

Apache allows for protecting directories and/or files with username/password authentication. This provides a moderate level of security, although the username and password are passed over the network in the clear and are subject to being sniffed. See the locally installed Apache manual, `http://localhost/manual/mod/directives.html` for more information on the `AuthType`, `AuthUserFile`, and other authentication directives.

Step 4.10.6. Use SSL for secure HTTP communications

The Secure Sockets Layer (SSL) protocol provides strong authentication and end-to-end encryption for TCP sockets. It is most often used in conjunction with secure Web applications. Open source software implementations of SSL exist as compile-time extensions to the Apache web server.

Note: Administrators of commercial Web sites in the U.S. and other countries may be restricted by patent law and licensing from using SSL's RSA encryption software. Operators of commercial Web sites should seriously consider purchasing a commercial SSL-capable Web server. See Appendix A for some pointers to popular secure Web server vendors.

Note also: Commercial Web sites should have digital certificates from recognized Certificate Authorities. The two primary sources for commercial certificates are Thawte Consulting (`http://www.thawte.com`) and Verisign, Inc. (`http://www.verisign.com`).

Step 4.10.6.1. Download OpenSSL and `mod_ssl`

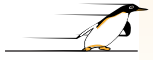
Extract the source code for the Apache server from the SRPM, or download the latest version from `ftp://ftp.apache.org`. The version numbers in the example below were the latest as of the writing of this guide. Check the FTP sites for the latest versions of each package.

```
[root]# cd /usr/local/src
[root]# wget ftp://ftp.apache.org/dist/apache_1.3.6.tar.gz
[root]# wget ftp://ftp.modssl.org/source/mod_ssl-2.3.10-1.3.6.tar.gz
[root]# wget ftp://ftp.openssl.org/source/openssl-0.9.3a.tar.gz
[root]# tar xzf apache_1.3.6.tar.gz
[root]# tar xzf mod_ssl-2.3.10-1.3.6.tar.gz
[root]# tar xzf openssl-0.9.3a.tar.gz
```

Step 4.10.6.2. Build OpenSSL

The Makefile for OpenSSL assumes that the Perl interpreter is in `/usr/bin/perl5`. Either edit the Makefile to change this to `/usr/bin/perl`, or redefine it on the command line. The latter method is illustrated below:

```
[root]# cd openssl-0.9.3a
[root]# make links PERL=/usr/bin/perl
[root]# ./Configure gcc
[root]# make
[root]# cd ..
```



▲ Step 4.10.6.3. Build Apache with `mod_ssl` module

Configuration of Apache to use `mod_ssl` can be done directly from the `mod_ssl` source directory:

```
[root]# cd mod_ssl-2.3.10-1.3.6
[root]# ./configure --with-apache=../apache_1.3.6 \
--with-ssleay=../openssl-0.9.3a --prefix=/usr/local/apache
[root]# cd ../apache_1.3.6
[root]# make
[root]# make certificate TYPE=test
[root]# make install
```

▲ Step 4.10.6.4. Start Apache with `mod_ssl` and test

Before starting the SSL version of Apache, stop the old daemon from running.

```
[root]# /etc/rc.d/init.d/httpd stop
[root]# /usr/local/apache/sbin/apachectl startssl
```

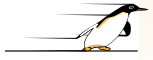
To test the server, from another host, enter:

```
[root]# netscape https://server-name/
```

After `netscape` prompts to accept the test certificate, a screen that says “It Worked!” should be displayed.

▲ Step 4.10.6.5. Read the `mod_ssl` documentation

Read `https://server-name/manual/mod/mod_ssl/` thoroughly. You may especially want to read the FAQ in this manual to learn how to create digital certificates.



Once your system has been configured to increase security, you can optimize the system for performance. Though it is beyond the scope of this document to discuss every aspect of Linux performance, a few optimizations will yield substantial performance gains.

STEP 5.1 KERNELS: THOUGHTS ABOUT CONFIGURATION, RECOMPILING, AND INSTALLING A NEW KERNEL

Like system partitioning, the use of kernel modules is somewhat of a religious debate. Red Hat and most other Linux distributions use modularized kernels to make the installation of Linux simple and recognition of hardware as plug-and-play as possible. Only the most basic operations are compiled into the kernel, and modules are loaded to implement support for specific hardware and software features. Kernel modules allow for the installation and removal of kernel features, such as IP Aliasing, without the need to re-compile the kernel or reboot the system.

On the flip side, the use of kernel modules is just one more thing that can break. Monolithic kernels only support the required hardware and features, yielding better performance. Kernel modules increase the risk of a security compromise through a rogue module. The module vs. monolith choice is yours but there are benefits to both methods.

Regardless of method, Red Hat 6.0 debuted with a stable version of the Linux 2.2.x kernel with great improvements in stability, performance and hardware support. Though it is out of the scope of this document, it is recommended that users read the KERNEL-HOWTO document and consider compiling a new kernel with built-in support for the specific hardware on the system.

Although newer kernels are generally better in terms of performance and reliability, some versions can introduce minor, and sometimes major, bugs. It's important to always keep recent backups and always test a new kernel before using it in an operational environment. Mission critical systems should stay away from the odd-numbered development kernels, 2.1.x, 2.3.x, and run only the latest version of the "stable" even numbered kernels, 2.0.x and now 2.2.x.

Out of the box, Red Hat, and most distributions, does not take full advantage of modern hardware. By minimizing optimizations, Linux will run on more legacy hardware. By tuning a step at a time, reliability and stability are maintained and performance is increased.

Step 5.2.1. TCP/IP Receive Window size

The TCP/IP acknowledge (ACK) receive window or RWIN defaults to a value of 0. For modern day Ethernet and Fast Ethernet networks, this is less than optimal. To fix this, edit `/sbin/ifup` and make the following changes:

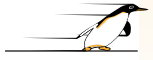
Line 110:
```
route add -net ${NETWORK} netmask ${NETMASK} ${DEVICE}
```
to:
```
route add -net ${NETWORK} netmask ${NETMASK} window 16384 ${DEVICE}
```

Line 112:
```
route add -host ${IPADDR} ${DEVICE}
```
to:
```
route add -host ${IPADDR} window 16384 ${DEVICE}
```

Line 117:
```
route add default gw ${GATEWAY} metric 1 ${DEVICE}
```
to:
```
route add default gw ${GATEWAY} metric 1 window 16384 ${DEVICE}
```

Line 125:
```
route add default gw ${GATEWAY} ${DEVICE}
```
to:
```
route add default gw ${GATEWAY} window 16384 ${DEVICE}
```

Line 134:
```
route add default gw $gw ${DEVICE}
```
to:
```
route add default gw $gw window 16384 ${DEVICE}
```



STEP 5.3 PACKET FIREWALLS AND LINUX IP MASQUERADING

A Linux server running a well-configured firewall is one of the most effective ways to protect the local server and any internal networks behind it. Linux 2.2.x kernels have a very stable packet firewall implementation that is administrated through a tool called `ipchains`. This tool gives the administrator the flexibility to define the type of traffic allowed in and out of the firewall. Note however, that kernels prior to version 2.2.11 had a bug in the IP fragmentation code that allowed an attacker to send carefully crafted IP fragments that would bypass `ipchains` rules. We recommend that administrators of enterprise systems install the latest version of the kernel (version 2.2.12 at the time of this writing).

Firewall technology is a book in itself (for example, “Building Internet Firewalls” Chapman & Zwicky, 1995, O’Reilly & Associates, ISBN 1-56592-124-0), and far beyond the scope of this guide. See the Resouces for a URL to the IPCHAINS and IP-MASQ HOWTOs for an in-depth discussion.

Step 5.3.1. Getting more from your external connection with IP Masquerade

A firewall protects standalone servers very well but it can also protect internal computers. A very common example of this configuration would be a person or company connecting their home/office network to the Internet. A wrinkle to this setup is that TCP/IP addresses used on the Internet are becoming a scarce and valued commodity. Thus, getting TCP/IP addresses for all of your internal computers can cost quite a bit of money. Network address translation (NAT) was developed to conserve Internet TCP/IP addresses while still allowing internal computers to access the Internet. For Linux, a form of NAT was developed called IP Masquerade, which is in common use by many Linux users today.

Step 5.3.2. A strong `/etc/rc.d/rc.firewall` ruleset

See Appendix D for the complete listing of a strong packet firewall script for an IP Masqueraded machine connected to the Internet via a Ethernet connection to a DSL or Cable modem, or another persistent network connection. There are extensive embedded comments to explain what the various rules do and how to modify the rules to suit your specific needs.

Step 5.3.3. Double check, install, and test the firewall

Once you have the `/etc/rc.d/rc.firewall` ruleset copied into the machine, it **very** important to tailor it to your specific environment. Make sure that the names of the internal and external interfaces are correct. Make sure that the TCP/IP addressing scheme of your internal network is configured properly.

Next, test the ruleset to be sure that it loads without errors, does not block any specific traffic that you need for your environment, and that it loads upon every reboot of the server.

Step 5.3.3.1. Make the ruleset executable

To be able to run the ruleset, you need to make the `/etc/rc.d/rc.firewall` script executable:

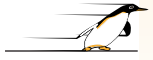
```
[root]# chmod 700 /etc/rc.d/rc.firewall
```

Step 5.3.3.2. Load the ruleset while at the console of the Linux server

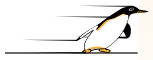
As shown in Step 2.8, monitor the console logs on `VTY7` and `VTY8`. If the Firewall script runs without any visible errors, the system should immediately begin to block traffic that you want. The console logs will show you what **is** and **isn't** blocked.

When you execute `/etc/rc.d/rc.firewall` the output of the script should look something like the following:

```
[root]# /etc/rc.d/rc.firewall
```



```
Loading IPCHAINS Firewall Version 3.12
-------------------------------------------------------------------------------
External IP: 192.168.0.14
External broadcast: 192.168.1.255
Default GW: 192.168.0.1
-------------------------------------------------------------------------------
  - Adding multicast route.
SIOCADDRT: File exists
  - Enabling IP forwarding.
  - Enabling dynamic TCP/IP address hacking.
  - Changing IP masquerading timeouts.
  - Loading masquerading modules.
  - Flushing all old rules and setting all default policies to REJECT
-------------------------------------------------------------------------------
Input Rules:
  - Setting input filters for traffic on the internal LAN.
  - Setting input filters for specific internal hosts.
  - Setting input filters for traffic from the external interface.
  - Setting input filters for public services (all interfaces).
  - Setting input filters for explicit external hosts.
  - Final input catch all rule.
-------------------------------------------------------------------------------
Output Rules:
  - Setting output filters for traffic on the internal LAN.
  - Setting output filters for specific internal hosts.
  - Setting input filters for traffic to the external interface.
  - Setting output filters for public services (all interfaces).
  - Reject specific outputs.
  - Setting output filters for explicit external hosts.
  - Final output catch all rule.
-------------------------------------------------------------------------------
Forwarding Rules:
  - Enable IP Masquerading from the internal LAN.
-------------------------------------------------------------------------------
Firewall implemented.
```



▲ Step 5.3.3.3. Test the firewall ruleset

While at the console of the Linux server, try various network applications like TELNET, FTP, SSH, etc. to computers on the Internet and on the internal LAN. If that test goes well, try a similar set of tests on one of the internal computers. If, for any reason, the machines don't respond properly, check the `/var/log/messages` file or on the console, watch `VTY7` for firewall hits.

■ Step 5.3.4. Analyze a typical IPCHAINS firewall ruleset hit

So you are starting to see strange IPCHAINS entries in `/var/log/messages`. Realistically, this is happening either because your firewall ruleset is too restrictive for your environment or it is working properly and blocking unwanted traffic. For example, once the firewall ruleset is installed, DHCP no longer works. Here is the Resulting IPCHAINS firewall log:

```
Aug  8 21:00:32 testpc kernel: Packet log: output REJECT eth0 PROTO=17
192.168.0.14:68 192.168.0.1:67 L=604 S=0x00 I=41786 F=0x0000 T=64
```

There is a LOT of information in this one line so let's break it down by section. Refer back to the logged hit as you read this.

- This firewall hit occurred on Aug 8 at 9:00:32pm on the "testpc" computer
- The ruleset hit was on the "output" side of the firewall (other hits can be "input" or "forward")
- The hit was then "REJECTed" (others can say "DENY" or "ACCEPT")
- The firewall hit was on the "eth0" interface
- This hit occurred over the "PROTO=17" protocol or UDP (other common protocol numbers are "1" for ICMP and "6" for TCP. A full list of other protocol numbers can be found in `/etc/protocols`)
- The source the hit came from was IP address 192.168.0.14 on port 67, which is for DHCP client (a full list of port numbers can be found in `/etc/services`)
- The destination address was IP 192.168.0.1 to port 68 which is DHCP Server
- The packet's length was 604 bytes long
- This packet did not have any "Type of Service" (TOS) set (don't worry about TOS, it's a low-level network flag denoting how the packet should be passed through the network, a value of 0 is normal)
- The packet's "IP ID" number was 41786 (again, don't worry about IP ID, it's used to reassemble fragmented packets)
- The packet was not fragmented

- The packet had a TimeToLive (TTL) of 64 (every hop across the Internet will subtract 1 from this number, when TTL=0, the packet is dropped)
- Though not available in some earlier Linux kernel versions, if there is an additional field after TTL, it reflects the IPCHAINS rule number which caused the packet to log

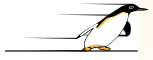
Putting it all together, we see that the OUTPUT rule for "DHCP Clients" was not enabled, resulting in a REJECTed packet. If you uncomment those two lines in the firewall ruleset and run the script again, DHCP will work fine.

Step 5.3.5. Running the firewall ruleset upon every reboot

To make the `/etc/rc.d/rc.firewall` ruleset run when Linux boots, edit `/etc/rc.d/init.d/network`. Look for the case statement. At the end of the "`start)`" case (immediately before the double semicolon "`;;`") insert the following lines:

```
#Load the IPCHAINS ruleset
/etc/rc.d/rc.firewall
```

This will invoke the firewall ruleset immediately after the network interfaces are initialized, minimizing the period of time the interfaces are vulnerable.



There are many security-related tools available for monitoring the computer from the inside (host-based) and the outside (network-based). In this step, we will not present exhaustive details of the installation and operation of each tool. We just want to make you aware of the most common tools available, their general purpose and operational parameters. You are encouraged to inspect the documentation and Web sites mentioned, and experiment with the tools to craft them into an integrated, personalized toolkit.

STEP 6.1 HOST-BASED MONITORING AND INTRUSION DETECTION

Host-based monitoring and intrusion detection (ID) tools generally concentrate on reading, processing and evaluating system log files. This implies that log files contain enough information to detect intrusion attempts in the first place. See Step 2.8.1 above for setting options in `/etc/syslogd.conf` and other logging options for maximizing the information available.

As with any ID system, the greatest problem is "false positives": results that look like intrusions, or at the very least trigger some action, when in fact the event is completely innocuous. All ID and monitoring tools require patience and persistence on the part of the administrator. As opposed to the best network access policy — default deny — the best intrusion detection policy is default allow. In other words, ignore only those events that you are sure are safe to ignore, and let the ID system report everything else. Time and experience will allow you to winnow out events that do not require your attention.

Step 6.1.1. Swatch, the Simple WATCHer

Swatch is a Perl program designed to read syslog files and generate events based on the content of the messages. Version 2.2 is supplied on the Red Hat Linux distribution media. Version 3.0 is in Beta testing at the time of this writing, available from `ftp://ftp.stanford.edu/general/security-tools/swatch/`. This version is a complete rewrite taking greater advantage of the features of Perl version 5, and with a completely revamped configuration file for scanning the syslog files.

The control file for swatch version 2.2 consists of a list of Perl regular expressions followed by "actions." See the manual page `swatch(5)` for a detailed description of the file format. Here is a very simple control file that will list only the lines in the syslog file that have the word "refused" in them:

```
/refused/       echo=bold
/.*/            ignore
```

To check `/var/log/messages` from the command line, do this:

```
[root]# swatch -c testrc -f /var/log/messages
*** swatch-2.2 (pid:32381) started at Fri Jul 25 15:12:08 EDT 1999
Jul 25 10:32:42 localhost sshd[16508]: refused connect from 192.168.1.1
Jul 25 10:33:31 localhost sshd[16530]: log: Rsa authentication refused
for root: bad modes for /root/.ssh/authorized_keys
Jul 25 20:11:22 localhost sshd[2312]: error: Fwd X11 connection from
127.0.0.1 refused by tcp_wrappers.
```

A good starting point for a control file under Red Hat Linux is `/usr/doc/swatch-2.2/config_files/swatchrc.personal`. It consists of a list of regular expressions that generate actions, followed by a list of regular expressions to ignore, and the final action is to simply echo everything else. This file can be tuned by adding actions and ignores and reiterating through `/var/log/messages` until you get results that are manageable.

Swatch can be configured to run through a single log file in a single pass, or it can be run as a daemon that continually monitors new log messages as they are generated. It can be configured with different control files to monitor mail logs, firewall logs, Samba logs, etc. See the manual page for `swatch(8)` for more information.

Step 6.1.2. Psionic Logcheck

Another of the many log file monitor programs is Logcheck from Psionic Software Systems. See Appendix for URLs to the Psionic Logcheck home page and other logfile monitors in the MetaLab Web site.

Logcheck comes with good documentation. Do a thorough read of the INSTALL text file before installing and running it. Installation is simple. After unpacking the compressed tar file, execute "`make linux`" in the source directory to copy the Linux-specific configuration files and script to `/usr/local/etc/`.

Logcheck generates reports that have three sections: active system attacks, security violations, and unusual system events. Active system attack notices are generated when a syslog entry matches any pattern in `logcheck.hacking`; security violations are generated from matches to patterns in `logcheck.violations`; and unusual events are generated by any line that does not match a pattern in `logcheck.ignore`. Read the comments in `logcheck.sh` for more detailed descriptions of these configuration files.

Like swatch, the administrator needs patience and persistence to weed out records that do not indicate genuine events and add those patterns to `logcheck.ignore`. At first, there are likely to be many false alarms. Investigate each reported instance to determine if, in fact, the reported event requires attention. If it does not and never will, a pattern to match that event can be put into `logcheck.ignore` and you will not see it reported again. It is best to make the pattern as specific as possible, so that unusual variations of the event are reported.

Step 6.1.3. Tripwire

Tripwire was originally developed by Gene Kim and Gene Spafford at the University of Purdue COAST Laboratory. The idea behind Tripwire is that in order to successfully compromise a system, some system files must be changed, added, or deleted in order to place back doors, trojans and exploit scripts. Tripwire generates a database of cryptographic signatures for important system binaries and configuration files and reports changes in any of these files over time.

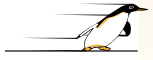
See appendix A for URLs for the source code for the Tripwire "Academic Source Release" version 1.3.1 and version 2.0 for Red Hat Linux. However it must be noted that Tripwire 2.0 is not supported for Red Hat 6.0, only for the earlier versions 5.2 and 5.1. As of the time of this writing, only the ASR 1.3.1 release works for Red Hat 6.0.

Read the documentation that comes with each release carefully for information about compiling, installing, and maintaining Tripwire.

Step 6.1.3.1. Tripwire databases

Before Tripwire, the only way to test the integrity of system files was by checking CRC checksums. Unfortunately, CRC 16- and 32-bit checksums are easily subverted. Tripwire builds a database of cryptographic checksums, such as MD5 and Snefru, that are nearly impossible to spoof and checks them regularly. If the checksum test fails, it is a very strong indication that the file in question has been modified.

The single greatest weakness in Tripwire was that, if an attacker can gain root access to the machine, he can modify or rebuild the Tripwire database to mask his intrusion. The only defense against this attack is to place the database on read-only media, such as a write-protected floppy or CD-R. One of the most significant innovations in Version 2 of Tripwire is cryptographic signing of the database and configuration files, which eliminates the need for placing Tripwire databases on read-only media. Hopefully, support for Red Hat 6.0 will be available by the time this guide goes to press.



▲ Step 6.1.3.2. Running Tripwire

The first time Tripwire is run, it builds a database of checksums and other information (UID, GID, mode, size, etc) for the files and directories listed in the configuration file. Then, each time it runs, it compares the information in the database to the current state of the system. When run interactively from the command line, it gives you the option of updating the database for each changed file it finds. So, if `/etc/group` has changed, and you were the one that changed it, you can update the database so that you will not be notified until the next time it changes (perhaps by an attacker).

▲ Step 6.1.3.3. Use rpm to verify package files

The database built by the Red Hat Package Manager (`rpm`) when it installs packages includes MD5 checksums, UID, GID, mode, size, etc. You can use `rpm` to verify the contents of the database against the current state of the files on disk and report any files that are different. This is a quick check for files that have changed.The disadvantage of this method, as opposed to the Tripwire method, is that with rpm verification you can not test changes over time, only since the files were installed. There is no provision for updating database entries that have changed with the knowledge and approval of the administrator.

To verify all package files, run:

```
[root]# rpm -Va
S.5....T c /etc/exports
S.5....T c /etc/hosts.allow
S.5....T c /etc/hosts.deny
.M...UG.   /mnt/cdrom
```

The entries for the files in `/etc` tell you that the size, MD5 checksum, and modification time have changed; for `/mnt/cdrom` the mode, user and group have changed.

Step 6.1.4. Psionic PortSentry

There are several tools for detecting port scans, Psionic PortSentry adds a twist: when a scan is detected it can take immediate action to block the scanning host from any further access to the machine. It works well in conjunction with Psionic Logcheck (Step 6.1.2). See Appendix A for a URL to the home page.

PortSentry comes with good documentation. Before compiling and installing, read `README.install`, `README.methods`, and `README.qa` thoroughly. PortSentry compiled "out-of-the-box" for the authors, without any changes necessary. The program and configuration files are installed in `/usr/local/psionic/portsentry` by default.

PortSentry can detect not only simple TCP connect-type scans, but many types of stealth scans, too, such as SYN/half-open, FIN, and NULL scans.

Before running PortSentry, review `portsentry.conf`. All the entries are documented in the configuration file, and in `README.install`. By default, the only action taken against scanning hosts is to place:

```
ALL: <attacker-IP>
```

at the end of `/etc/hosts.deny`. Since you should already have "`ALL: ALL`" at the end of `/etc/hosts.deny`, this is ineffective. In fact, this method is completely ineffective if any services have been allowed to the attacking host in `/etc/hosts.allow`, since TCP wrappers read `/etc/hosts.allow` for a match before reading `/etc/hosts.deny`.

On the other hand, using the `KILL_ROUTE` option in `portsentry.conf` is highly effective if configured correctly. There are two basic approaches to the `KILL_ROUTE` option: 1) add an entry in the routing table to send responses to the attacking host to a fictitious destination; and 2) add firewall packet filter rules with `ipchains/ipfwadm` to drop all packets from the attacking host. Adding a bogus route to the routing table may still allow some UDP-based attacks to get through (see `README.install` for more information). The preferred method is packet filtering with `ipchains`:

```
KILL_ROUTE="/sbin/ipchains -I input -s $TARGET$ -j DENY -l"
```

Note that adding automatic packet filtering leaves open the possibility of a denial-of-service attack, where the attacker spoofs a scan from another host in order to block that host from legitimate access.

Step 6.2.1. Tiger, the Texas A&M system checker

Tiger is a tool that inspects security-related settings of UNIX computers. Tiger was just recently upgraded (for the first time since 1994) to support checking of Red Hat Linux systems. See Appendix A for a URL to the Tiger home page and the source code.

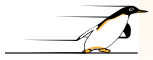
Step 6.2.2. Install and configure Tiger

Installing is simple: unpack the compressed tar file and execute "`make install`". This places the configuration and script files in `/usr/local/tiger`. The `Makefile` does not create the intermediate directories Tiger relies on though, so after installation, execute the following:

```
[root]# mkdir -m 700 -p /var/spool/tiger/bin
[root]# mkdir -m 700 -p /var/spool/tiger/logs
[root]# mkdir -m 700 -p /var/spool/tiger/work
```

Also, if you are running Bash version 2 as the default shell, you will need to edit `/usr/local/tiger/systems/Linux/2/config` and comment out the line that starts with "`GROUPS=`". In Bash 2, `GROUPS` is a read-only variable, and the script will abort if it tries to assign to it.

The checks that Tiger performs are set out in `/usr/local/tiger/tigerrc`. There are several sample `tigerrc` files in the source directory, and they are well commented. The default `tigerrc` pretty much tries to test everything it can. For Red Hat Linux version 6, most checks will work as advertised, except signature checking. You can turn this off by editing `tigerrc` and changing `Tiger_Check_SIGNATURES=Y to Tiger_Check_SIGNATURES=N`. By the same token, you can turn off running Crack on the password file by editing the entry for `Tiger_Run_CRACK`.



Step 6.2.3. Running Tiger

To run Tiger:

```
[root]# cd /usr/local/tiger
[root]# ./tiger
```

A number of messages are printed on standard output while Tiger runs through its checks. When the run is complete, the results are stored in `/var/spool/tiger/logs/security.report.<host>.<domain>.<date>-<time>`. Be prepared to have a lot of messages in the first security report. Reported problems look like:

```
—WARN—  [acc001w] Login ID postgres is disabled, but still has a valid shell
        (/bin/bash).

—FAIL—  [perm007f] /etc/aliases should not have group read.
—FAIL—  [perm007f] /etc/aliases should not have world read.
```

A WARN entry is something Tiger considers undesirable, but not terrible. A FAIL entry is something Tiger thinks you should change. The string in square brackets indicates how to get more information about the specific warning with the `tigexp` program:

```
[root]# ./tigexp acc001w

The listed login ID is disabled in some manner ('*' in passwd field, etc),
but the login shell for the login ID is a valid shell (from /etc/shells or
the system equivalent).  A valid shell can potentially enable the login ID
to continue to be used.  The login shell should be changed to something
that doesn't exist, or to something like /bin/false.
```

Step 6.2.4. Changing Tiger checks

Going through the security report, you will probably find some warnings and failures that you think are perfectly fine. The base information Tiger uses for comparison can be found in `/usr/local/tiger/systems/Linux/2`. For example, the file `file_access_list` contains the suggested ownership and mode bits for a number of system files. If you prefer that `/etc/aliases` is allowed group- and world-read permissions, you can edit this file to set the proper mode bits, and you will no longer get a FAIL message for those permissions.

Once again, patience and persistence are needed here. Follow the suggested changes in the security report and judiciously modify the base information until Tiger's security reports return the most meaningful information for your site.

Step 6.2.5 TARA, an updated version of Tiger

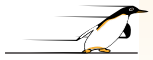
Because Tiger has only recently seen any new development, Advanced Research Corporation has taken the base code and enhanced it to produce the Tiger Analytical Research Assistant (TARA). See Appendix A for a URL to the TARA home page.

Installation, configuration, and operation of TARA is almost exactly the same as the original Tiger, even down to the same directory names. Note that the default `tigerrc` file enables far fewer system checks than Tiger. Review this file before running TARA and enable the checks that you think are relevant.

STEP 6.3 NETWORK-BASED VULNERABILITY ANALYSIS: LOOKING FROM THE OUTSIDE IN

There are many, many tools designed to scan hosts on a network for open ports, general security vulnerabilities, or specific exploitable services. Before any network scanning is done, the most important thing to secure is your own backside... get authorization to perform the scanning, in writing if possible. Most government and commercial organizations have specific policies for who can perform scanning, the scope of the scanning, and what can and can not be scanned for. Unauthorized violation of these policies can lead to sanctions, unemployment, and even legal action.

After you get authorization, post notice that a scan is going to be done and when. Some administrators are of the opinion that letting others know about an impending scan gives users an opportunity to plug holes beforehand, skewing the results of the scan. Well, at least the results are skewed in the right direction, and you will avoid problems with local users' intrusion detection programs unexpectedly raising red flags.



Before the scan, test the scanning software on an isolated network segment to make sure that the scanning tool does not have an adverse effect on the machines it is scanning. Some scanners are very efficient at crashing computers.

After the scan, disseminate information about the vulnerabilities that were found on a need-to-know basis. Results of vulnerability scans should be treated like any other sensitive information in your organization. Don't post the results on the company Intranet Web server. Summarize the results and discuss them with each responsible party privately.

Step 6.3.1. SATAN derivatives: SARA and SAINT

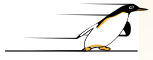
The Security Administrator Tool for Analyzing Networks (SATAN) caused quite a stir when it was released in 1994 by Dan Farmer and Wietse Venema. It had an innovative design for the user interface, using HTML and Web technology to effect a simple point-and-click process for scanning from 1 to thousands of hosts at one time.

The original release scanned for well-known vulnerabilities and exploits. The design, however, allowed for enhancing it with other checks and tests rather easily, and the authors assumed that it would be developed quickly. Two updated versions of SATAN are currently available: the Security Auditor's Research Assistant (SARA); and the Security Administrator's Integrated Network Tool (SAINT).

Both versions are under active development, in fact one of the original authors of SAINT is the currently the developer of SARA, and the systems track each other. Be sure to get the latest version of the program, earlier versions are known to have problems compiling under Red Hat 6.0. To compile and run either program, unpack the compressed tar file and execute "`perl reconfig`" to set up paths to the Perl interpreter and Web browser, and "`make linux`" to compile the scanner and its support programs.

When you invoke the scanner (`sara` or `saint`), it will invoke your Web browser and load the main page. Scanning proceeds from Data Management, to select the database to store results in, to Target Selection, to pick the host(s) to scan, to Report and Analysis. When the scan is over, results can be displayed by Danger Level, Type of Vulnerability, or Vulnerability Count.

There is much more to both systems, of course. Consult the documentation for more information.



Step 6.3.2 Nessus

Nessus (`http://www.nessus.org`) is an open-source security scanner that uses a client/server architecture. The Nessus daemon, `nessusd`, runs the scans and checks for vulnerabilities under the control of the client, which is written with a Gtk GUI interface. Nessus has a library of "plugins" that perform vulnerability tests. As of this writing there were 208 plugins, testing such things as IMAP and rpc.mountd overflows, CGI script bugs, RPC services, WinGate proxy holes, and a series of denial-of-service attacks, just to name a few.

Compiling and installing should be as simple as:

```
[root]# ./configure; make; make install
```

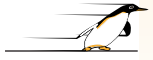
The `nessusd` server must be run first. The first time `nessusd` runs it computes a host key and builds a default users file `/usr/local/nessus/nessus.users`. This file controls who may connect to the server. To add an entry for root, run `nessusd` again as:

```
[root]# nessusd -P root,pass
```

where `pass` is an initial password. Starting the client, `nessus`, you first must connect to the server. Enter "`root`" in the "`Login:`" window and click "`Log in now!`". You will be prompted for the password entered above only the first time you log in, after that a private key is generated and used for all subsequent logins.

Note that Nessus tests for denial of service exploits by default, and our tests show that you are very likely to crash just about every machine you scan if this option is enabled. Before performing a scan, go to the Plugins tab and disable Denial of Service.

After a scan, Nessus pops up a report window with information for each scanned host. Security problems are highlighted with a red bullet, areas of concern are labeled with a black bullet. In both cases, a brief description is printed. Reports can be saved to disk for later study and comparison.

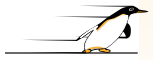


Step 6.3.3 Nmap port scanner

Nmap scans hosts for open ports. Unlike SARA/SAINT and Nessus, it does not test for specific vulnerabilities. Nmap is freely available from http://www.insecure.org/nmap/. There are a lot of scanning options: TCP connect() scanning, SYN half-open, FIN, NULL, UDP, ICMP, and SYN/FIN with IP fragments to bypass packet filters. It can try to identify the OS type of the remote host by using TCP/IP stack fingerprinting.

Compiling and installing is as easy as running make in the source directory. To scan localhost for all open TCP and UDP ports, for example to double-check your work from Steps 3.1, 3.2, and 3.3, execute:

```
[root]# ./nmap -sT -sU 127.0.0.1
WARNING:  -sU is now UDP scan — for TCP FIN scan use -sF

Starting nmap V. 2.2-BETA4 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on localhost (127.0.0.1):
Port    State   Protocol   Service
21      open    tcp        ftp
22      open    tcp        ssh
23      open    tcp        telnet
25      open    tcp        smtp
80      open    tcp        http
98      open    tcp        linuxconf
111     open    udp        sunrpc
111     open    tcp        sunrpc
113     open    tcp        auth
608     open    udp        sift-uft
610     open    tcp        npmp-local
618     open    udp        unknown
627     open    udp        unknown
629     open    tcp        unknown
632     open    udp        unknown
634     open    tcp        ginad
637     open    udp        unknown
639     open    tcp        unknown
1024    open    udp        unknown
1024    open    tcp        unknown
1026    open    udp        unknown
2049    open    udp        nfs

Nmap run completed — 1 IP address (1 host up) scanned in 5 seconds
```



Of course, if your host has these many ports open, you still have some work to do. To try to guess the operating system of the host at 192.168.1.1, execute:

```
[root]# ./nmap -O 192.168.1.1

Starting nmap V. 2.2-BETA4 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on garth (192.168.1.1):

Port    State   Protocol   Service
21      open    tcp        ftp
23      open    tcp        telnet
25      open    tcp        smtp
111     open    tcp        sunrpc
113     open    tcp        auth
515     open    tcp        printer
6000    open    tcp        X11

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=6118242 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.1.132; 2.2.0-pre1 - 2.2.2

Nmap run completed — 1 IP address (1 host up) scanned in 2 seconds
```

Nmap version 2.2-BETA4 (the latest version as of this writing) comes with a Gtk GUI front-end called `nmapfe`. The GUI makes it easy to select the host(s) or network(s) to scan and the options for scanning. The output from the nmap run is printed in its own window and can be stored in a log file for later comparison and study.

Step 6.3.4 Commercial products

Internet Security Systems (`http://www.iss.net`) markets commercial tools for host-based security assessment (System Scanner™), host- and network-based intrusion detection (RealSecure™), and network vulnerability scanning (Internet Scanner™). ISS products are mature professional tools. Evaluation copies of System Scanner and Internet Scanner are available free for download. The evaluation copies work only on the localhost.

Network Associates, Inc. (`http://www.nai.com`) markets CyberCop Scanner, which performs network vulnerability scanning for hosts, firewalls, hubs and switches. CyberCop is extensible through the Custom Audit Scripting Language (CASL). It has an automatic update feature to keep the software and vulnerability database up to date.

APPENDIX A
Resources and References

The contents of all the Appendices in this guide are available on the World Wide web at http://www.sans.org/linux.htm.

Step 2:

Installation HOWTO:	http://metalab.unc.edu/pub/Linux/docs/HOWTO/Installation-HOWTO
TrinityOS	http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html
Secure-syslog	http://www.core-sdi.com/ssyslog/
Syslog-ng	http://www.balabit.hu/products/syslog-ng.html
Nsyslogd	http://coombs.anu.edu.au/~avalon/nsyslog.html

Step 3:

CIAC login banners information bulletin	http://ciac.llnl.gov/ciac/bulletins/j-043.shtml
Linux Journal article on TCP Wrappers	http://linuxjournal.com:8080/lj-issues/issue40/2180.html
Linux software archives:	
Metalab:	http://metalab.unc.edu/pub/Linux/
Freshmeat:	http://www.freshmeat.net/
AutoRPM home page	http://www.kaybee.org/~kirk/html/linux.html
Linux Security HOWTO	http://metalab.unc.edu/pub/Linux/docs/HOWTO/Security-HOWTO.html
Linux Administrator's Security Guide	http://www.securityportal.com/lasg/
RedHat Security Information	http://www.redhat.com/corp/support/docs/errata.html (Select General Errata for your version, look for "security" in the Web page)

Subscribe to Red Hat mailing list for security and update announcements:

```
$ Mail -s subscribe redhat-announce-list@redhat.com </dev/null
```

Caldera Systems Security Advisories	http://www.calderasystems.com/news/security/index.html
Caldera-announce mailing list	http://www.calderasystems.com/support/forums/caldera-announce.html
SUSE Electronic mail security alerts	http://www.suse.com/MailingLists/index.html
Debian Security Information	http://www.debian.org/security/

Debian Mailing list for security and update announcements:

```
$ Mail -s subscribe debian-security-announce-request@lists.debian.org </dev/null
```

Step 4:

SSH	ftp://ftp.cs.hut.fi/pub/ssh
BIND	http://www.isc.org/view.cgi?/products/BIND/index.phtml
named chroot jail	http://www.linux.com/security/articles/july/sec_bind.phtml
Sendmail	http://www.sendmail.org/current-release.html
Qmail	http://www.qmail.org
Postfix	http://www.porcupine.org
Eudora Qpopper	http://www.eudora.com/free/servers.html
Cyrus IMAPD	http://asg.web.cmu.edu/cyrus/imapd/
sendmail relaying	http://www.sendmail.org/tips/relaying.html
`stunnel`	http://mike.daewoo.com.pl/computer/stunnel/
`sslwrap`	http://www.rickk.com/sslwrap/
LPRng	http://www.astart.com/LPRng.html
Samba	http://www.samba.org
Apache	http://www.apache.org
Commercial secure web servers:	
Covalent Raven SSL Module:	http://www.covalent.net/raven/ssl/
Red Hat Linux E-Commerce Server:	http://store.redhat.com/commerce/
Roxen:	http://www.roxen.com/
Stronghold:	http://www.c2.net/products/sh2/index.php3
Zeus:	http://www.zeustech.net/

Step 5:

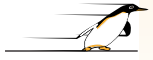
Linux kernels	http://www.kernel.org http://www.kernelnotes.org
IP Masquerade HOWTO	http://ipmasq.cjb.net
IP CHAINS HOWTO	http://www.rustcorp.com/linux/ipchains/

Step 6:

Psionic Logcheck	http://www.psionic.com/abacus/logcheck/
Other syslog filters	http://metalab.unc.edu/pub/Linux/system/admin/log/
Tripwire Security home page	http://www.tripwiresecurity.com
Tripwire ASR 1.3.1	http://www.tripwiresecurity.com/products/ASR1_3.html
Tripwire 2.0	http://www.tripwiresecurity.com/products/2_0Linux.html
TARA home page	http://www.arc.com/tara/
Psionic PortSentry	http://www.psionic.com/abacus/portsentry/
Tiger home page	http://www.net.tamu.edu/network/tools/tiger.html
Tiger source code	http://www.net.tamu.edu/ftp/security/TAMU/
SAINT home page	http://www.wwdsi.com/saint/
SARA home page	http://www.arc.com/sara/
Nmap	http://www.insecure.org/nmap/

Books and articles for installation, administration, advanced security, etc.

"Building Internet Firewalls," Chapman & Zwicky, 1995, O'Reilly & Associates, ISBN 1-56592-124-0
"DNS and BIND," 3rd Edition, Ablitz and Liu, O'Reilly and Associates, 1998
"sendmail," Costales and Allman, O'Reilly & Associates, 1997
"Samba: Integrating UNIX and Windows," John D. Blair, SSC, Inc. ISBN 1-57831-006-7



APPENDIX B
Stock Red Hat 6.0
/etc/inetd.conf

The following is the default `/etc/inetd.conf` file installed with Red Hat 6.0. Any line not preceded by a comment character, "#", is available — and vulnerable — to the Internet. See Step 3.1 for more information.

```
#
# inetd.conf                    This file describes the services that will be available
#                               through the INETD TCP/IP super server.  To re-configure
#                               the running INETD process, edit this file, then send the
#                               INETD process a SIGHUP signal.
#
# Version:      @(#)/etc/inetd.conf   3.10 05/27/93
#
# Authors:      Original taken from BSD UNIX 4.3/TAHOE.
#                               Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
# Modified for Debian Linux by Ian A. Murdock <imurdock@shell.portal.com>
#
# Modified for RHS Linux by Marc Ewing <marc@redhat.com>
#
# Further modified by Olaf Kirch <okir@caldera.com> for Caldera Open Linux
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
# To re-read this file after changes, just do a 'killall -HUP inetd'
#
# Note: builtin UDP services now silently drop packets from ports < 512.
#echo            stream      tcp       nowait      root        internal
#echo            dgram       udp       wait        root        internal
#discard         stream      tcp       nowait      root        internal
#discard dgram   udp         wait      root        internal
#daytime         stream      tcp       nowait      root        internal
#daytime dgram   udp         wait      root        internal
#chargen         stream      tcp       nowait      root        internal
#chargen dgram   udp         wait      root        internal
#time            stream      tcp       nowait      root        internal
#time            dgram       udp       wait        root        internal
#
# These are standard services.
#
ftp             stream      tcp       nowait      root        /usr/sbin/tcpd in.ftpd -l -a
telnet          stream      tcp       nowait   root           /usr/sbin/tcpd   in.telnetd
#gopher         stream      tcp       nowait   root           /usr/sbin/tcpd   gn
```

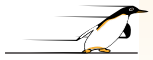
APPENDIX B
Stock Red Hat 6.0
/etc/inetd.conf

```
# do not uncomment smtp unless you *really* know what you are doing.
# smtp is handled by the sendmail daemon now, not smtpd.  It does NOT
# run from here, it is started at boot time from /etc/rc.d/rc#.d.
#smtp           stream     tcp       nowait      root        /usr/bin/smtpd   smtpd
#nntp           stream     tcp       nowait      root        /usr/sbin/tcpd   in.nntpd
#
# Shell, login, exec and talk are BSD protocols.
#
shell           stream     tcp       nowait      root        /usr/sbin/tcpd   in.rshd
login           stream     tcp       nowait      root        /usr/sbin/tcpd   in.rlogind
#exec           stream     tcp       nowait      root        /usr/sbin/tcpd   in.rexecd
#talk           dgram      udp       wait        nobody.tty  /usr/sbin/tcpd   in.talkd
#ntalk          dgram      udp       wait        nobody.tty  /usr/sbin/tcpd   in.ntalkd
#dtalk          stream     tcp       wait        nobody.tty  /usr/sbin/tcpd   in.dtalkd
#
# Pop and imap mail services et al
#
#pop2           stream     tcp       nowait      root        /usr/sbin/tcpd   ipop2d
#pop3           stream     tcp       nowait      root        /usr/sbin/tcpd   ipop3d
#imap           stream     tcp       nowait      root        /usr/sbin/tcpd   imapd
#
# The Internet UUCP service.
#
#uucp           stream     tcp       nowait      uucp        /usr/sbin/tcpd   /usr/sbin/uucico-
l
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#
#tftp           dgram      udp       wait        root        /usr/sbin/tcpd   in.tftpd
#bootps         dgram      udp       wait        root        /usr/sbin/tcpd   bootpd
#
# cfinger is for GNU finger, which is currently not in use in RHS Linux
#
finger          stream     tcp       nowait      nobody      /usr/sbin/tcpd   in.fingerd -u
#cfinger        stream     tcp       nowait      root        /usr/sbin/tcpd   in.cfingerd
#
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
#
#systat         stream     tcp       nowait      nobody      /usr/sbin/tcpd    /bin/ps -auwwx
#netstat        stream     tcp       nowait      nobody      /usr/sbin/tcpd    /bin/netstat —inet
#
# Authentication
#
#auth           stream     tcp       nowait      root        /usr/sbin/tcpd in.identd -i -t30
auth            stream     tcp       nowait      nobody      /usr/local/sbin/oidentd -i
swat            stream     tcp       nowait.     400 root    /usr/sbin/tcpd swat
#
# End of inetd.conf
```

APPENDIX C
Red Hat SYSH init script for the SSH daemon

The following SYSV init script follows the general conventions for Red Hat 6.0 init scripts as outlined in the online document `/usr/doc/initscripts-4.16/sysvinitfiles`.

```
##!/bin/bash
#
#               /etc/rc.d/init.d/sshd
#
# sshd                          Start the Secure Shell daemon
#
# chkconfig: 345 50 50
# description:  The Secure Shell daemon, sshd, allows for strong \
#               authentication of, and encrypted communications with, \
#               remote clients using the Secure Shell Protocol
# processname:  sshd
# pidfile:      /var/run/sshd.pid
# config:       /etc/sshd_config

# Source function library.
. /etc/rc.d/init.d/functions

SSHD=/usr/local/sbin/sshd
SSHD_CONFIG=/etc/sshd_config

case "$1" in
   start)
               echo -n "Starting SSH services: "
               if [ -x $SSHD -a -f $SSHD_CONFIG ]
               then
                                    daemon $SSHD
               else
                                    echo_failure
               fi
               echo
               touch /var/lock/subsys/sshd
               ;;
   stop)
               echo -n "Shutting down SSH services: "
               killproc sshd
               echo
               rm -f /var/lock/subsys/sshd
               ;;
   status)
               status sshd
               ;;
   restart)
               $0 stop; $0 start
               ;;
   reload)
               killall -HUP sshd
               ;;
   *)
               echo "Usage: sshd {start|stop|status|reload|restart}"
               exit 1
               ;;
esac
```

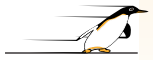


APPENDIX D
A strong packet firewall ruleset

```
#!/bin/sh
#------------------------------------
# Author: David A. Ranch
# Based on the TrinityOS IPCHAINS firewall v3.12
# http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html
#------------------------------------
# This configuration assumes the following (DSL / Cablemodem setup):
#
#         1)  The external interface is running on "eth0"
#         2)  The external IP address is dynamically assigned
#         3)  The internal IP Masqueraded network interface is "eth1"
#         4)  The internal network is addressed within the private
#             192.168.1.x TCP/IP addressing scheme per RFC1918
#
#    ****
#    NOTE:    All 2.2.x Linux kernels prior to 2.2.11 have a fragmentation
#    ****     bug that renders all strong IPCHAINS rulesets void.  It
#             is CRITICAL that users upgrade the Linux kernel to 2.2.11+
#             for proper firewall security.
#
#------------------------------------

#**********************************************************************
# Initializing
#**********************************************************************
echo -e "\n\nLoading IPCHAINS Firewall Version 3.12"
echo "------------------------------------"

#------------------------------------
# Variables
#------------------------------------

# The loopback interface and address
LOOPBACKIF="lo"
LOOPBACKIP="127.0.0.1"

# External interface device.
#
# NOTE:  PPP and SLIP users will want to replace this interface
#        with the correct modem interface such as "ppp0" or "sl0"
#
EXTIF="eth0"
echo External Interface: $EXTIF
```

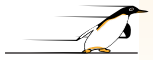
```
# IP address of the external interface
#
# NOTE: Red Hat users of DHCP to get TCP/IP addresses (Cablemodems, DSL, etc)
#       will need to install and use a different DHCP client than the stock
#       client called "pump".  One recommended DHCP client is called "dhcpcd"
#       and can found in Appendix A.
#
#       The stock Red Hat DHCP client doesn't allow the ability to have scripts
#       run when DHCP gets a TCP/IP address.  Specifically, DHCP delves out
#       TCP/IP addresses to its clients for a limited amount of time; this
#       called a "lease".  When a DHCP lease expires, the client will query the
#       DHCP server for a lease renewal.  Though the DHCP client will usually
#       get back its original TCP/IP address, this is NOT always guaranteed.
#       With this understood, if you receive a different TCP/IP address than
#       the IPCHAINS firewall was configured for, the firewall will block ALL
#       network access in and out of the Linux server because that was what it
#       was configured to do.
#
#       As mentioned above, the key to solve this problem is to use a DHCP
#       client program that can re-run the /etc/rc.d/rc.firewall ruleset once a
#       new TCP/IP address is set.  The new ruleset will make the required
#       changes to the rulesets to allow network traffic from and to your new
#       TCP/IP address.
#
#       With the dhcpcd program, it will need to executed with the following
#       command line option to have the firewall ruleset re-run upon every DHCP
#       lease renew:
#
#                                -c /etc/rc.d/rc.firewall
#
# Static TCP/IP addressed users: For EXTIP, EXTBROAD, and EXTGW, simply replace
# the pipelines with your correct TCP/IP address, broadcast address, and
# external gateway, respectively.
#
# eg:   EXTIP="172.22.43.23"
#
EXTIP=`/sbin/ifconfig | grep -A 4 $EXTIF | awk '/inet/ { print $2 } ' | sed -e s/addr://`
echo External IP: $EXTIP
```

```
# Broadcast address of the external network
EXTBROAD=`/sbin/ifconfig | grep -A 1 $EXTIF | awk '/Bcast/ { print $3 }' | sed -e s/Bcast://`
echo External broadcast: $EXTBROAD

# Gateway for the external network
EXTGW=`/sbin/route -n | grep -A 4 UG | awk '{ print $2}'`
echo Default GW: $EXTGW
echo " -- "

# Internal interface device.
INTIF="eth1"

# IP address on the internal interface
INTIP="192.168.1.1"
echo Internal IP: $INTIP

# IP network address of the internal network
INTLAN="192.168.1.0/24"

# IP Port Forwarded Addresses
#
# IP address of an internal host that should have external traffic forwarded to
# Port forwarding allows external traffic to directly connect to an INTERNAL
# Masq'ed machine. An example need for port forwarding is the need for external
# users to directly contact a WWW server behind the MASQ server.
#
# NOTE:  Port forwarding is well beyond the scope of this documentation to
#        explain the security issues implied in opening up access like this.
#        Please see Appendix A to find the IP-MASQ-HOWTO for a full explanation.
#
# Disabled by default.
#PORTFWIP="192.168.1.20"

# IP Mask for all IP addresses
UNIVERSE="0.0.0.0/0"

# IP Mask for broadcast transmissions
BROADCAST="255.255.255.255"

# Specification of the high unprivileged IP ports.
UNPRIVPORTS="1024:65535"

# Specification of X Window System (TCP) ports.
```

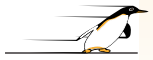


```
XWINDOWS_PORTS="6000:6010"

# The TCP/IP address of your slave DNS servers (if any).
# This is OPTIONAL!
#
# Disabled by default.
#SECONDARYDNS="172.31.44.22"

# The TCP/IP addresses of a specifically allowed EXTERNAL hosts
#
# Disabled by default.
#SECUREHOST="172.20.2.11"
#SECUREHOST2="172.20.2.12"

# TCP/IP addresses of INTENRAL hosts network allowed to directly
#         connect to the Linux server.  All internal hosts are allowed
#         per default.
#
# Disabled by default
#HOST1IP="192.168.1.10"
#HOST2IP="192.168.1.11"

# Logging state.
#
# Uncomment the " " line and comment the "-l" line if you want to
# disable logging of some of more important the IPCHAINS rulesets.  #
# The output of this logging can be found in the /var/log/messages
# file.  It is recommended that you leave this setting enabled.
# If you need to reduce some of the logging, edit the rulesets and
# delete the "$LOGGING" syntax from the ruleset that you aren't
# interested in.
#
# LOGGING=" "
LOGGING="-l"

echo "————————————————————————————————————"
```

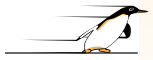
```
#--------------------------------------
# Debugging Section
#--------------------------------------
# If you are having problems with the firewall, uncomment the lines
# below and then re-run the firewall to make sure that the firewall
# is not giving any errors, etc.  The output of this debugging
# script will be in a file called /tmp/rc.firewall.dump
#--------------------------------------
#
#echo "  - Debugging."
#echo Loopback IP: $LOOPBACKIP > /tmp/rc.firewall.dump
#echo Loopback interface name: $LOOPBACKIF >> /tmp/rc.firewall.dump
#echo Internal interface name: $INTIF >> /tmp/rc.firewall.dump
#echo Internal interface IP: $INTIP >> /tmp/rc.firewall.dump
#echo Internal LAN address: $INTLAN >> /tmp/rc.firewall.dump
#echo ------------------------------- >> /tmp/rc.firewall.dump
#echo External interface name: $EXTIF >> /tmp/rc.firewall.dump
#echo External interface IP: $EXTIP >> /tmp/rc.firewall.dump
#echo External interface broadcast IP: $EXTBROAD >> /tmp/rc.firewall.dump
#echo External interface default gateway: $EXTGW >> /tmp/rc.firewall.dump
#echo ------------------------------- >> /tmp/rc.firewall.dump
#echo External secondary DNS: $SECONDARYDNS >> /tmp/rc.firewall.dump
#echo External secured host: $SECUREHOST >> /tmp/rc.firewall.dump

#--------------------------------------
# General
#--------------------------------------
# Performs general processing such as setting the multicast route
# and DHCP address hacking.
#
# Multicast is a powerful, yet seldom used aspect of TCP/IP for multimedia
# data. Though it isn't used much now (because most ISPs don't enable multicast
# on their networks, it will be very common in a few more years. Check out
# www.mbone.com for more detail.
#
# Adding this feature is OPTIONAL.
#
# Disabled by default.
#echo "  - Adding multicast route."
#/sbin/route add -net 224.0.0.0 netmask 240.0.0.0 dev $EXTIF
```



```
# Turn on IP Forwarding in the Linux kernel
#
# There are TWO methods of turning on this feature.  The first method is the
# Red Hat way. Edit the /etc/sysconfig/network file and change the
# "FORWARD_IPV4" line to say:
#
#         FORWARD_IPV4=true
#
# The second method is shown below and can executed at any time while the
# system is running.
#
echo "   - Enabling IP forwarding."
echo "1" > /proc/sys/net/ipv4/ip_forward

# Disable IP spoofing attacks.
#
# This drops traffic addressed for one network though it is being received on a
# different interface.
#
echo "   - Disabling IP Spoofing attacks."
for file in /proc/sys/net/ipv4/conf/*/rp_filter
do
 echo "1" > $file
done

# Comment the following out of you are not using a dynamic address
echo "   - Enabling dynamic TCP/IP address hacking."
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#-------------------------------------
# Masquerading Timeouts
#-------------------------------------
# Set timeout values for masq sessions (seconds).
#
# Item #1 - 2 hrs timeout for TCP session timeouts
# Item #2 - 10 sec timeout for traffic after the TCP/IP "FIN" packet is received
# Item #3 - 60 sec timeout for UDP traffic
#
echo "   - Changing IP masquerading timeouts."
/sbin/ipchains -M -S 7200 10 60
```

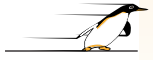


```
#————————————————————————————
# Masq Modules
#————————————————————————————
# Most TCP/IP-enabled applications work fine behind a Linux IP
# Masquerade server.  But, some applications need a special
# module to get their traffic in and out properly.
#
# Note:   Some applications do NOT work though IP Masquerade server at ALL such
#         as any H.323-based program.  Please the IP-MASQ HOWTO for more details.
#
# Note #2:Only uncomment the modules that you REQUIRE to be loaded.
#         The FTP module is loaded by default.
#————————————————————————————
echo "  - Loading masquerading modules."

#/sbin/modprobe ip_masq_cuseeme
/sbin/modprobe ip_masq_ftp
#/sbin/modprobe ip_masq_irc
#/sbin/modprobe ip_masq_quake
#/sbin/modprobe ip_masq_raudio
#/sbin/modprobe ip_masq_vdolive

#————————————————————————————
# Default Policies
#————————————————————————————
# Set all default policies to REJECT and flush all old rules.
#————————————————————————————
echo "  - Flushing all old rules and setting all default policies to REJECT "

# Flush all old rulesets
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward

# Change default policies to REJECT.
#
# We want to only EXPLICTLY allow what traffic is allowed IN and OUT of the
# firewall.  All other traffic will be implicitly blocked.
/sbin/ipchains -P input REJECT
/sbin/ipchains -P output REJECT
/sbin/ipchains -P forward REJECT
```

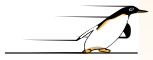


```
#*********************************************************************
# Input Rules
#*********************************************************************
echo "————————————————————————————————————"
echo "Input Rules:"

#————————————————————————————————
# Incoming Traffic on the Internal LAN
#————————————————————————————————
# This section controls the INPUT traffic allowed to flow within the internal
# LAN.  This means that all input traffic on the local network is valid.  If
# you want to change this default setting and only allow certain types of
# traffic within your internal network, you will need to comment this following
# line and configure individual ACCEPT lines for each TCP/IP address you want
# to let through.  A few example ACCEPT lines are provided below for
# demonstration purposes.
#
# Sometimes it is useful to allow TCP connections in one direction but not the
# other.  For example, you might want to allow connections to an external HTTP
# server but not connections from that server.  The naive approach would be to
# block TCP packets coming from the server. However, the better approach is to
# use the -y flag which will block only the packets used to request a
# connection.
#————————————————————————————————
echo "  - Setting input filters for traffic on the internal LAN."

# Local interface, local machines, going anywhere is valid.
#
# Comment this line out if you want to only allow specific traffic on the
# internal network.
/sbin/ipchains -A input -j ACCEPT -i $INTIF -s $INTLAN -d $UNIVERSE

# Loopback interface is valid.
/sbin/ipchains -A input -j ACCEPT -i $LOOPBACKIF -s $UNIVERSE -d $UNIVERSE

# DHCP Server.
#
# If you have configured a DHCP server on the Linux machine to serve IP
# addresses to the internal network, you will need to enable this section.
#
# This is an example of how to let input traffic flow through the local
# LAN if we have rejected all prior requests above.
#
# Disabled by default
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p udp -s $UNIVERSE bootpc -d $BROADCAST/0 bootps
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $UNIVERSE bootpc -d $BROADCAST/0 bootps
```

```
#--------------------------------------
# Explicit Access from Internal LAN Hosts
#--------------------------------------
# This section is provided as an example of how to allow only SPECIFIC hosts on
# the internal LAN to access services on the firewall server.  Many people
# might feel that this is extreme but many system attacks occur from the
# INTERNAL networks.
#
# Examples given allow access via FTP, FTP-DATA, SSH, and TELNET.
#
# In order for this ruleset to work, you must first comment out the line above
# that provides full access to the internal LAN by all internal hosts. You will
# then need to enable the lines below to allow any access at all.
#--------------------------------------
#echo "  - Setting input filters for specific internal hosts."

# First allowed internal host to connect directly to the Linux server
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp-data
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ssh
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP telnet

# Second allowed internal host to connect directly to the Linux server
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp-data
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ssh
#/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP telnet

#--------------------------------------
# Incoming Traffic from the External Interface
#--------------------------------------
# This ruleset will control specific traffic that is allowed in from
# the external interface.
#--------------------------------------
#
echo "  - Setting input filters for traffic from the external interface."
```



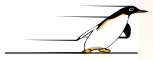
```
# Remote interface, claiming to be local machines, IP spoofing, get lost & log
/sbin/ipchains -A input -j REJECT -i $EXTIF -s $INTLAN -d $UNIVERSE $LOGGING

# DHCP Clients.
#
# If you get a dynamic IP address for your ADSL or Cablemodem connection, you
# will need to enable these lines.
#
# Enabled by default.
/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p udp -s $UNIVERSE bootps -d $BROADCAST/0  bootpc
/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE bootps -d $BROADCAST/0  bootpc

# FTP:  Allow external users to connect to the Linux server ITSELF for
#       PORT-style FTP services.  This will NOT work for PASV FTP transfers.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ftp-data

# HTTP: Allow external users to connect to the Linux server ITSELF for
#       HTTP services.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP http

# ICMP: Allow ICMP packets from all external TCP/IP addresses.
#
# NOTE: Disabling ICMP packets via the firewall ruleset can do far more than
# just stop people from pinging your machine.  Many aspects of TCP/IP and its
# associated applications rely on various ICMP messages.  Without ICMP, both
# your Linux server and internal Masq'ed computers might not work.
#
/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p icmp -s $UNIVERSE -d $EXTIP

# NFS: Reject NFS traffic FROM and TO external machines.
#
# NOTE: NFS is one of the biggest security issues an administrator will face.
# Do NOT enable NFS over the Internet or any non-trusted networks unless you
# know exactly what you are doing.
#
/sbin/ipchains -A input -j REJECT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP 2049
/sbin/ipchains -A input -j REJECT -i $EXTIF -p tcp -s $UNIVERSE 2049 -d $EXTIP
```



```
# NNTP: Allow external computers to connect to the Linux server ITSELF
#       for NNTP (news) services.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP nntp

# NTP:  Allow external computers to connect to the Linux server ITSELF for
#       NTP (time) updates
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ntp

# TELNET: Allow external computers to connect to the Linux server ITSELF for
#         TELNET access.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP telnet

# SSH server:  Allow external computers to connect to the Linux server ITSELF
#              for SSH access.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE -d $EXTIP ssh

#--------------------------------------
# Incoming Traffic on all Interfaces
#--------------------------------------
# This will control input traffic for all interfaces.  This is
# usually used for what could be considered as public services.
#--------------------------------------
echo "  - Setting input filters for public services (all interfaces)."

# AUTH: Allow the authentication protocol, ident, to function on all
#       interfaces but disable it in /etc/inetd.conf.  The reason to
#       allow this traffic in but block it via Inetd is because some
#       legacy TCP/IP stacks don't deal with REJECTed "auth" requests
#       properly.
#
/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $UNIVERSE auth

# BOOTP/DHCP: Reject all stray bootp traffic.
#
# Disabled by default.
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE bootpc
```

```
# DNS:  If you are running an authoritative DNS server, you must open
#       up the DNS ports on all interfaces to allow lookups.  If you are
#       running a caching DNS server, you will need to at least open the DNS
#       ports to internal interfaces.
#
#       It is recommend to secure DNS by restricting zone transfers and split
#       DNS servers as documented in Step 4.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $UNIVERSE domain
#/sbin/ipchains -A input -j ACCEPT -p udp -s $UNIVERSE -d $UNIVERSE domain

# RIP:  Reject all stray RIP traffic.  Many improperly configured
#       networks propagate network routing protocols to the edge of the
#       network.  The follow line will allow you explicitly filter it here
#       without logging to SYSLOG.
#
# Disabled by default.
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE route

# SAMBA:Reject all stray SAMBA traffic. Many networks propagate the
#       chatty SMB network protocols to the edge of the network.  The
#       following line will allow you explicitly filter it here without
#       logging to SYSLOG.
#
# Disabled by default.
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE netbios-ns
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE netbios-dgm
#/sbin/ipchains -A input -j REJECT -p udp -s $UNIVERSE -d $UNIVERSE netbios-ssn

# SMTP: If this server is an authoritative SMTP email server, you must
#       allow SMTP traffic to all interfaces.
#
# Disabled by default.
#/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $EXTIP smtp

#--------------------------------------
# Explicit INPUT Access from external LAN Hosts
#--------------------------------------
# This controls external access from specific external hosts (secure hosts).
# This example permits FTP, FTP-DATA, SSH, POP-3 and TELNET traffic from a
# secure host INTO the firewall. In addition to these input rules, we must also
# explicitly allow the traffic from the remote host to get out.  See the rules
# in the output section for more details
#
# Disabled as default.
#--------------------------------------
echo "  - Setting input filters for explicit external hosts."
```

```
# The secure host
#
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP ftp-data
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP ssh
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP pop-3
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST -d $EXTIP telnet
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST2 -d $EXTIP telnet
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST2 -d $EXTIP ftp
#/sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $SECUREHOST2 -d $EXTIP ftp-data

#--------------------------------------
# Port Forwarding
#--------------------------------------
# Port forwarding allows external traffic to directly connect to an INTERNAL
# Masq'ed machine. An example need for port forwarding is the need for external
# users to directly contact a WWW server behind the MASQ server.
#
# NOTE:  Port forwarding is well beyond the scope of this documentation to
#        explain the security issues implied in opening up access like this.
#        Please see Appendix A to read the IP-MASQ-HOWTO for a full explanation.
#
# Do not use ports greater than 1023 for redirection ports.
#
# Disabled by default.
#--------------------------------------
#echo "   * Enabling Port Forwarding onto internal hosts."
#/usr/sbin/ipmasqadm portfw -f
#echo "   * Forwarding SSH traffic on port 26 to $PORTFWIP"
#/usr/sbin/ipmasqadm portfw -a -P tcp -L $EXTIP 26 -R $PORTFWIP 22

# HIGH PORTS:
#
# Enable all high unprivileged ports for all reply TCP/UDP traffic
#
# NOTE:  The use of the "! -y" flag filters TCP traffic that doesn't have the
#        SYN bit set.  In other words, this means that any traffic that is
#        trying to initiate traffic to your server on a HIGH port will be
#        rejected.
#
#        The only HIGH port traffic that will be accepted is either return
#        traffic that the server originally initiated or UDP-based traffic.
#
# NOTE2: Please note that port 20 for ACTIVE FTP sessions should NOT use
#        SYN filtering.  Because of this, we must specifically allow it in.
#
echo "   - Enabling all input REPLY (TCP/UDP) traffic on high ports."
```

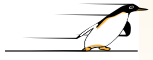


```
/sbin/ipchains -A input -j ACCEPT ! -y -p tcp -s $UNIVERSE -d $EXTIP $UNPRIVPORTS
/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE ftp-data -d $EXTIP $UNPRIVPORTS
/sbin/ipchains -A input -j ACCEPT -p udp -s $UNIVERSE -d $EXTIP $UNPRIVPORTS

#————————————————————————————
# Catch All INPUT Rule
#————————————————————————————
#
echo "  - Final input catch all rule."

# All other incoming is denied and logged.
/sbin/ipchains -A input -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING


#*********************************************************************
# Output Rules
#*********************************************************************
echo "————————————————————————————————"
echo "Output Rules:"

#————————————————————————————
# Outgoing Traffic on the Internal LAN
#————————————————————————————
# This ruleset provides policies for traffic that is going out on the internal
# LAN.
#
# In this example, all traffic is allowed out.  Therefore there is no
# requirement to implement individual filters.  However, as with the input
# section above, examples are given for demonstrative purposes.  It is also
# noted that the same rules, outlined above, apply regarding the order of the
# filtering rules.
#————————————————————————————
echo "  - Setting output filters for traffic on the internal LAN."

# Local interface, any source going to local net is valid.
/sbin/ipchains -A output -j ACCEPT -i $INTIF -s $UNIVERSE -d $INTLAN

# Loopback interface is valid.
/sbin/ipchains -A output -j ACCEPT -i $LOOPBACKIF -s $UNIVERSE -d $UNIVERSE
```

```
# DHCP: If you have configured a DHCP server on this Linux machine, you
#       will need to enable the following ruleset.
#
# Disabled by default.
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p udp -s $INTIP/32 bootps -d $BROADCAST/0 bootpc
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTIP/32 bootps -d $BROADCAST/0 bootpc

# HTTP: The following is an example of how to allow HTTP traffic to an
#       intranet WWW server without allowing access from the external
#       network.
#
# Disabled by default.
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTIP/32 http -d $INTLAN


#--------------------------------------
# Explicit Output from Internal LAN Hosts
#--------------------------------------
# The following rulesets only allow SPECIFIC hosts on the internal LAN to
# access services on this firewall server itself.  Many people might feel that
# this is extreme but many system attacks occur from the INTERNAL network as
# well.
#
# Examples given allow access via FTP, FTP-DATA, SSH, and TELNET.
#
# In order for this ruleset to work, you must first comment out the line above
# that provides full access to the internal LAN by all internal hosts.
#
# Disabled by default.
#--------------------------------------
#echo "  - Setting output filters for specific internal hosts."

# First host
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ftp-data
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP ssh
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST1IP -d $INTIP telnet

# Second host
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ftp-data
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP ssh
#/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $HOST2IP -d $INTIP telnet
```

```
#--------------------------------------
# Outgoing Traffic on the External Interface
#--------------------------------------
# This ruleset will control what traffic can go out on the external interface.
#--------------------------------------
echo "  - Setting input filters for traffic to the external interface."

# Reject outgoing traffic to the local net from the remote interface,
# stuffed routing; deny & log
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $UNIVERSE -d $INTLAN $LOGGING

# Reject outgoing traffic from the local net from the external interface,
# stuffed masquerading, deny and log
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $INTLAN -d $UNIVERSE $LOGGING

# DHCP Client: If your Linux server is connected via DSL or a Cablemodem
#              connection and you get dynamic DHCP addresses, you will need to
#              enable the following rulesets.
#
# Enabled by default.
/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE bootpc -d $UNIVERSE bootps
/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p udp -s $UNIVERSE bootpc -d $UNIVERSE bootps

# FTP: Allow FTP traffic (the Linux server is a FTP server)
#
# Disabled by default.
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp -d $UNIVERSE
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp-data -d $UNIVERSE

# HTTP: Allow HTTP traffic (the Linux server is a WWW server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP http -d $UNIVERSE

# NTP: Allow NTP updates (the Linux server is a NTP server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ntp -d $UNIVERSE

# TELNET: Allow telnet traffic (the Linux server is a TELNET server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP telnet -d $UNIVERSE

# SSH server: Allow outgoing SSH traffic (the Linux server is a SSH server)
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ssh -d $UNIVERSE
```

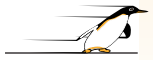
```
#--------------------------------------
# Outgoing Traffic on all Interfaces
#--------------------------------------
# This will control output traffic for all interfaces.  This is
# usually used for what could be considered as public services.  It
# is noted that we provide a few rejection rulesets as examples but
# these are not required due to the overall REJECT statement above.
#--------------------------------------
echo "  - Setting output filters for public services on all interfaces."

# AUTH: Allow authentication tap indent on all interfaces (but disable it
#       in /etc/inetd.conf).
#
/sbin/ipchains -A output -j ACCEPT -p tcp -s $UNIVERSE auth -d $UNIVERSE

# DNS: If you your Linux server is an authoritative DNS server, you must
# enable this ruleset
#
# Disabled by default
#/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP domain -d $UNIVERSE
#/sbin/ipchains -A output -j ACCEPT -p udp -s $EXTIP domain -d $UNIVERSE

# ICMP: Allow ICMP traffic out
#
# NOTE: Disabling ICMP packets via the firewall ruleset can do far
# more than just stop people from pinging your machine.  Many aspects
# of TCP/IP and its associated applications rely on various ICMP
# messages.  Without ICMP, both your Linux server and internal Masq'ed
# computers might not work.
#
/sbin/ipchains -A output -j ACCEPT -p icmp -s $UNIVERSE -d $UNIVERSE

# NNTP: This allows NNTP-based news out.
#
/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP nntp -d $UNIVERSE

# SMTP: If the Linux servers is either an authoritative SMTP server or
# relay, you must allow this ruleset.
#
/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP smtp -d $UNIVERSE
```



```
#--------------------------------------
# Specific Output Rejections
#--------------------------------------
# These rulesets reject specific traffic that you do not want out of
# the system.
#--------------------------------------
echo "  - Reject specific outputs."

# RPC.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE sunrpc $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP sunrpc -d $UNIVERSE $LOGGING

# Mountd.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 635 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP 635 -d $UNIVERSE $LOGGING

# PPTP.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 1723 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 1723 $LOGGING

# Remote Winsock.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 1745 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 1745 $LOGGING

# NFS.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 2049 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP 2049 -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 2049 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP 2049 -d $UNIVERSE $LOGGING

# PcAnywhere.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 5631 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 5631 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 5632 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE 5632 $LOGGING

# Xwindows.
#
# NOTE: See variable section above for the example range (6000:6010 by default)
# Xwindows can use far more than just ports 6000-6010.
#
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE $XWINDOWS_PORTS $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE $XWINDOWS_PORTS $LOGGING
```

```
# NetBus.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 12345 $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE 12346 $LOGGING

# NetBus Pro.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 20034 $LOGGING

# BackOrofice
/sbin/ipchains -A output -j REJECT -i $EXTIF -p udp -s $EXTIP -d $UNIVERSE/0 31337 $LOGGING

# Win Crash Trojan.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 5742 $LOGGING


# Socket De Troye.
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 30303 $LOGGING

# Unknown Trojan Horse (Master's Paradise [CHR])
/sbin/ipchains -A output -j REJECT -i $EXTIF -p tcp -s $EXTIP -d $UNIVERSE/0 40421 $LOGGING



#------------------------------------
# Output to Explicit Hosts
#------------------------------------
# This controls output to specific external hosts (secure hosts).  This example
# implementation allows ssh and pop-3 protocols out to the secure host.  In
# addition to these rules, we must also explicitly allow the traffic in from
# the remote host.  See the input rules above to see this take place.
#
# Disabled by default.
#------------------------------------
echo "  - Setting output filters for explicit external hosts."

# The secure host
#
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp -d $SECUREHOST $UNPRIV-
PORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp-data -d $SECUREHOST $UNPRIVPORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ssh -d $SECUREHOST $UNPRIVPORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP pop-3 -d $SECUREHOST $UNPRIVPORTS
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP telnet -d $SECUREHOST $UNPRIVPORT
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP telnet -d $SECUREHOST2 $UNPRIVPORT
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp -d $SECUREHOST2 $UNPRIVPORT
#/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $EXTIP ftp-data -d $SECUREHOST2 $UNPRIVPORT
```

```
# Allow all High Ports for return traffic.
#
echo "  - Enabling all output REPLY (TCP/UDP) traffic on high ports."
/sbin/ipchains -A output -j ACCEPT -p tcp -s $EXTIP $UNPRIVPORTS -d $UNIVERSE
/sbin/ipchains -A output -j ACCEPT -p udp -s $EXTIP $UNPRIVPORTS -d $UNIVERSE


#------------------------------------
# Catch All Rule
#------------------------------------
echo "  - Final output catch all rule."

# All other outgoing is denied and logged.  This ruleset should catch
# everything including samba that hasn't already been blocked.#
/sbin/ipchains -A output -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING


#*********************************************************************
# Forwarding Rules
#*********************************************************************
#
echo "----------------------------------------"
echo "Forwarding Rules:"


#------------------------------------
# Enable TCP/IP forwarding and masquerading from the Internal LAN
#------------------------------------

# Masquerade from local net on local interface to anywhere.
#
echo "  - Enable IP Masquerading from the internal LAN."
/sbin/ipchains -A forward -j MASQ -i $EXTIF -s $INTLAN -d $UNIVERSE

# Catch all rule, all other forwarding is denied.
#

/sbin/ipchains -A forward -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING

#*********************************************************************
# The end
#*********************************************************************
echo "----------------------------------------"
echo -e "Firewall implemented. \n\n"
```

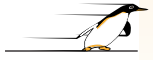


APPENDIX E
Script to modify default permissions of system binaries

The default file permissions on most Linux distributions give away too much information to the user, and/or allow ordinary users to perform actions they have no business doing. The settings below may be paranoid but it's better to be safe than sorry.

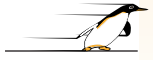
NOTE: most of these permissions will work for a common installation but some files won't exist depending on what packages were and weren't installed. Run this script everytime you reinstall Linux, or install updated packages.

```
#!/bin/sh
#————————————————————————————
# Author: David A. Ranch
# Based on the TrinityOS file permissions corrections
#————————————————————————————

MODE="o-rwx"

# Files in /bin
cd /bin
chmod $MODE linuxconf mount mt setserial umount

# Files in /sbin
cd /sbin
chmod $MODE badblocks ctrlaltdel chkconfig debugfs depmod dump*
chmod $MODE fdisk fsck* ftl* halt hdparm hwclock if* init insmod isapnp
chmod $MODE kerneld killall* lilo mgetty mingetty mk* mod* netreport
chmod $MODE pam* pcinitrd pnpdump portmap quotaon restore runlevel
chmod $MODE stinit swapon tune2fs uugetty

# Files in /usr/bin
cd /usr/bin
chmod $MODE control-panel comanche eject gnome* gpasswd kernelcfg

chmod 755 lp*
chmod 4755 lpr

#NOTE: I feel setting "lpr" to allow any group to execute it is
#      a bad thing.
#
#      I would like to add UNIX users and even the Samba process to
#      the "lp" group already defined in /etc/groups and then be able
#      to put  things back to 4750.  BUT.. this really isn't possible.
#      Linux doesn't support multiple groups per file and Linux
#      doesn't support access lists (ACLs') yet.  So.. you either have
#      either leave these files SUID or run LPRng.

chmod $MODE minicom netcfg
```



APPENDIX E
Script to modify default permissions of system binaries

```
# Files in /usr/sbin
cd /usr/sbin
chmod $MODE at* crond dhc* edquota exportfs ftpshut group* grp*
chmod $MODE imapd in.* inetd ipop* klogd logrotate lp*
chmod 755 lsof
chmod $MODE makemap mouseconfig named* nmbd newusers ntp* ntsysv
chmod $MODE pppd pw* quota* rdev repquota rotatelogs rpc* samba
chmod $MODE setup showmount smb* squid syslogd taper tcpd* time*
chmod $MODE tmpwatch tunelp user* vi* xntp*
```



ADDITIONAL RESOURCES FROM THE SANS INSTITUTE

The SANS Institute is a cooperative research and education organization through which system administrators, security professionals, and network administrators share the lessons they are learning.

It offers educational conferences and in-depth courses, cooperative research reports, and electronic digests of authoritative answers to current questions.

TO ORDER:
http://www.sansstore.org/

ELECTRONIC DIGESTS

SANS NewsBites
A summary, distributed via email, of the dozen most important news articles that have been published on computer security during the past week. Each entry includes a brief highlight of the article and a url to allow you to read the whole story if it is still posted. To Subscribe, send email to info@ sans.org with the subject "SANS NewsBites."

The SANS Network Security Digest
Published every month, and distributed via email, the SANS Network Security Digest reports on the most important new security threats and provides guidance on where to fine the latest patches or additional information on the threats. Each issue can be read in about eight minutes. The SANS Digest is written by Michele Guel with assistance from Matt Bishop, Gene Spafford, Steve Bellovin, Bill Cheswick, Gene Schultz, Marcus Ranum, Rob Kolstad, Hal Pomeranz, Dorothy Denning, and several other leading experts. Subscribe to this digest by sending a message with the subject "subscribe" to digest@sans.org.

The NT Digest
The digest provides updates to NT Security: Step-by-Step plus up-to-date guidance on new Hotfixes and Service Packs that should and should not be implemented. It also summarizes new threats and new bugs found in NT and its services. Subscribe to this digest by sending a message with the subject "subscribe nt digest" to digest@sans.org.

ROADMAP TO NETWORK SECURITY POSTER

The SANS Roadmap To Network Security Wall Poster
Updated twice a year, these posters present "top ten" lists of answers to common questions: the best security books, the best security web sites, the biggest threats, the vendor contacts, the top tools and more. They are mailed automatically to all Network Security Digest subscribers and people who attend the Institute conferences.

COOPERATIVE RESEARCH REPORTS AND PROJECTS

Intrusion Detection Shadow Style: A Step-by-Step Guide
A booklet describing how to perform advanced intrusion detection. It uses the open source, public domain Shadow software developed at the Naval Surface Warfare Center and used to protect more than 15,000 hosts through high-speed intrusion detection and analysis.

The SANS Salary Survey
Published annually, the survey reports salaries of sysadmin, networking, and security professionals based on their primary operating environment (UNIX, NT, Netware, or combination) where they live, the type and size of employer, the machines they manage, whether they are employees or consultants, and other characteristics. It also reports the size of their raises, by salary level, and the principal reasons report for above-average raises. More than 15,000 people participated in the 1999 survey.

Windows NT Security: Step-by-Step
A consensus of security professionals from seventy-seven large user organizations who worked together to develop a list of ninety-three actions in eight phases that should be done to secure and NT server. 36 pages.

Computer Security Incident Handling: Step-by-Step
A consensus of the leading incident handling agencies and experts plus fifty other experienced incident handling professionals. 44 pages.

Windows NT PowerTools: Administrator's Consensus
This is a consensus report which two-hundred and twenty NT administrators shared their experiences in implementing and using twenty of the most popular tools for improving efficiency and security on Windows NT systems.

Solaris Security: Step-by -Step
An extremely valuable collection of the expert knowledge on how to make a Solaris system ready for the internet.

Linux Security: Step-by-Step
Red Hat and other Linux variants have great technology. To use them in the world of business they also need great security. This booklet provides the roadmap.