Linux Home Networking II
Websites At Home

Chapter 1

Why Host Your Own Site?

= =

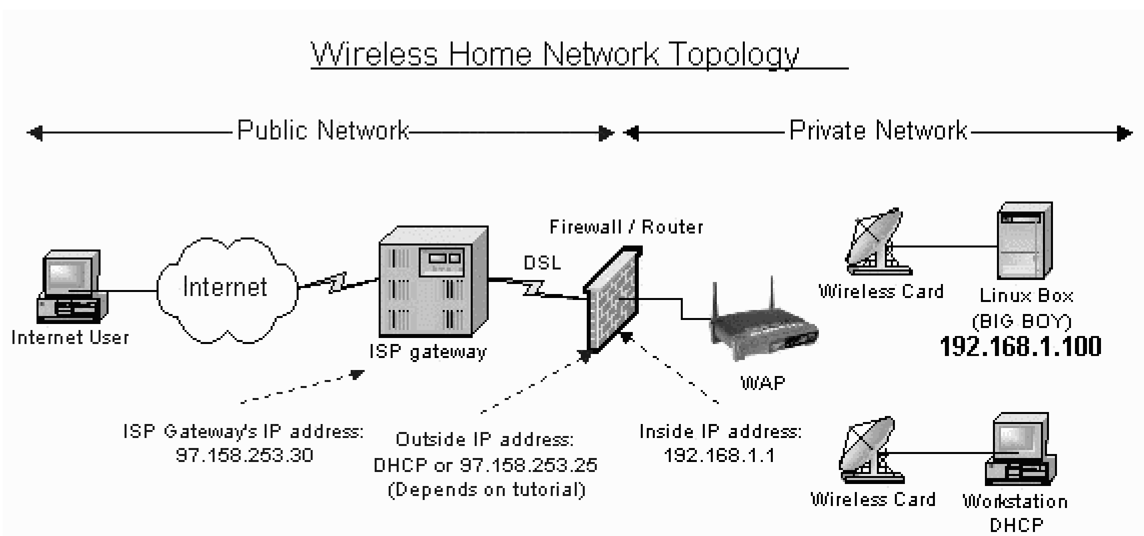
In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

We have assumed you want to host your website in your home or home office using a topology similar to that in the diagram below. Before you do, you should at least weigh the merits of such a move.

Network Diagram



Alternatives To Home Web Hosting

It is easy to find virtual hosting companies on the Web which will offer to host a simple website for about $10 per month.

The steps are fairly straight forward:

Sign up for the virtual hosting service. They will provide you with a login name and password, the IP address of your site plus the name of a private directory on a shared web server in which you'll place your web pages.

Register your domain name, such as www.my-site.com, with companies like Register.com, Verisign or RegisterFree.com. You must make sure your new domain name's DNS records point to the DNS server of the virtual hosting company.

Upload your web pages to your private virtual hosting directory.

Test viewing your site using your IP address in your web browser. It takes about 3-4 days for DNS to propagate across the Web, so you'll probably have to wait at least that long before you'll be able to view your site using your domain, www.my-site.com.

The virtual hosting provider will also offer free backups of your site, technical support, a number of email addresses and an easy to use web based GUI to manage your settings. For an additional charge, many will also provide an e-commerce feature which will allow you to have a shopping cart and customer loyalty programs.

Factors To Consider Before Hosting Yourself

Virtual hosting is the ideal solution for many small websites. There are a number of reasons why you may want to move your website to your home or small office.

Home Based Websites

Pros

Cost: It is possible to host a website on most DSL connections. A website can be hosted on this data circuit for the only additional hardware cost of a network switch and a web server. You should be able to buy this equipment second hand for about $100. If your home already has DSL there would be no additional network connectivity costs. So for a savings of $10 per month the project should pay for itself in less than a year.

New Skills: There is also the additional benefit of learning the new skills required to set up the site. Changes can be made with little delay.

Availability: Reliable virtual hosting facilities may not be available in your country and/or you may not have access to the foreign currency to host your site abroad.

Cons

Lost Services: You lose the convenience of many of the services such as backups, security audits, load balancing, DNS, redundant hardware, data base services and technical support offered by the virtual hosting company. For the home based website these are usually not big issues.

Security: One important factor to consider is the security of your new server. Hosting providers may provide software patches to fix security vulnerabilities on your web servers and may even provide a firewall to protect it. These services may be more difficult to implement at home. There is a chapter on the **iptables** Linux firewall and general security policies for Linux servers to help you overcome these shortcomings.

Technical Ability: Your service provider may have more expertise in setting up your site than you do.

Small Office Based Websites

Pros

Increased Control: You will be able to manage all aspects of your website if it is hosted on a server based either in-house or within your control at a remote data center.

Availability: Reliable virtual hosting facilities may not be available in your country and you may not have access to the foreign currency to host your site abroad.

Cost: The cost of using an external web hosting provider will increase as you purchase more systems administration services. You will eventually be able to justify hosting your website in-house based on this financial fact. In order to determine the break even point of the proposal, you will have to consider the following:

	In-house Web Hosting	
Savings	**Costs**	**Risks**
Monthly out sourced web hosting fee Elimination of the cost of delays to implement desired services.	New hardware & software Possible new application development. Training The percentage of IT staff's time installing and maintaining the site Potential cost of the risks (% likelihood of failure per month X cost of failure)	Likelihood of a failure and expected duration The cost of both the failure and post failure recovery (Hardware, software, data restoration, time)

Cons

Lost Services: You won't have access to the services provided by your old service provider, which may have been highly desirable and cost effective.

Security: Always weigh the degree of security maintained by your hosting provider with that which you expect to provide in-house. Proceed with the server migration only if you feel your staff can handle the job. The chapter on the Linux **iptables** firewall should help make the decision easier.

Technical Ability: You may have to incur additional training costs to ensure that your IT staff has the necessary knowledge to do the job internally.

How To Migrate From An External Provider

The chapter on DNS has a detailed explanation of the steps involved in migrating your website from an external hosting provider to your home or small office. You should also read the sections on mail and web server configuration to help provide a more rounded understanding of the steps involved.

Chapter 2

Linux Firewalls Using iptables

= =

In This Chapter

Chapter 2

© Peter Harrison, www.linuxhomenetworking.com

= =

You can convert your Linux box into a firewall using the IPtables package. This page shows how to convert your Linux box into:

- A firewall while simultaneously being your home website's mail, web and DNS server.
- A router that will use NAT and port forwarding to both protect your home network and have another web server on your home network while sharing the public IP address of your firewall

What Is iptables?

Originally, the most popular firewall / NAT package running on Linux was ipchains. It had a number of limitations, the primary one being that it ran as a separate program and not as part of the kernel. The Netfilter organization decided to create a new product called iptables in order to rectify this shortcoming. As a result of this, iptables is considered a faster and more secure alternative. iptables has now become the default firewall package installed under RedHat Linux.

Download And Install The Iptables Package

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail. The latest version of the RPM for RedHat 9.0 is **iptables-1.2.7a-2.i386.rpm.** Install the package using the following command:

```
[root@bigboy tmp]# rpm -Uvh iptables-1.2.7a-2.i386.rpm
Preparing...    ########################################### [100%]
1:iptables      ########################################### [100%]
[root@bigboy tmp]#
```

How To Get iptables Started

You can start/stop/restart iptables after booting by using the following commands:

```
[root@bigboy tmp]# /etc/init.d/iptables start
[root@bigboy tmp]# /etc/init.d/iptables stop
[root@bigboy tmp]# /etc/init.d/iptables restart
```

To get iptables configured to start at boot:

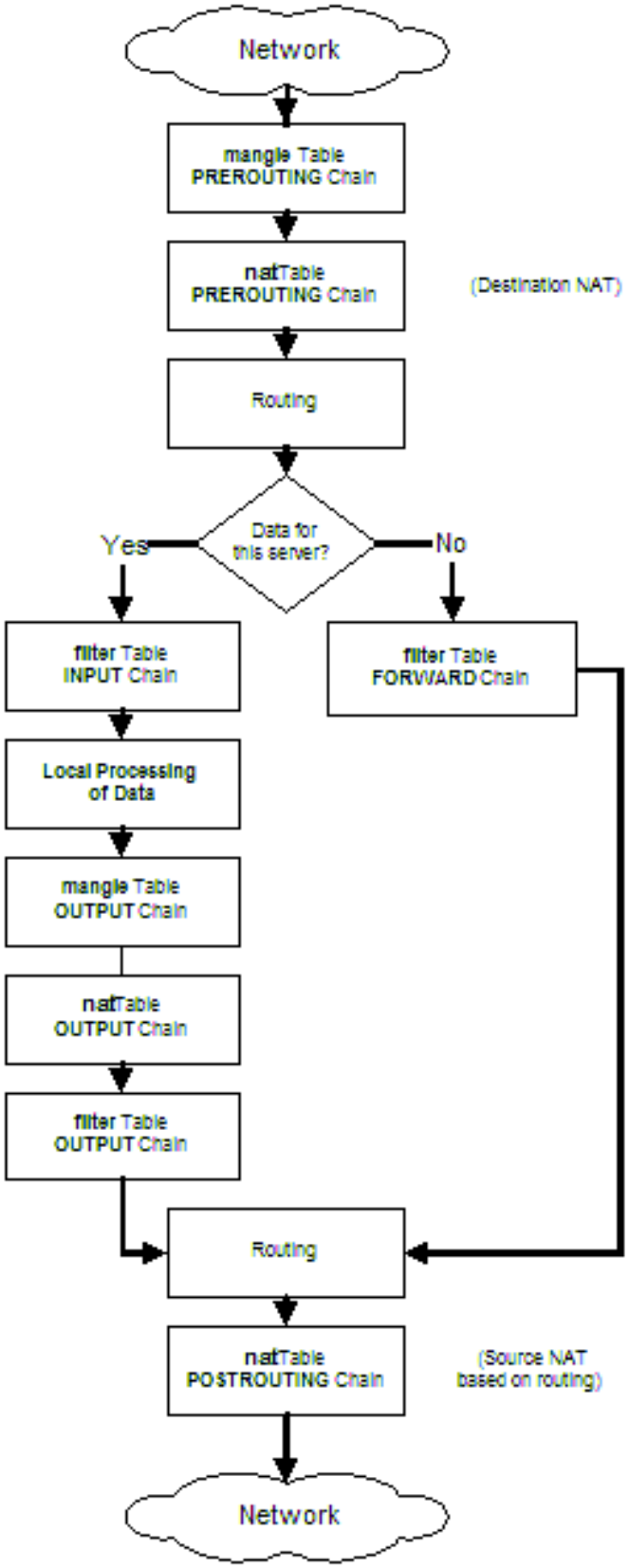
```
[root@bigboy tmp]# chkconfig --level 345 iptables on
```

Packet Processing In iptables

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queue is dedicated to a particular type of packet activity and is controlled by an associated packet transformation / filtering chain. Don't worry if this all seems confusing, there'll be tables and examples of how the concepts are all interlinked.

For example, the chart and graphic below describe the steps taken by iptables when a packet traverses the firewall.

Iptables Packet Flow Diagram



Processing For Packets Routed By The Firewall

Packet flow	Intercepted by iptables chain (Queue)	Packet transformation table associated with this queue	Description of possible modifications by iptables using this transformation table
Packet enters the NIC and is passed to iptables	Mangle	PREROUTING	Modification of the TCP packet quality of service bits. (Rarely used)
	Nat	PREROUTING	Destination network address translation (DNAT). Frequently used to NAT connections from the Internet to your home network
Packet passed to the Linux routing engine	N/A	N/A	N/A Determines whether the packet is destined to a local application or should be sent out another NIC interface
Packet passed back to iptables	Filter	FORWARD	Packet filtering: Packets destined for servers accessible by another NIC on the firewall.
	Nat	POSTROUTING	Source network address translation (SNAT). Frequently used to NAT connections from your home network to the Internet
Packet transmitted out the other NIC	N/A	N/A	N/A

Packet Processing For Data Received By The Firewall

Packet flow	Actions by Operating System	Packet intercepted by iptables table (Queue)	Packet transformation chain associated with this queue	Description of possible modifications by iptables using this transformation table
Packet destined for firewall	Packet enters the NIC from remote server. The packet is intercepted by the iptables mangle, then nat queues	mangle	PREROUTING	Modification of the TCP packet quality of service bits. (Rarely used)
		nat	PREROUTING	Destination network address translation (DNAT)
	The packet is then passed from iptables to the Linux routing engine. The routing engine passes the packet to the target application via the iptables filter queue	filter	INPUT	Packet filtering: Packets destined for the firewall.
The application receives the packet from iptables then processes it.				

Packet Processing For Data Sent By The Firewall

Packet flow	Actions by Operating System	Packet intercepted by iptables table (Queue)	Packet transformation chain associated with this queue	Description of possible modifications by iptables using this transformation table
The application sends data to a remote server				
Packet originating from firewall	The packet is intercepted by iptables which then processes it in the mangle, nat and filter tables	mangle	OUTPUT	Modification of the TCP packet quality of service bits. (Rarely used)
		nat	OUTPUT	Source network address translation (Rarely used)
		filter	OUTPUT	Packet filtering: Packets destined for other servers / devices.
	The packet is then passed to the Linux routing engine which forwards the packet out the correct NIC The packet is intercepted by the iptables nat table	nat	POSTROUTING	Source network address translation (SNAT)
Packet transmitted out a NIC				

Targets And Jumps

You don't have to rely solely on the built-in chains provided by iptables, you can create your own chains. These can be accessed by making them the targets of "jumps" in the built-in chains. So in summary, the targets/jumps tell the rule what to do with a packet that matches the rule perfectly.

There are a number of built-in targets that most rules may use.

Descriptions Of The Most Commonly Used Targets

Target	Description	Most common options
ACCEPT	iptables stops further processing. The packet is handed over to the end application or the operating system for processing	N/A
DROP	iptables stops further processing. The packet is blocked	N/A
LOG	The packet information is sent to the syslog daemon for logging iptables continues processing with the next rule in the table As you can't LOG and DROP at the same time, it is common to have two similar rules in sequence. The first will LOG the packet, the second will DROP it.	--log-prefix "string" Tells iptables to prefix all log messages with a user defined string. Frequently used to tell why the logged packet was dropped
REJECT	Works like the DROP target, but will also return an error message to the host sending the packet that was blocked	--reject-with *qualifier* The qualifier tells what type of reject message is returned. These include: icmp-port-unreachable (default) icmp-net-unreachable icmp-host-unreachable icmp-proto-unreachable icmp-net-prohibited icmp-host-prohibited tcp-reset echo-reply
DNAT	Used to do Destination Network Address Translation. ie.rewriting the destination IP address of the packet	--to-destination *ipaddress* Tells iptables what the destination IP address should be

Target	Description	Most common options
SNAT	Used to do Source Network Address Translation. ie.rewriting the source IP address of the packet The source IP address is user defined	--to-source *<address>[-<address>][:<port>-<port>]* Specifies the source IP address and ports to be used by SNAT.
MASQUERADE	Used to do Source Network Address Translation. ie.rewriting the source IP address of the packet By default the source IP address is the same as that used by the firewall's interface	*[--to-ports <port>[-<port>]]* Specifies the range of source ports the original source port can be mapped to.

Important Iptables Command Switch Operations

We'll now explore how to use iptables command switches used to create your firewall.

General Iptables Match Criteria

iptables command Switch	Description
`-t <table>`	If you don't specify a table, then the filter table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
`-A`	Append rule to end of a chain
`-F`	Flush. Deletes all the rules in the selected table
`-p <protocol-type>`	Match protocol. Types include, icmp, tcp, udp, all
`-s <ip-address>`	Match source IP address
`-d <ip-address>`	Match destination IP address
`-i <interface-name>`	Match "input" interface on which the packet enters.

iptables command Switch	Description
-o <interface-name>	Match "output" interface on which the packet exits

Example:

```
iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1  -p TCP -j ACCEPT
```

In this example iptables is being configured to allow the firewall to accept TCP packets coming in on interface eth0 from any IP address destined for the firewall's IP address of 192.168.1.1

Common TCP and UDP Match Criteria

switches used with -p tcp	Description
--sport <*port*>	TCP source port Can be a single value or a range in the format: *start-port-number:end-port-number*
--dport <port>	TCP destination port Can be a single value or a range in the format: *starting-port:ending-port*
--syn	Used to identify a new connection request ! --syn means, not a new connection request

switches used with -p udp	Description
--sport <*port*>	TCP source port Can be a single value or a range in the format: *starting-port:ending-port*
--dport <port>	TCP destination port Can be a single value or a range in the format: *starting-port:ending-port*

Example:

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP
\
         -sport 1024:65535 -dport 80 -j ACCEPT
```

In this example iptables is being configured to allow the firewall to accept TCP packets to be routed when they enter on interface eth0 from any IP address destined for IP address of 192.168.1.58 that is reachable via interface eth1. The source port is in the range 1024 to 65535 and the destination port is port 80 (www/http)

Common ICMP (Ping) Match Criteria

Matches used with ---icmp-type	Description
--icmp-type <type>	The most commonly used types are echo-reply and echo-request

Example:

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type echo-reply   -j ACCEPT
```

In this example iptables is being configured to allow the firewall send ICMP echo-requests (pings) and in turn, accept the expected ICMP echo-replies.

Common Match Extensions Criteria

TCP/UDP match extensions used with -m multiport	Description
--sport <port, port>	A variety of TCP/UDP source ports separated by commas
--dport <port, port>	A variety of TCP/UDP destination ports separated by commas

TCP/UDP match extensions used with -m multiport	Description
--dport <port, port>	A variety of TCP/UDP ports separated by commas. Source and destination ports are assumed to be the same.

Match extensions used with -m state	Description
--state <*state*>	The most frequently tested states are: ESTABLISHED The packet is part of a connection which has seen packets in both directions NEW The packet is the start of a new connection RELATED The packet is starting a new secondary connection. This is a common feature of protocols such as an FTP data transfer, or an ICMP error.

Example:

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP
\
         -sport 1024:65535 -m multiport -dport 80,443 -j ACCEPT

iptables -A FORWARD -d 0/0 -o eth0 -s 192.168.1.58 -i eth1 -p TCP
\
         -m state --state ESTABLISHED -j ACCEPT
```

This is an expansion on the previous example. Here iptables is being configured to allow the firewall to accept TCP packets to be routed when they enter on interface eth0 from any IP address destined for IP address of 192.168.1.58 that is reachable via interface eth1. The source port is in the range 1024 to 65535 and the destination ports are port 80 (www/http) and 443 (https).

We are also allowing the return packets from 192.168.1.58 to be accepted too. Instead of stating the source and destination ports, it is sufficient to allow packets related to established connections using the -m state and --state ESTABLISHED options.

Using User Defined Chains

As stated in the introduction, iptables can be configured to have user-defined chains. This feature is frequently used to help streamline the processing of packets. For example, instead of having a single chain for all protocols, it is possible to have a chain that determines the protocol type for the packet and then hands off the actual final processing to a protocol specific chain. In other words, you can replace a long chain with a main stubby chain pointing to multiple stubby chains thereby shortening the total length of all chains the packet has to pass through.

Example:

```
iptables -A INPUT -i eth0  -d 206.229.110.2 -j fast-input-queue
iptables -A OUTPUT -o eth0 -s 206.229.110.2 -j fast-output-queue

iptables -A fast-input-queue  -p icmp -j icmp-queue-in
iptables -A fast-output-queue -p icmp -j icmp-queue-out

iptables -A icmp-queue-out -p icmp --icmp-type echo-request \
         -m state --state NEW -j ACCEPT

iptables -A icmp-queue-in -p icmp --icmp-type echo-reply -j ACCEPT
```

In this example we have six queues with the following characteristics to help assist in processing speed:

Chain	Description
INPUT	The regular built-in INPUT chain in iptables
OUTPUT	The regular built-in OUTPUT chain in iptables
fast-input-queue	Input chain dedicated to specific protocols
fast-output-queue	Output chain dedicated to specific protocols
icmp-queue-out	Output queue dedicated to ICMP
icmp-queue-in	Intput queue dedicated to ICMP

Sample iptables Scripts

Here are some sample scripts you can use to get iptables working for you. It is best to invoke these from your **/etc/rc.d/rc.local** file so that the firewall script is run every time you boot up. Pay special attention to the logging example at the end.

The "basic initialization" script snippet should also be included in all your scripts to ensure the correct initialization of your chains should you decide to restart your script after startup. This chapter also includes other snippets that will help you get basic functionality. It should be a good guide to get you started.

You then can use the Appendix to find a detailed script once you feel more confident. It shows you how to allow your firewall to:

Be used as a Linux Web / Mail / DNS server

Be the NAT router for your home network

Prevent various types of attacks using corrupted TCP, UDP and ICMP packets.

Outbound passive FTP access from the firewall

There are also simpler code snippets in the Appendix for:

Inbound and outbound FTP connections to / from your firewall

Basic Initialization

It is a good policy, in any iptables script you write, to initialize your chain and table settings with known values. The "filter" table's INPUT, FORWARD and OUTPUT chains should DROP packets by default for the best security. However, it is not good policy to make your "nat" and "mangle" tables DROP packets by default. This is because these tables are queried before the "filter" table, and if all packets that don't match the "nat" and "mangle" rules are DROP-ped, then they will not reach the the INPUT, FORWARD and OUTPUT chains and won't be processed.

Additional ALLOW rules should be added to the end of this script snippet.

```
#!/bin/bash

#----------------------------------------------------------------
# Load modules for FTP connection tracking and NAT - You may need
# them later
#----------------------------------------------------------------

modprobe ip_conntrack_ftp
modprobe iptable_nat

#----------------------------------------------------------------
# Initialize all the chains by removing all the rules
# tied to them
#----------------------------------------------------------------

iptables --flush
iptables -t nat --flush
iptables -t mangle --flush
```

```
#--------------------------------------------------------------
# Now that the chains have been initialized, the user defined
# chains should be deleted. We'll recreate them in the next step
#--------------------------------------------------------------

iptables --delete-chain
iptables -t nat --delete-chain
iptables -t mangle --delete-chain

#--------------------------------------------------------------
# If a packet doesn't match one of the built in chains, then
# The policy should be to drop it
#--------------------------------------------------------------

iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP

#--------------------------------------------------------------
# The loopback interface should accept all traffic
# Necessary for X-Windows and other socket based services
#--------------------------------------------------------------

iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Allowing DNS Access To Your Firewall

You'll almost certainly want your firewall to make DNS queries to the Internet. The following statements will apply not only for firewalls acting as DNS clients but also for firewalls working in a caching or regular DNS server role.

```
#--------------------------------------------------------------
# Allow outbound DNS queries from the FW and the replies too
#
# - Interface eth0 is the internet interface
#
# Zone transfers use TCP and not UDP. Most home networks
# / websites using a single DNS server won't require TCP
statements
#
#--------------------------------------------------------------

iptables -A OUTPUT -p udp -o eth0 --dport 53 --sport 1024:65535 \
         -j ACCEPT

iptables -A INPUT -p udp -i eth0 --sport 53 --dport 1024:65535 \
         -j ACCEPT
```

Allowing WWW And SSH Access To Your Firewall

This sample snippet is for a web server that is managed remotely by its system administrator via secure shell (SSH) sessions. Inbound packets destined for ports 80 and 22 are allowed thereby making the first steps in establishing a connection. It isn't necessary to specify these ports for the return leg as outbound packets for all established connections are allowed. Connections initiated by persons logged into the webserver will be denied as outbound NEW connection packets aren't allowed.

```
#--------------------------------------------------------------
# Allow previously established connections
# - Interface eth0 is the internet interface
#--------------------------------------------------------------

iptables -A OUTPUT -o eth0 -m state --state ESTABLISHED,RELATED \
  -j ACCEPT

#--------------------------------------------------------------
# Allow port 80 (www) and 22 (SSH) connections to the firewall
#--------------------------------------------------------------

iptables -A INPUT -p tcp -i eth0 --dport 22 –sport 1024:65535 \
  -m state –state NEW -j ACCEPT

iptables -A INPUT -p tcp -i eth0 --dport 80 –sport 1024:65535 \
  -m state –state NEW -j ACCEPT
```

Allowing Your Firewall To Access The Internet

The following iptables sample script allows a user on the firewall to use a web browser to surf the Internet. TCP port 80 is used for HTTP traffic and port 443 is used for HTTPS (secure HTTP frequently used for credit card transactions). HTTPS is also used by RedHat Linux servers using up2date.

```
#--------------------------------------------------------------
# Allow port 80 (www) and 443 (https) connections to the firewall
#--------------------------------------------------------------

iptables -A OUTPUT -j ACCEPT -m state --state NEW \
  -o eth0 –p tcp -m multiport --dport 80,443 --sport 1024:65535

#--------------------------------------------------------------
# Allow previously established connections
# - Interface eth0 is the internet interface
#--------------------------------------------------------------
iptables -A INPUT -j ACCEPT -m state --state
ESTABLISHED,RELATED  \
-i eth0 –p tcp
```

If you want all TCP traffic originating from the firewall to be accepted then you can remove the following section from the snippet above:

```
-m multiport --dport 80,443 --sport 1024:65535
```

Allow Your Home Network To Access The Firewall

In this example, **eth1** is directly connected to a home network using IP addresses from the 192.168.1.0 network. All traffic between this network and the firewall is simplistically assumed to be trusted and allowed.

Further rules will be needed for the interface connected to the Internet to allow only specific ports, types of connections and possibly even remote servers to have access to your firewall and home network.

```
#--------------------------------------------------------------
# Allow all bidirectional traffic from your firewall to the
# protected network
# - Interface eth1 is the private network interface
#--------------------------------------------------------------

iptables -A INPUT   -j ACCEPT -p all -s 192.168.1.0/24 -i eth1
iptables -A OUTPUT  -j ACCEPT -p all -d 192.168.1.0/24 -o eth1
```

Masquerading (Many to One NAT)

As explained in the Introduction to Networking chapter, masquerading is another word for what many call "many to one" NAT. In other words, traffic from all devices on one or more protected networks will appear as if it originated from a single IP address on the Internet side of the firewall.

Note: You can force your firewall to use a pre-defined IP address for many to one NAT by using an IP alias together with iptable's POSTROUTING queue. This can come in handy when you want machines from one network to all NAT to one alias address, and those from another network to all NAT to a second alias. The advantage of masquerading is that you never have to specify the NAT IP address as it always uses the firewall's interface's main IP address. This makes it much easier to configure NAT with DHCP. There is an example using the POSTROUTING statement for NAT using fixed IP addresses in the static NAT section below. Though it may be tempting, configuring IP aliases with DHCP is not a good idea as it may cause conflicts with other DHCP client computers.

iptables requires the iptables_nat module to be loaded with the "modprobe" command for the masquerade feature to work. Masquerading also depends on the Linux operating system being configured to support routing between the internet and private network interfaces of the firewall. This is done by enabling "IP forwarding" or routing by giving the file **/proc/sys/net/ipv4/ip_forward** the value "1" as opposed to the default disabled value of "0".

Once masquerading has been achieved using the POSTROUTING chain of the "nat" table, iptables will have to be configured to allow packets to flow between the two interfaces. This is done using the FORWARD chain of the "filter" table. More specifically, packets related to NEW and ESTABLISHED connections will be allowed outbound to the Internet, while only

packets related to ESTABLISHED connections will be allowed inbound. This helps to protect the home network from persons trying to initiate connections from the Internet. An example follows:

```
#--------------------------------------------------------------
# Load the NAT module
#--------------------------------------------------------------

modprobe iptable_nat


#--------------------------------------------------------------
# Allow masquerading
# Enable routing by modifying the ip_forward /proc filesystem file
# - Interface eth0 is the internet interface
# - Interface eth1 is the private network interface
#--------------------------------------------------------------

iptables -A POSTROUTING -t nat -o eth0 -s 192.168.1.0/24 -d 0/0 \
         -j MASQUERADE

echo 1 > /proc/sys/net/ipv4/ip_forward

#--------------------------------------------------------------
# Prior to masquerading, the packets are routed via the filter
# table's FORWARD chain.
# Allowed outbound: New, established and related connections
# Allowed inbound : Established and related connections
#--------------------------------------------------------------

iptables -A FORWARD -t filter -i eth1 -m state \
         --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -t filter -i eth0 -m state \
         --state ESTABLISHED,RELATED     -j ACCEPT
```

Note: If you configure your firewall to do masquerading, then if should be the used as the default gateway for all your servers on the network.

Port Forwarding Type NAT (DHCP DSL)

In many cases home users may get a single DHCP public IP address from their ISP. If their Linux firewall is their interface to the Internet and they want to host a website on one of the NAT protected home servers then they will have to use the “port forwarding” technique.

Here the combination of the firewall's single IP address, the remote server’s IP address and the source/destination port of the traffic can be used to uniquely identify a traffic flow. All traffic that matches a particular combination of these factors may then be forwarded to a single server on the private network.

Port forwarding is handled by the PREROUTING chain of the "nat" table. As in masquerading, the iptables_nat module will have to be loaded and routing enabled for port forwarding to work. Routing too will have to be allowed in iptables with the FORWARD chain, this would include all NEW inbound connections from the Internet matching the port

forwarding port plus all future packets related to the ESTABLISHED connection in both directions. An example follows:

```
#-----------------------------------------------------------------
# Load the NAT module
#-----------------------------------------------------------------

modprobe iptable_nat

#-----------------------------------------------------------------
# Get the IP address of the Internet interface eth0 (linux only)
#
# You'll have to use a different expression to get the IP address
# for other operating systems which have a different ifconfig
output
# or enter the IP address manually in the PREROUTING statement
#
# This is best when your firewall gets its IP address using DHCP.
# The external IP address could just be hard coded (“typed in
# normally”)
#-----------------------------------------------------------------

external_int="eth0"
external_ip="`ifconfig $external_int | grep 'inet addr' | \
                      awk '{print $2}' | sed -e 's/.*://'`"

#-----------------------------------------------------------------
# Allow port forwarding for traffic destined to port 80 of the
# firewall’s IP address to be forwarded to port 8080 on server
# 192.168.1.200
#
# Enable routing by modifying the ip_forward /proc filesystem file
# - Interface eth0 is the internet interface
# - Interface eth1 is the private network interface
#-----------------------------------------------------------------

iptables -t nat -A PREROUTING -p tcp -i eth0 -d $external_ip \
     --dport 80 --sport 1024:65535 -j DNAT --to 192.168.1.200:8080

echo 1 > /proc/sys/net/ipv4/ip_forward

#-----------------------------------------------------------------
# After DNAT, the packets are routed via the filter table's
# FORWARD chain.
# Connections on port 80 to the target machine on the private
# network must be allowed.
#-----------------------------------------------------------------

iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.200 \
    --dport 8080 --sport 1024:65535 -m state --state NEW -j ACCEPT

iptables -A FORWARD -t filter -i eth1 -m state \
         --state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -t filter -i eth0 -m state \
         --state ESTABLISHED,RELATED -j ACCEPT
```

```
# PREROUTING statements for 1:1 NAT
# (Connections originating from the Internet)

iptables -t nat -A PREROUTING -d 97.158.253.26 -i eth0 \
         -j DNAT --to-destination 192.168.1.100
iptables -t nat -A PREROUTING -d 97.158.253.27 -i eth0 \
         -j DNAT --to-destination 192.168.1.101
iptables -t nat -A PREROUTING -d 97.158.253.28 -i eth0 \
         -j DNAT --to-destination 192.168.1.102

# POSTROUTING statements for 1:1 NAT
# (Connections originating from the home network servers)

iptables -t nat -A POSTROUTING -s 97.158.253.26 -o eth0 \
         -j SNAT --to-source 192.168.1.100
iptables -t nat -A POSTROUTING -s 97.158.253.27 -o eth0 \
         -j SNAT --to-source 192.168.1.101
iptables -t nat -A POSTROUTING -s 97.158.253.28 -o eth0 \
         -j SNAT --to-source 192.168.1.102

# POSTROUTING statements for Many:1 NAT
# (Connections originating from the entire home network)

iptables -t nat -A POSTROUTING -s 192.168.1.0/24 \
         -j SNAT -o eth1 --to-source 97.158.253.29

# Allow forwarding to each of the servers configured for 1:1 NAT
# (For connections originating from the Internet. Notice how you
# use the real IP addresses here)

iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.100 \
    -m multiport --dport 80,443,22 -m multiport --sport 1024:65535
\
    -m state --state NEW -j ACCEPT

iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.101 \
    -m multiport --dport 80,443,22 -m multiport --sport 1024:65535
\
    -m state --state NEW -j ACCEPT

iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.102 \
    -m multiport --dport 80,443,22 -m multiport --sport 1024:65535
\
    -m state --state NEW -j ACCEPT

# Allow forwarding for all New and Established SNAT connections
# originating on the home network AND already established
# DNAT connections
iptables -A FORWARD -t filter -i eth1 -m state \
         --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Allow forwarding for all 1:1 NAT connections originating on
# the Internet that have already passed through the NEW forwarding
# statements above
iptables -A FORWARD -t filter -i eth0 -m state \
         --state ESTABLISHED,RELATED -j ACCEPT
```

Logging & Troubleshooting

You track packets passing through the iptables list of rules using the LOG target. You should be aware that the LOG target:

will log all traffic that matches the iptables rule in which it is located.

automatically writes an entry to the **/var/log/messages** file and then executes the next rule.

Therefore if you want to log only unwanted traffic then you have to add a matching rule with a DROP target immediately after the LOG rule. If you don't, you'll find yourself logging both desired and unwanted traffic with no way of discerning between the two as by default iptables doesn't state why the packet was logged in its log message.

This example logs a summary of failed packets to the file **/var/log/messages**. You can use the contents of this file to determine what TCP/UDP ports you need to open to provide access to specific traffic that is currently stopped.

```
#-----------------------------------------------------------------
# Log and drop all other packets to file /var/log/messages
# Without this we could be crawling around in the dark
#-----------------------------------------------------------------

iptables -A OUTPUT -j LOG
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG
iptables -A OUTPUT -j DROP
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
```

Here are some examples of the output of this file:

Firewall denying replies to DNS queries (UDP port 53) destined to server 192.168.1.102 on the home network.

```
Feb 23 20:33:50 bigboy kernel: IN=wlan0 OUT=
MAC=00:06:25:09:69:80:00:a0:c5:e1:3e:88:08:00 SRC=192.42.93.30
DST=192.168.1.102 LEN=220 TOS=0x00 PREC=0x00 TTL=54 ID=30485
PROTO=UDP SPT=53 DPT=32820 LEN=200
```

Firewall denying Windows NetBIOS traffic (UDP port 138)

```
Feb 23 20:43:08 bigboy kernel: IN=wlan0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:06:25:09:6a:b5:08:00 SRC=192.168.1.100
DST=192.168.1.255 LEN=241 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF
PROTO=UDP SPT=138 DPT=138 LEN=221
```

Firewall denying Network Time Protocol (NTP UDP port 123)

```
Feb 23 20:58:48 bigboy kernel: IN= OUT=wlan0 SRC=192.168.1.102
DST=207.200.81.113 LEN=76 TOS=0x10 PREC=0x00 TTL=64 ID=0 DF
PROTO=UDP SPT=123 DPT=123 LEN=56
```

Note: The traffic in all these examples isn't destined for the firewall. Therefore you should check your INPUT, OUTPUT, FORWARD and NAT related statements. If the firewall's IP address is involved, then you should focus on the INPUT, OUTPUT statements
If nothing shows up in the logs, then follow the steps in the Network Troubleshooting chapter to determine whether the data is reaching your firewall at all, and if it is not, the location your network that could be causing the problem.
Troubleshooting NAT: As a general rule, you won't be able to access the public NAT IP addresses from servers on your home network. Basic NAT testing will require you to ask a friend to try to connect to your home network from the Internet.
You can then use the logging output in **/var/log/messages** to make sure that:
the translations are occurring correctly and
iptables isn't dropping the packets after translation occurs

Chapter 3

Linux FTP Server Setup

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

This chapter will show you how to convert your Linux box into an FTP server using the VSFTP package. The RedHat software download site runs on VSFTP.

FTP Overview

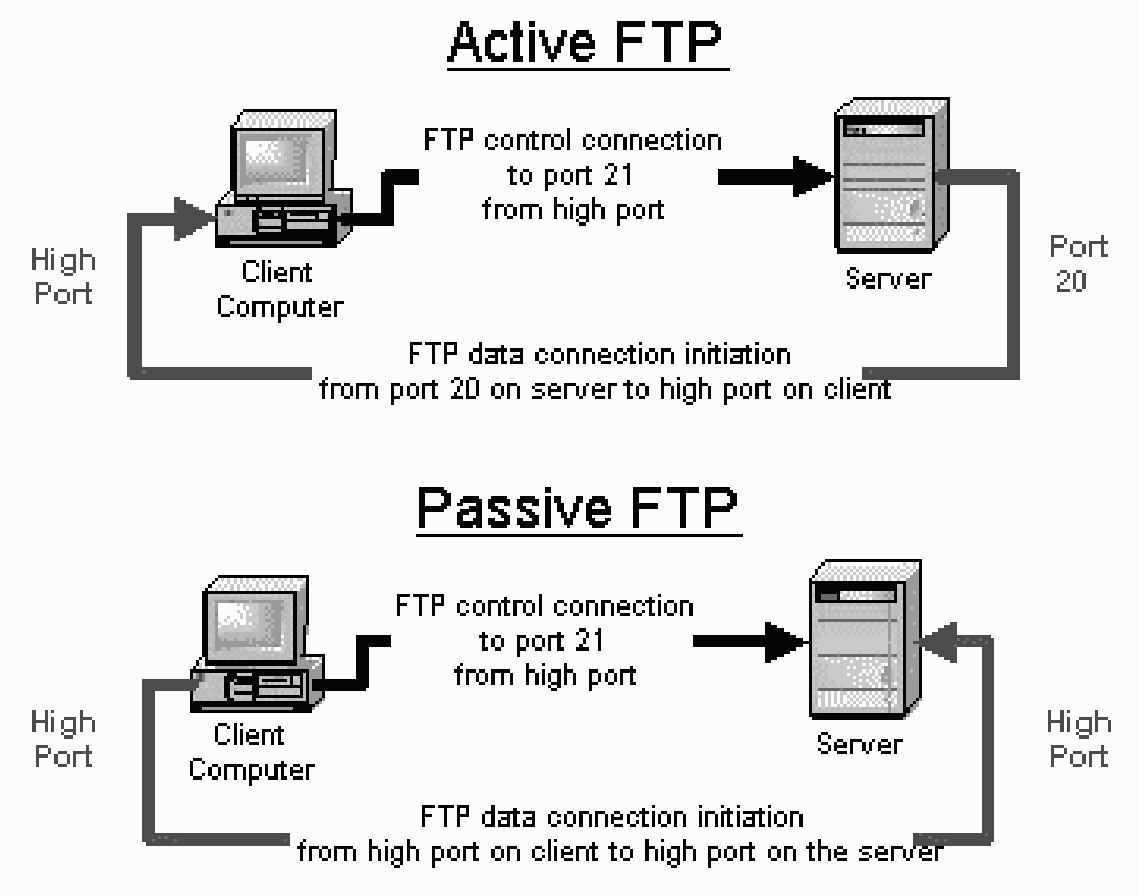
File Transfer Protocol (FTP) is a common method of copying files between computer systems. Two TCP ports are used to do this:

FTP Control Channel - TCP Port 21

All commands you send and the ftp server's responses to those commands will go over the control connection, but any data sent back (such as "**ls**" directory lists or actual file data in either direction) will go over the data connection.

FTP Data Channel - TCP Port 20

Used for all data sent between the client and server.



Active FTP

Active FTP works as follows:

Your client connects to the FTP server by establishing an FTP control connection to port 21 of the server. Your commands such as **'ls'** and **'get'** are sent over this connection.

Whenever the client requests data over the control connection, the **server** initiates data transfer connections back to the client. The source port of these data transfer connections is always port 20 on the server, and the destination port is a high port on the client.

Thus the **'ls'** listing that you asked for comes back over the "port 20 to high port connection", not the port 21 control connection.

FTP active mode data transfer therefore does this in a counter intuitive way to the TCP standard as it selects port 20 as it's source port (not a random high port > 1024) and connects back to the client on a random high port that has been pre-negotiated on the port 21 control connection.

Active FTP may fail in cases where the client is protected from the Internet via many to one NAT (masquerading). This is because the firewall will not know which of the many servers behind it should receive the return connection.

Passive FTP

Passive FTP works as follows:

Your client connects to the FTP server by establishing a FTP control connection to port 21 of the server. Your commands such as **'ls'** and 'get' are sent over that connection.

Whenever the client requests data over the control connection, the **client** initiates the data transfer connections to the server. The source port of these data transfer connections is always a high port on the client with a destination port of a high port on the server.

Passive FTP should be viewed as the server never making an active attempt to connect to the client for FTP data transfers.

Passive FTP works better for clients protected by a firewall as the client always initiates the required connections.

Problems With FTP And Firewalls

FTP frequently fails when the data has to pass through a firewall as FTP uses a wide range of unpredictable TCP ports and firewalls are designed to limit data flows to predictable TCP ports. There are ways to overcome this as explained in the following sections.

The Appendix has examples of how to configure the **vsftpd** Linux filewall to function with both active and passive FTP.

Client Protected By A Firewall Problem

Typically firewalls don't let any incoming connections at all, this will frequently cause active FTP not to function. This type of FTP failure has the following symptoms:

The **active** ftp connection appears to work when the client initiates an outbound connection to the server on port 21. The connection appears to hang as soon as you do an "ls" or a "dir" or a "get". This is because the firewall is blocking the return connection from the server to the client. (From port 20 on the server to a high port on the client)

Solutions

Here are the general firewall rules you'll need to allow FTP clients through a firewall:

Client Protected by Firewall - Required Rules for FTP

Method	Source Address	Source Port	Destination Address	Destination Port	Connection Type
Allow outgoing control connections to server					
Control Channel	FTP client/ network	High	FTP server**	21	New
	FTP server**	21	FTP client/ network	High	Established*
Allow the client to establish data channels to remote server					
Active FTP	FTP server**	20	FTP client /network	High	New
	FTP client/ network	High	FTP server**	20	Established*
Passive FTP	FTP client/ network	High	FTP server**	High	New
	FTP server**	High	FTP client/ network	High	Established*

*Many home based firewall/routers automatically allow traffic for already established connections. This rule may not be necessary in all cases.

** in some cases, you may want to allow all Internet users to have access, not just a specific client server or network.

Server Protected By A Firewall Problem

Typically firewalls don't let any connections come in at all. FTP server failure due to firewalls in which the **active** ftp connection from the client doesn't appear to work at all

Solutions

Here are the general firewall rules you'll need to allow FTP severs through a firewall

Server Protected by Firewall - Required Rules for FTP

Method	Source Address	Source Port	Destination Address	Destination Port	Connection Type
Allow incoming control connections to server					
Control Channel	FTP client/ network**	High	FTP server	21	New
	FTP server	21	FTP client/ network**	High	Established*
Allow server to establish data channel to remote client					
Active FTP	FTP server	20	FTP client/network**	High	New
	FTP client/ network**	High	FTP server	20	Established*
Passive FTP	FTP client/ network**	High	FTP server	High	New
	FTP server	High	FTP client/ network**	High	Established*

*Many home based firewall/routers automatically allow traffic for already established connections. This rule may not be necessary in all cases.

** in some cases, you may want to allow all Internet users to have access, not just a specific client server or network.

How To Download And Install The VSFTP Package

As explained previously, RedHat software is installed using RPM packages. In version 9.0 of the operating system, the VSFTP RPM file is named:

```
vsftpd-1.1.3-8.i386.rpm
```

Downloading and installing RPMs isn't hard. If you need a refresher, the RPM chapter covers how to do this in detail.

Now download the file to a directory such as **/tmp** and install it using the "rpm" command:

```
[root@bigboy tmp]# rpm -Uvh vsftpd-1.1.3-8.i386.rpm
Preparing... ########################################### [100%]
1:vsftpd     ########################################### [100%]
[root@bigboy tmp]#
```

How To Get VSFTP Started

Redhat Version 9.0 And Newer

You can start/stop/restart vsftpd after booting by using the following commands:

```
[root@bigboy tmp]# /etc/init.d/vsftpd start
[root@bigboy tmp]# /etc/init.d/vsftpd stop
[root@bigboy tmp]# /etc/init.d/vsftpd restart
```

To get vsftpd configured to start at boot:

```
[root@bigboy tmp]# chkconfig --level 345 vsftpd on
```

Redhat Version 8.0 And Older

The starting and stopping of VSFTP is controlled by xinetd via the **/etc/xinetd.d/vsftpd** file. VSFTP is deactivated by default, so you'll have to edit this file to start the program. Make sure the contents look like this. The disable feature must be set to "no" to accept connections.

```
service ftp
{
   disable = no
   socket_type = stream
   wait = no
   user = root
   server = /usr/sbin/vsftpd
   nice = 10
}
```

You will then have to restart xinetd for these changes to take effect using the startup script in the **/etc/init.d** directory.

```
[root@aqua tmp]#  /etc/init.d/xinetd restart
Stopping xinetd: [  OK  ]
Starting xinetd: [  OK  ]
[root@aqua tmp]#
```

Naturally, to disable VSFTP once again, you'll have to edit **/etc/xinetd.d/vsftpd**, set "disable" to "yes" and restart xinetd.

Testing To See If VSFTP Is Running

You can always test whether the VSFTP process is running by using the **netstat –a** command which lists all the TCP and UDP ports on which the server is listening for traffic. The example below shows the expected output, there would be no output at all if VSFTP wasn't running.

```
[root@bigboy root]# netstat -a | grep ftp
tcp        0      0      *:ftp          *:*         LISTEN
[root@bigboy root]#
```

What Is Anonymous FTP?

Anonymous FTP is used by web sites that need to exchange files with numerous unknown remote users. Common uses include downloading software updates and MP3s to uploading diagnostic information for a technical support engineer's attention. Unlike regular FTP where you login with a user-specific username, anonymous FTP only requires a username of "anonymous" and your email address for the password. Once logged in to a VSFTP server, you'll automatically have access to only the default anonymous FTP directory **/var/ftp** and all its subdirectories.

As seen in the chapter on RPMs, using anonymous FTP as a remote user is fairly straight forward. VSFTP can be configured to support user based and or anonymous FTP in its configuration file.

The vsftpd.conf File

VSFTP only reads the contents of its **/etc/vsftpd.conf** (before RedHat 9.0) or **/etc/vsftpd/vsftpd.conf** (RedHat 9.0 and newer) configuration file when it starts, so you'll have to restart xinetd each time you edit the file in order for the changes to take effect.

This file uses a number of default settings you need to know. By default, VSFTP runs as an anonymous FTP server. Unless you want any remote user to log into to your default FTP directory using a username of "ananoymous" and a password that's the same as their email address, I would suggest turning this off. The configuration file's **anonymous_enable** instruction can be commented out by using a "#" to disable this feature. You'll also want to simultaneously enable local users to be able to log in by uncommenting the **local_enable** instruction.

By default VSFTP only allows anonymous FTP downloads to remote users, not uploads from them. Also by default, VSFTP doesn't allow remote users to create directories on your FTP server and it logs FTP access to the **/var/log/vsftpd.log** log file.

The configuration file is fairly straight forward as you can see in the snippet below. Remove/add the "#" at the beginning of the line to "activate/deactivate" the feature on each line.

```
# Allow anonymous FTP?
anonymous_enable=YES
...
...
```

```
# Uncomment this to allow local users to log in.
local_enable=YES
...
...
# Uncomment this to enable any form of FTP write command.
# (Needed even if you want local users to be able to upload files)
write_enable=YES
...
...
# Uncomment to allow the anonymous FTP user to upload files. This
only
# has an effect if global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
...
...
# Uncomment this if you want the anonymous FTP user to be able to
create
# new directories.
#anon_mkdir_write_enable=YES
...
...
# Activate logging of uploads/downloads.
xferlog_enable=YES
...
...
# You may override where the log file goes if you like.
# The default is shown# below.
#xferlog_file=/var/log/vsftpd.log
```

FTP Security Issues

The /etc/vsftpd.ftpusers File

For added security you may restrict FTP access to certain users by adding them to the list of users in this file. Do not delete entries from the default list, it is best to add.

Anonymous Upload

If you want remote users to write data to your FTP server then it is recommended you create a write-only directory within **/var/ftp/pub**. This will allow your users to upload, but not access other files uploaded by other users. Here are the commands to do this:

```
[root@bigboy tmp]# mkdir /var/ftp/pub/upload
[root@bigboy tmp]# chmod 733 /var/ftp/pub/upload
```

FTP Greeting Banner

Change the default greeting banner in the **vsftpd.conf** file to make it harder for malicious users to determine the type of system you have.

```
ftpd_banner= New Banner Here
```

Using SCP As Secure Alternative To FTP

One of the disadvantages of FTP is that it does not encrypt your username and password. This could make your user account vulnerable to an unauthorized attack from a person eavesdropping on the network connection. Secure Copy (SCP) provides encryption and could be considered as an alternative to FTP for trusted users. SCP however does not support anonymous services, a feature that FTP does.

Example #1:

FTP Users With Only Read Access To A Shared Directory

In this example, anonymous FTP is not desired, but a group of trusted users need to have read only access to a directory for downloading files. Here are the steps:

Enable FTP. Edit the **/etc/xinetd.d/vsftp** and set the disable value to "no".

Disable anonymous FTP. Comment out the **anonymous_enable line** in the **vsftpd.conf** file like this:

```
# Allow anonymous FTP?
# anonymous_enable=YES
```

Enable individual logins by making sure you have the **local_enable** line uncommented in the **vsftpd.conf** file like this:

```
# Uncomment this to allow local users to log in.
local_enable=YES
```

Create a user group and shared directory. In this case we'll use "**/home/ftp-users**" and a user group name of "**ftp-users**" for the remote users.

```
[root@bigboy tmp]# groupadd ftp-users
[root@bigboy tmp]# mkdir /home/ftp-docs
```

Make the directory accessible to the ftp-users group.

```
[root@bigboy tmp]# chmod 750 /home/ftp-docs
[root@bigboy tmp]# chown root:ftp-users /home/ftp-docs
```

Add users, and make their default directory **/home/ftp-docs**

```
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user1
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user2
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user3
[root@bigboy tmp]# useradd -g ftp-users -d /home/ftp-docs user4
[root@bigboy tmp]# passwd user1
[root@bigboy tmp]# passwd user2
[root@bigboy tmp]# passwd user3
[root@bigboy tmp]# passwd user4
```

Copy files to be downloaded by your users into the **/home/ftp-docs** directory

Change the permissions of the files in the **/home/ftp-docs** directory for read only access by the group

```
[root@bigboy tmp]# chown root:ftp-users /home/ftp-docs/*
[root@bigboy tmp]# chmod 740 /home/ftp-docs/*
```

Users should now be able to log in via ftp to the server using their new user names and passwords. If you absolutely don't want any FTP users to be able to write to any directory then you should comment out the **write_enable** line in your **vsftpd.conf** file like this:

```
#write_enable=YES
```

Restart vsftp for the configuration file changes to take effect.

Sample Login Session To Test Funtionality

Check for the presence of a test file on the ftp client server.

```
[root@smallfry tmp]# ll
total 1
-rw-r--r-- 1 root root 0 Jan 4 09:08 testfile
[root@smallfry tmp]#
```

Connect to bigboy via FTP

```
[root@smallfry tmp]# ftp 192.168.1.100
Connected to 192.168.1.100 (192.168.1.100).
220 ready, dude (vsFTPd 1.1.0: beat me, break me)
Name (192.168.1.100:root): user1
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

As expected, we can't do an upload transfer of "testfile" to bigboy.

```
ftp> put testfile
local: testfile remote: testfile
227 Entering Passive Mode (192,168,1,100,181,210)
553 Could not create file.
ftp>
```

We can view and download a copy of the VSFTP RPM

```
ftp> ls
227 Entering Passive Mode (192,168,1,100,35,173)
150 Here comes the directory listing.
-rwxr----- 1 0 502 76288 Jan 04 17:06 vsftpd-1.1.0-1.i386.rpm
226 Directory send OK.
ftp> get vsftpd-1.1.0-1.i386.rpm vsftpd-1.1.0-1.i386.rpm.tmp
local: vsftpd-1.1.0-1.i386.rpm.tmp remote: vsftpd-1.1.0-
1.i386.rpm
227 Entering Passive Mode (192,168,1,100,44,156)
150 Opening BINARY mode data connection for vsftpd-1.1.0-
1.i386.rpm (76288 bytes).
226 File send OK.
76288 bytes received in 0.499 secs (1.5e+02 Kbytes/sec)
ftp> exit
221 Goodbye.
[root@smallfry tmp]#
```

As expected, we can't do anonymous ftp.

```
[root@smallfry tmp]# ftp 192.168.1.100
Connected to 192.168.1.100 (192.168.1.100).
220 ready, dude (vsFTPd 1.1.0: beat me, break me)
Name (192.168.1.100:root): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp> quit
221 Goodbye.
[root@smallfry tmp]#
```

Chapter 4

Telnet, TFTP and XINETD

===

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

===

Xinetd is a program used to start and stop a variety of Linux data communication applications. Some of these applications, such as Telnet, are installed by default in RedHat Linux, others such as TFTP, need to be installed afterwards.

Xinetd is pre-installed in newer RedHat versions and is configured to startup automatically at boot time. The configuration files for each of the network programs xinetd controls are located in the **/etc/xinetd.d** directory. Once you have edited the configuration files you'll have to restart xinetd to make the configurations take effect.

```
[root@bigboy tmp]# /etc/init.d/xinetd restart
Stopping xinetd: [ OK ]
Starting xinetd: [ OK ]
[root@bigboy tmp]#
```

Remember that you will also be restarting all the other xinetd controlled processes. If you are logged into your Linux box via Telnet and then restart xinetd you will disconnect your self from your Linux server as Telnet would be restarted too.

Telnet

What is Telnet?

Telnet is a program that allows users to log into your server and get a command prompt just as if they were logged into the VGA console. Telnet is installed and enabled by default on RedHat Linux.

One of the disadvantages of Telnet is that the data is sent as clear text. This means that it is possible for someone to use a network analyzer to peek into your data packets and see your username and password. A more secure method for remote logins would be via Secure Shell (SSH) which uses varying degrees of encryption.

The command to do remote logins via telnet from the command line is simple. You enter the word "telnet" and then the IP address or server name to which you want to connect.

Here is an example of someone logging into a remote server named "smallfry" from server "bigboy". The user looks at the routing table and then logs out.

```
[root@bigboy root]# telnet 192.168.1.105
Trying 192.168.1.105...
Connected to 192.168.1.105.
Escape character is '^]'.

Linux 2.4.18-14 (smallfry.my-site.com) (10:35 on Sunday, 05
January 2003)

Login: peter
Password:
Last login: Fri Nov 22 23:29:44 on ttyS0
You have new mail.
[peter@smallfry peter]$
[peter@smallfry peter]$ netstat -nr
Kernel IP routing table
Destination     Gateway       Genmask         Flags  MSS Window irtt
Iface
255.255.255.255 0.0.0.0       255.255.255.255
UH     40  0      0    wlan0
192.168.1.0     0.0.0.0       255.255.255.0   U      40  0      0    w
lan0
127.0.0.0       0.0.0.0       255.0.0.0       U      40  0      0    l
o
0.0.0.0         192.168.1.1   0.0.0.0         UG     40  0      0    w
lan0
[peter@smallfry peter]$ exit
logout

Connection closed by foreign host.
[root@bigboy root]#
```

Setting Up A Telnet Server

By default Telnet is installed enabled on RedHat Linux. You can test whether the Telnet process is running with the following command which is used to check the TCP/UDP ports on which your server is listening:

```
[root@bigboy root]# netstat -a | grep telnet
tcp        0        0        *:telnet        *:*        LISTEN
[root@bigboy root]#
```

If you want to *disable* Telnet then edit the file **/etc/xinetd.d/telnet** and set the **disable** parameter to "yes".

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
# unencrypted username/password pairs for authentication.
service telnet
```

```
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes
}
```

You'll then have to restart xinetd for the new settings to take effect.

```
[root@bigboy tmp]# /etc/init.d/xinetd restart
Stopping xinetd: [ OK ]
Starting xinetd: [ OK ]
[root@bigboy tmp]#
```

TFTP

What is TFTP?

Cisco and other networking equipment manufacturers allow you to backup live configurations from routers and switches to workstations via the TFTP protocol. A remotely stored configuration file is always good to have. The reverse is also true, configurations can be loaded from the server to the network device. A common use of this reverse TFTP is the application of access control lists (ACLs) and even passwords from a centralized file.

Setting up a TFTP server

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail. The latest TFTP server version of the RPM for RedHat 9.0 is tftp-server0.32-4. Here are the steps to setting up the software:

Install the package using the following command:

```
[root@bigboy tmp]# rpm -Uvh tftp-server-0.32-4.i386.rpm
```

Edit the file **/etc/xinetd.d/tftp** and set disable to "no".

```
# default: off
# description: The tftp server serves files using the trivial
# file transfer \
# protocol. The tftp protocol is often used to boot diskless \
# workstations, download configuration files to
# network-aware printers, \
# and to start the installation process for some operating
```

```
systems.
service tftp
{
  socket_type = dgram
  protocol = udp
  wait = yes
  user = root
  only_from = 192.168.1.1
  server = /usr/sbin/in.tftpd
  server_args = -s /tftpboot
  disable = no
  per_source = 11
  cps = 100 2
}
```

In this example, xinetd will only allow the TFTP server to accept connections from the router / switch / firewall with an address of 192.168.1.1. You can extend this list with commas in between or just comment it out al together for global access.

Create a /tftpboot directory with global read write privileges

```
[root@bigboy tmp]# chmod 777 /tftpboot
```

Restart xinetd

```
[root@bigboy tmp]# /etc/init.d/xinetd restart
Stopping xinetd: [ OK ]
Starting xinetd: [ OK ]
[root@bigboy tmp]#
```

Each device must have a configuration file in the **/tftpboot** directory. Here's an example of what to do for a SOHO firewall with the name "pixfw" and a configuration filename that matches cisco's standard naming scheme of *"devicename"-config*

```
[root@bigboy tmp]# touch /tftpboot/pixfw-config
[root@bigboy tmp]# chmod 666 /tftpboot/pixfw-config
[root@bigboy tmp]# ll /tftpboot/
total 1631
-rw-rw-rw- 1 root root 3011 Oct 29 14:09 pixfw-config
[root@bigboy tmp]#
```

Configuring Cisco Devices for TFTP

You'll now have to configure your Cisco router / firewall to use the TFTP server. The following examples assume that the TFTP server's IP address is 192.168.1.100

Cisco PIX firewall

- Log onto the device, get into enable mode and then enter the TFTP commands

```
pixfw> enable
Password: ********
pixfw# configure terminal
pixfw(config)# tftp-server inside 192.168.1.100 /pixfw-config
pixfw(config)# exit
```

- Save the configuration to non volatile memory

```
pixfw# write memory
Building configuration...
Cryptochecksum: 3af43873 d35d6f06 51f8c999 180c2342
[OK]
pixfw#
```

- Save the configuration to the TFTP server

```
pixfw# write network
Building configuration...
TFTP write '/pixfw-config' at 192.168.1.100 on interface 1
[OK]
pixfw#
```

Cisco Switch Running CATOS

- Log onto the device, get into enable mode and then enter the TFTP commands

```
ciscoswitch> (enable) wr net
This command shows non-default configurations only.
Use 'write network all' to show both default and non-default
configurations.
IP address or name of remote host? [192.168.1.100]
Name of configuration file?[ciscoswitch-config]
Upload configuration to ciscoswitch-config on 192.168.1.100
(y/n) [n]? y
.........
Finished network upload. (30907 bytes)
ciscoswitch> (enable)
```

Cisco Router

- Log onto the device, get into enable mode, then configure mode and then enter the TFTP commands

```
ciscorouter> enable
ciscorouter#write net
Remote host [192.168.1.100]? 192.168.1.100
Name of configuration file to write [ciscorouter-config]?
ciscorouter-config
Write file ciscorouter-config on host 192.168.1.100? [confirm]
y
ciscorouter# exit
```

Cisco CSS 111000 "Arrowpoints"

- Log onto the device and then enter the tftp commands

```
ciscocss# copy running-config tftp 192.168.1.100 ciscocss-
config
Working..(\) 100%
Connecting (/)
Completed successfully.
ciscocss# exit
```

Cisco Local Director

- Log onto the device, get into enable mode, then configure mode and then enter the TFTP commands

```
ciscold> ena
Password:
ciscold# write net 192.168.1.100 ciscold-config
Building configuration...

writing configuration to //ciscold-config on
192.168.1.100:69 ...
[OK]
ciscold# exit
```

Using TFTP To Restore Your Router Configuration

One of the benefits of having a TFTP server is that you can save your configuration files on a remote server's hard disk. This can be very useful in the event of a router failure after which you need to reconfigure the device from scratch. One of the simplest ways of doing this using TFTP is to:

Connect your router to the local network of the TFTP server

Give your router the bare minimum configuration that allows it to ping your TFTP server. (No access controls or routing protocols)

Use the copy command to copy the backup configuration from the TFTP server to your startup configuration in NVRAM.

Disconnect the router from the network

Reload the router without saving the live running configuration to overwrite the startup configuration. On rebooting, the router will copy the startup configuration stored in NVRAM into a clean running configuration environment

Log into the router via the console and verify the configuration is OK
Reconnect the router to the networks on which it was originally connected
Here are the commands:

```
ciscorouter> enable
Password: ********
ciscorouter# write erase
ciscorouter# copy tftp:file-name startup-config
ciscorouter# reload
```

Please be aware that the "write erase" command erases your NVRAM startup configuration and should always be used with great care.

Chapter 5

Secure Remote Logins And File Copying

==========

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

==========

Here is some information on how to both remotely log in and copy files using a secure encrypted connection with Secure Shell (SSH) and Secure Copy (SCP). Similar programs such as Telnet and FTP can be security hazards as they do not encrypt their passwords. SSH and SCP could be the right choice for you.

Using Secure Shell As A Replacement For Telnet

When you use Telnet to remotely log into a Linux box you run the risk of people being able to eavesdrop on your network wire to see your username and password as unencrypted text. None of the data flow in Telnet is encrypted, so all your activities can be monitored.

RedHat Linux comes standard with Secure Shell (SSH) installed. This provides an encrypted data stream for you to use when you log in from one machine to another. When logging in from another Linux/UNIX machine you use the "**ssh**" command. There are GUI based SSH clients available for Windows, see the references in the bibliography.

You can get SSH configured to start at boot by using the **chkconfig** command.

```
[root@bigboy tmp]# chkconfig --level 35 sshd on
```

You can also start/stop/restart SSH after booting by running the sshd initialization script.

```
[root@bigboy tmp]# /etc/init.d/sshd start
[root@bigboy tmp]# /etc/init.d/sshd stop
[root@bigboy tmp]# /etc/init.d/sshd restart
```

Remember to restart the SSH process every time you make a change to the configuration files for the changes to take effect on the running process.

Testing To See If SSH Is Running

You can test whether the SSH process is running with the following command. You should get a response of plain old process ID numbers:

```
[root@bigboy tmp]# pgrep sshd
```

The etc/ssh/sshd_config File

The SSH configuration file is called **/etc/ssh/sshd_config**. By default SSH listens on all your NICs and uses TCP port 22. See the configuration snippet below:

```
# The strategy used for options in the default sshd_config shipped
with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options change a
# default value.

#Port 22
#Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::
```

If you are afraid of people trying to hack in on a well known TCP port, then you can change port 22 to something else that won't interfere with other applications on your system, such as port 435

First make sure your system isn't listening on port 435, using the "**netstat**" command and using "**grep**" to filter out everything that doesn't have the string "435".

```
[root@bigboy root]# netstat -an | grep 435
[root@bigboy root]#
```

No response, OK. Change the Port line in **/etc/ssh/sshd_config** to mention 435 and remove the "#" at the beginning of the line. If port 435 is being used, pick another port and try again.

```
Port 435
```

Restart SSH

```
[root@bigboy tmp]# /etc/init.d/sshd restart
```

Check to ensure SSH is running on the new port

```
[root@bigboy root]# netstat -an | grep 435
tcp    0    0    192.168.1.100:435    0.0.0.0:*    LISTEN
[root@bigboy root]#
```

Using SSH To Login To A Remote Machine

Using SSH is similar to Telnet. To login from another Linux box use the "ssh" command with a **"-l"** to specify the username you wish to login as. If you leave out the **"-l"**, your username will not change. Here are some examples for a server named "**smallfry**" in your **/etc/hosts** file.

User "root" Logs In To smallfry As User "root"

```
[root@bigboy tmp]# ssh smallfry
```

User "root" Logs In To smallfry As User "peter"

The examples below assume that you have created a user called "**peter**" on **smallfry**.

Using default port 22

```
[root@bigboy tmp]# ssh -l peter smallfry
```

Using port 435

```
[root@bigboy tmp]# ssh -l peter -p 435 smallfry
```

What You Should Expect To See When You Log In

The first time you log in, you will get a warning message saying that the remote host doesn't know about your machine. Something like this:

```
[root@bigboy tmp]# ssh smallfry
Host key not found from the list of known hosts.
!! If host key is new or changed, ssh1 protocol is vulnerable to an
!! attack known as false-split, which makes it relativily easy to
!! hijack the connection without the attack being detected. It is
!! highly advisable to turn StrictHostKeyChecking to "yes" and
!! manually copy host keys to known_hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'smallfry' added to the list of known hosts.
root@smallfry's password:
Last login: Thu Nov 14 10:18:45 2002 from 192.168.1.98
No mail.
[root@smallfry tmp]#
```

Deactivating Telnet once SSH is installed

Now you need to switch off Telnet. The Telnet server is controlled by the xinetd network security program. Xinetd is installed by default in RedHat 7.3 and newer. The configuration files for each of the network programs it controls is located in the **/etc/xinetd.d** directory. Edit the file **/etc/xinetd.d/telnet** and set the **disable** parameter to "yes".

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
# unencrypted username/password pairs for authentication.
service telnet
{

    flags = REUSE
    socket_type = stream
    wait = no
    user = root
    server = /usr/sbin/in.telnetd
    log_on_failure += USERID
    disable = yes

}
```

Now restart xinetd.

```
[root@bigboy tmp]# /etc/init.d/xinetd restart
Stopping xinetd: [ OK ]
Starting xinetd: [ OK ]
[root@bigboy tmp]#
```

Using SCP as a more secure replacement for FTP

From a networking perspective, FTP isn't very secure as usernames, passwords and data are sent across the network unencrypted. More secure forms such as SFTP (Secure FTP) and SCP (Secure Copy) are available as a part of the Secure Shell package that is normally installed by default on RedHat. There is a windows scp client called WinSCP which can be downloaded at:

http://winscp.vse.cz/eng/

Secure Copy (SCP) is installed in parallel with SSH and they always run simultaneously on the same TCP port. SCP doesn't support anonymous downloads like FTP.

Copying Files To The Local Linux Box

Command Format:

```
scp username@address:remotefile localdir
```

Examples:

Copy file **/tmp/software.rpm** on the remote machine to the local directory **/usr/rpm**

```
[root@bigboy tmp]# scp root@smallfry:/tmp/software.rpm /usr/rpm
```

Copy file **/tmp/software.rpm** on the remote machine to the local directory **/usr/rpm** using TCP port 435

```
[root@bigboy tmp]# scp -P 435 root@smallfry:/tmp/software.rpm /usr/rpm
```

Copying Files To The Remote Linux Box

Command Format:

```
scp filename username@address:remotedir
```

Examples:

Copy file **/etc/hosts** on the local machine to directory **/tmp** on the remote server.

```
[root@bigboy tmp]# scp /etc/hosts root@192.168.1.103:/tmp
```

Copy file **/etc/hosts** on the local machine to directory **/tmp** on the remote server using TCP port 435

```
[root@bigboy tmp]# scp -P 435 /etc/hosts
root@192.168.1.103:/tmp
```

Copying files using SCP without a password

From time to time you may want to write scripts that will allow you to copy files between servers without being prompted for passwords. This can make them simpler to write and also prevents you from having to embed the password in your code.

SCP has a feature which allows you to do this. You no longer have to worry about prying eyes seeing your passwords nor worrying about your script breaking when someone changes the password.

There are some security risks though. The feature is automatically applied to SSH as well. There is the possibility of someone using your account to login to the target server by entering the username alone. It is therefore best to implement this using unprivileged accounts on both the source and target servers.

In the case below, we'll be enabling this feature in one direction (from server "bigboy" to server "smallfry") and only using the unprivileged account called "filecopy".

Source Server Configuration

Here are the steps you need to do on the server from which you will be sending the files:

Generate your SSH encryption keypair for the "filecopy" account. Hit the enter key each time you are prompted for a password to be associated with the keys. (ie. Do NOT enter a password.)

```
[filecopy@bigboy filecopy]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key
(/filecopy/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/filecopy/.ssh/id_dsa.
Your public key has been saved in
/filecopy/.ssh/id_dsa.pub.
The key fingerprint is:
1e:73:59:96:25:93:3f:8b:50:39:81:9e:e3:4a:a8:aa
filecopy@bigboy
[filecopy@bigboy filecopy]#
```

These keyfiles are stored in the .ssh subdirectory of your home directory. View the contents of that directory. The file named "id_dsa" is your private key, and "id_dsa.pub" is the public key which you will be sharing with your target server. Non RedHat versions may use different filenames, use the SSH man pages to verify this.

```
[filecopy@bigboy filecopy]# cd .ssh
[filecopy@bigboy filecopy]# ls
id_dsa  id_dsa.pub  known_hosts
[filecopy@bigboy .ssh]#
```

Copy the ONLY public key to the home directory of the account to which you will be sending the file.

```
[filecopy@bigboy .ssh]# scp id_dsa.pub
filecopy@smallfry:.ssh/public-key.tmp
```

Target Server Configuration

Here are the steps you need to do on the server which will receive the file

Now log into smallfry and go to the ssh directory

```
[filecopy@smallfry home]# cd .ssh
```

Append the public-key.tmpfile to the end of the authorized_keys file. The authorized_keys file contains a listing of all the public keys from machines that are allowed to connect to your smallfry account without a password. Non RedHat versions may use different filenames, use the SSH man pages to verify this.

```
[filecopy@smallfry .ssh]# cat ~/.ssh/public-key.tmp>>
authorized_keys
```

From now on you can ssh and scp as user "filecopy" from server bigboy to smallfry without being prompted for a password.

Chapter 6

Configuring DNS

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

You can make your Linux box into your home network's DNS nameserver, here's how.

What Is DNS?

As explained on the introduction to networking concepts chapter, the Domain Name System (DNS) is the way in which a URL or domain like www.linuxhomenetworking.com is converted to an IP address.

What Is BIND?

BIND is an acronym for the "Berkeley Internet Name Domain" project which maintains the DNS related software suite that runs under Linux. The most well known program in BIND is "named", the daemon that responds to DNS queries from remote machines.

When To Use A DNS Caching Nameserver

DNS caching servers should be used by the machines on your network to provide DNS information that it has learned from the authoritative DNS servers of the Internet. Caching DNS servers then store (or cache), the most frequently requested information to reduce the lookup overhead of subsequent queries. If you want to advertise your website www.my-site.com to the

rest of the world, then a regular DNS server is what you require. Setting up a caching DNS server is fairly straightforward and will work whether or not your ISP provides you with a static or dynamic Internet IP address.

Once you have set up your caching DNS server you will then have to configure each of your home network PCs to use it as their DNS server. If your home PCs get their IP addresses using DHCP, then you will have to configure your DHCP server to make it aware of the IP address of your new DNS server. Off the shelf router/firewall appliances used in most home networks will usually act as both the caching DNS and DHCP server. In this case a separate DNS server is unnecessary.

When To Use A Regular DNS Server

If you host your own website at home with full control of all the web domains and your ISP provides you with a "fixed" or "static" IP address, then a regular DNS server would be the way to go. A caching DNS nameserver is only used as a reference, regular nameservers are used as the authoritative source of information.

Note: Regular nameservers are also caching nameservers by default.

When To Use Dynamic DNS

Your DSL ISP will assign IP addresses to your home either with an unchanging, "fixed" or "static" IP address or via a changing "DHCP" method. If your router/firewall is getting its Internet IP address using DHCP then you must consider dynamic DNS. This chapter assumes that you are using "static" Internet IP addresses.

How To Download and Install The BIND Packages

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail.

As of this writing the latest version of the BIND suite for RedHat 9.0 was version 9.2.1-16. It comes standard with the RedHat installation CDs.

```
[root@bigboy tmp]# rpm -Uvh bind-9.2.1-16.i386.rpm
```

How To Get BIND Started

You can use the chkconfig command to get BIND configured to start at boot:

```
[root@bigboy tmp]# chkconfig --level 35 named on
```

To start/stop/restart BIND after booting

```
[root@bigboy tmp]# /etc/init.d/named start
[root@bigboy tmp]# /etc/init.d/named stop
[root@bigboy tmp]# /etc/init.d/named restart
```

Remember to restart the BIND process every time you make a change to the conf file for the changes to take effect on the running process.

The /etc/resolv.conf File

This file is used by DNS clients (servers not running BIND) to determine both the location of their DNS server and the domains to which they belong. It generally has two columns, the first contains a keyword and the second contains the desired value(s) separated by commas. Here is a list of keywords:

Keyword	Value
Nameserver	IP address of your DNS nameserver. There should be only one entry per “nameserver” keyword. If there is more than one nameserver, you’ll need to have multiple “nameserver” lines.
Domain	The local domain name to be used by default. If the server is bigboy.my-site.com, then the entry would just be my-site.com
Search	If you refer to another server just by its name without the domain added on, DNS on your client will append the server name to each domain in this list and do an nslookup on each to get the remote servers’ IP address. This is a handy time saving feature to have so that you can refer to servers in the same domain by only their servername without having to specify the domain. The domains in this list must separated by spaces.

Here is a sample configuration in which:

The client server’s main domain is my-site.com, but it also is a member of domains my-site.net and my-site.org which should be searched for short hand references to other servers.

Two nameservers, 192.168.1.100 and 192.168.1.102 provide DNS name resolution.

```
domain my-site.com
search my-site.com net my-site.net my-site.org
nameserver 192.168.1.100
nameserver 192.168.1.102
```

Configuring A Caching Nameserver

The RedHat default installation of BIND is configured to convert your Linux box into a caching nameserver. The only file you have to edit is **/etc/resolv.conf** in which you’ll have to comment out the reference to your previous DNS server (most likely your router) with a "**#**" or make it point to the server itself using the universal localhost IP address of 127.0.0.1

Old Entry

```
nameserver 192.168.1.1
```

New Entry

```
# nameserver 192.168.1.1
```

or:

```
nameserver 127.0.0.1
```

The next step is to make all the other machines on your network point to the caching DNS server as their primary DNS server.

Configuring A Regular Nameserver

For the purposes of this tutorial, the subnet that has been assigned to you by your ISP is 97.158.253.24 with a subnet mask of 255.255.255.248 (/29).

Configuring named.conf

The main DNS configuration is kept in the file **/etc/named.conf** which is used to tell BIND where to find the configuration files for each domain you own. There are usually two zone areas in this file:

Forward zone file definitions which list files to map domains to IP addresses

Reverse zone file definitions which list files to map IP addresses to domains

In this example the forward zone for www.my-site.com is being set up by placing the following entries at the bottom of the **/etc/named.conf** file. The zone file is named my-site.zone and, though not explicitly stated, the file my-site.zone should be located in the default directory of **/var/named**.

```
zone "my-site.com" {

  type master;
  notify no;
  allow-query { any; };
  file "my-site.zone";

};
```

You can also insert additional entries in the **/etc/named.conf** file to reference other web domains you host. Here is an example for my-other-site.com using a zone file named **my-other-site.zone**.

```
zone "my-other-site.com" {

  type master;
  notify no;
```

```
  allow-query { any; };
  file "my-other-site.zone";

};
```

The reverse zone definition below is optional for a home / SOHO DSL based web site. It just makes you able to do an **nslookup** query on the 97.158.253.x IP address and get back the true name of the server assigned that IP address. This is rarely done for home based sites. It is especially difficult to do this with your DSL ISP if you have less than 256 static IP addresses (also known as a "Class C" block of addresses).

Note: the reverse order of the IP address in the zone section is important.

```
zone "253.158.97.in-addr.arpa" {
  type master;
  notify no;
  file "253.158.97";
};
```

Configuring The Zone Files

In all zone files, you can place a comment at the end of any line by inserting a semi-colon **";"** character then typing in the text of your comment.

By default, your zone files are located in the directory **/var/named**.

Each zone file contains a variety of records (eg. SOA, NS, MX, A and CNAME) which govern different areas of BIND. I'll explain of them below and then follow it all up with an example.

The SOA Record

The very first record is the Start of Authority (SOA) record which contains general administrative and control information about the domain. Though you would normally think a record would be a single line, the SOA format spans several. It is the most counter-intuitive of them all, the rest of the records are relatively straight forward.

SOA Record Format

Line #	Column #	Name	Description
1	1, 2, 3	@ IN SOA	Signifies that the SOA record is about to begin
	4	Nameserver	Fully qualified name of your primary nameserver. Must be followed by a "."
	5	Email	The email address of the nameserver administrator. The regular "@" in the e-mail address must be replaced with a "." instead. The email address must also be followed by a "."
	6	" ("	Signifies that we're about to define some performance related variables.
2	1	Serial	A serial number for the current configuration. Usually in the date format YYYYMMDD with single digit incremented number tagged to the end.
3	1	Refresh	Tells the slave DNS server how often it should check the master DNS server. Slaves aren't really used in home / SOHO environments.
4	1	Retry	The slave's retry interval to connect the master in the event of a connection failure. Slaves aren't really used in home / SOHO environments.
5	1	Expire	Total amount of time a slave will retry to contact the master before expiring the data it contains. Slaves aren't really used in home / SOHO environments.
6	1	Minimum TTL	The amount of time external caching DNS servers should keep your DNS information before flushing the data from the cache. As of BIND version 9 this value is overridden by the **$TTL** command at the very top of the configuration file.
	2	") "	Signifies that we're all finished with the variables.

NS, MX, A And CNAME Records

Unlike the SOA record, the NS, MX, A and CNAME records each occupy a single line and the records each have a very similar layout.

NS, MX, A and CNAME Record Formats

Record	Description	First Column	Second Column	Third Column	Fourth Column
NS	Lists the name of the nameserver for the domain	Blank	"NS"	IP address or CNAME of the nameserver	N/A
MX	Lists the mail servers for your domain such as my-site.com	Domain, followed by a "."	"MX"	Mail server priority	CNAME of mail server or the mailserver's FQDN** followed by a "."
A	Maps an IP address to each server in your domain. There must always be an entry for localhost 127.0.0.1	Server name	"A"	IP address of server	N/A
CNAME	Provides additional alternate "alias" names for servers listed in the "A" records.	"alias" or "nickname" for server	"CNAME"	"A" record name for server	N/A

**The Fully Qualified Domain Name (FQDN) is the full DNS name of the server such as mail.my-site.com

Note: If you don't put a "." at the end of a host name in a SOA, NS, A or CNAME record, BIND will automatically tack on the domain name. So an "A" record with "www" will be assumed to refer to www.my-site.com. This may be OK in most cases, but if you forget to put the "." after the domain in the MX record for my-site.com, BIND will attach the my-site.com at the end, and you will find your mail server only accepting mail for the domain my-site.com.mysite.com.

Sample Forward Zone File

Here is a working example of the zone file for my-site.com.

```
;
; Zone file for my-site.com
;
; The full zone file
```

```
;
$TTL 3D
@       IN      SOA     www.my-site.com. hostmaster.my-site.com. (
                        200211152       ; serial#
                        3600            ; refresh, seconds
                        3600            ; retry, seconds
                        3600            ; expire, seconds
                        3600 )          ; minimum, seconds
;
                NS      www             ; Inet Address of
nameserver

my-site.com.    MX      10 mail         ; Primary Mail Exchanger

;
localhost       A       127.0.0.1
www             A       97.158.253.26
mail            CNAME   www
```

Notice that in this example:

Server www.my-site.com is nameserver for my-site.com. In corporate environments there may be a separate nameserver for this purpose. Primary nameservers are more commonly called “ns1” and secondary nameservers “ns2”, but in the home / SOHO environment it is not necessary to differentiate.

The minimum TTL is set to 3600 seconds, but the overriding **$TTL** value is 3 days. So remote DNS caching servers will store learned DNS information from your zone for 3 days before flushing it out of their caches.

The MX record for my-site.com points to the server named mail.my-site.com

"mail" is actually a CNAME or "alias" for the web server "www". So here we have an example of the nameserver, mail server and web server being the same machine. If they were all different machines, then you’d have an "A" record entry for each like the example below.

```
www             A       97.158.253.26
mail            A       97.158.253.134
ns              A       97.158.253.125
```

The serial number is extremely important. You MUST increment it after editing the file or else BIND will not apply the changes you made when you restart "named".

Sample Reverse Zone File

Now we need to make sure that we can do an nslookup query on all our home network’s PCs and get their correct IP addresses. This is very important if you are running a mail server on your network as sendmail typically will only relay mail from hosts whose IP addresses resolve correctly in DNS. Here is a sample reverse zone file for our network.

```
;
; Filename: 192-168-1.zone
;
; Zone file for 192.168.1.x
```

```
;
$TTL 3D
@       IN      SOA     www.my-site.com.  hostmaster.my-
site.com. (
                                200303301          ; serial number
                                8H                 ; refresh, seconds
                                2H                 ; retry, seconds
                                4W                 ; expire, seconds
                                1D )               ; minimum, seconds
;
                        NS      www                ; Nameserver
Address
;

100                     PTR     bigboy.my-site.com.
103                     PTR     smallfry.my-site.com.
102                     PTR     ochorios.my-site.com.
105                     PTR     reggae.my-site.com.

32                      PTR     dhcp-32.my-site.com.
33                      PTR     dhcp-33.my-site.com.
34                      PTR     dhcp-34.my-site.com.
35                      PTR     dhcp-35.my-site.com.
36                      PTR     dhcp-36.my-site.com.
```

Note: I have included entries for addresses 192.168.1.32 to 192.168.1.36 which are the addresses our DHCP server issues. SMTP mail relay wouldn't work for PCs that get their IP addresses via DHCP if these lines weren't included.

You may also want to create a reverse zone file for the public NAT IP addresses for your home network. Unfortunately ISP's won't usually delegate this ability for anyone with less than a "Class C" block of 256 IP addresses. Most home DSL sites wouldn't qualify.

What You Need To Know About NAT And DNS

The above examples assume that the queries will be coming from the Internet with the zone files returning information related to the external 97.158.253.26 address of the webserver.

What do the PCs on your home network need to see? They need to see DNS references to the real IP address of the webserver, 192.168.1.100. This is because NAT won't work properly if a PC on your home network attempts to connect to the external 97.158.253.26 NAT IP address of your webserver.

Don't worry. BIND has a way around this called "views". The views feature allows you to force BIND to use pre-defined zone files for queries from certain subnets. This means it's possible to use one set of zone files for queries from the Internet and another set for queries from your home network.

Here's a summary of how it's done:

Place your zone statements in the **/etc/named.conf** file in one of two "views" sections. The first section will be called "internal" and will list the zone files to be used by your internal network. The second view called "external" will list the zone files to used for Internet users.

For example; you could have a reference to a zone file called **my-site.zone** for lookups related to the 97.158.253.X network which Internet users would see. This **/etc/named.conf** entry would be inserted in the “external” section. You could also have a file called **my-site-home.zone** for lookups by home users on the 192.168.1.0 network. This entry would be inserted in the “internal” section. The creation of the **my-site-home.zone** file is fairly easy. Just copy it form the **my-site.zone** file and replace all references to 97.158.253.X with references to 192.168.1.X

You must also tell the DNS server which addresses you feel are “internal” and “external”. This is done by first defining access control lists (ACLs) and then referring to these lists within each view section with the **match-clients** statement. There are some built-in ACLs:

“localhost” which refers to the DNS server itself;

“localnets” which refers to all the networks to which the DNS server is directly connected;

“any” which is self explanatory.

Note: You must place your “localhost”, “0.0.127.in-addr.arpa” and "." zone statements in the “internal” views section. Remember to increment your serial numbers!

Here is a sample configuration snippet for the **/etc/named.conf** file I use for my home network. All the statements below were inserted after the **“options”** and **“controls”** sections in the file.

```
// ACL statement

acl "trusted-subnet" { 192.168.17/24; };


view "internal" { // What the home network will see

    match-clients { localnets; localhost; "trusted-subnet"; };

        zone "." IN {
                type hint;
                file "named.ca";
        };

        zone "localhost" IN {
                type master;
                file "localhost.zone";
                allow-update { none; };
        };

        zone "0.0.127.in-addr.arpa" IN {
                type master;
                file "named.local";
                allow-update { none; };
        };

        zone "1.168.192.in-addr.arpa" IN {
                type master;
                file "192-168-1.zone";
```

```
                allow-update { none; };
        };

        zone "my-site.com" {
                type master;
                notify no;
                file "my-site-home.zone";
                allow-query { any; };
        };

        zone "my-other-site.com" {
                type master;
                notify no;
                file "my-other-site-home.zone";
                allow-query { any; };
        };

};


view "external" { // What the Internet will see

    match-clients { any; };
    recursion no;


        zone "my-site.com" {
                type master;
                notify no;
                file "my-site.zone";
                allow-query { any; };
        };


        zone "my-other-site.com" {
                type master;
                notify no;
                file "my-other-site.zone";
                allow-query { any; };
        };

};
```

Note: In the above example I included an ACL for network 192.168.17.0 /24 called “trusted-subnet” to help clarify the use of ACLs in more complex environments. Once the ACL was defined, I then inserted a reference to the “trusted-subnet” in the match-clients statement in the “internal” view. So in this case the local network (192.168.1.0 /24), the other trusted network (192.168.17.0) and localhost will get DNS data from the zone files in the “internal” view. Remember, this is purely an example. The home network we have been using doesn’t need to have the ACL statement at all as the built in ACLs “localnets” and “localhost” are sufficient. Our network won’t need the “trusted-subnet” section in the match-clients line either.

Loading Your New Configuration Files

Make sure your file permissions and ownership are OK in /var/named

```
[root@bigboy tmp]# cd /var/named
[root@bigboy named]# ll
total 6
-rw-r--r-- 1 named named 195  Jul 3 2001  localhost.zone
-rw-r--r-- 1 named named 2769 Jul 3 2001  named.ca
-rw-r--r-- 1 named named 433  Jul 3 2001  named.local
-rw-r--r-- 1 root  root  763  Oct 2 16:23 my-site.zone
[root@bigboy named]# chown named *
[root@bigboy named]# chgrp named *
[root@bigboy named]# ll
total 6
-rw-r--r-- 1 named named 195  Jul 3 2001  localhost.zone
-rw-r--r-- 1 named named 2769 Jul 3 2001  named.ca
-rw-r--r-- 1 named named 433  Jul 3 2001  named.local
-rw-r--r-- 1 named named 763  Oct 2 16:23 my-site.zone
[root@bigboy named]#
```

The configuration files above will not be loaded until you issue the following command to restart the named process that controls DNS (Make sure to increment your configuration file serial number before doing this):

```
[root@bigboy tmp]# /etc/init.d/named restart
```

Make Sure Your /etc/hosts File Is Correctly Updated

The chapter covering Linux networking topics explains how to do this. Some programs such as sendmail require a correctly configured **/etc/hosts** file even though DNS is correctly configured.

Configure Your Firewall

The sample network we're using assumes that the BIND nameserver and Apache web server software run on the same machine protected by a router/firewall. The actual IP address of the server is 192.168.1.100, which is a private IP address. You'll have to employ NAT in order for Internet users to be able to gain access to the server via the Public IP address we chose, namely 97.158.253.26. If your firewall is a Linux box, you may want to consider taking a look on the iptables chapter on how to do the NAT and allow DNS traffic trough to your nameserver.

Fix Your Domain Registration

Remember to edit your domain registration for "my-site.com", or whatever it is, so that at least one of the nameservers is your new nameserver. (97.158.253.26 in this case).

Domain registrars such as Verisign and RegisterFree usually provide a web interface to help you manage your domain.

Once, you've logged in with the registrar's username and password. You'll have to do the following two steps:

First, you'll have to create a new nameserver record entry for the IP address 97.158.253.26 to map to ns.my-site.com or www.my-site.com or whatever your nameserver is called. (This screen will prompt you for both the server's IP address and name)

Then you'll have to assign ns.my-site.com to handle your domain. (This screen will prompt you for the server name only)

Sometimes, the registrar will require at least two registered nameservers per domain. If you only have one, then you could either:

Create a second nameserver record entry with the same IP address, but different name.

Give your web server a second IP address using an IP alias, create a second NAT entry on your firewall and then create the second nameserver record entry with the new IP address, and different name.

It normally takes about 3-4 days for your updated DNS information to be propagated to all 13 of the world's root ("super duper") nameservers. You'll therefore have to wait about this amount of time before you'll start noticing people hitting your new website site.

How To Migrate Your Website In-House

It is important to have a detailed migration plan if you currently use an external company to host your website and wish to move the site to a server at home or in your office. At the very least it should include the following steps:

There is no magic bullet which will allow you to tell all the caching DNS servers in the world to flush their caches of your zone file entries. Your best alternative will be to request your existing service provider to set the TTL on my-site.com in the DNS zone file to a very low value, say 1 minute. As the TTL is usually set to 3 days, it will take at least 3-5 days for all remote DNS servers to recognize the change. Once the propagation is complete, it will take only 1 minute to see the results of the final DNS configuration switch to your new server. If anything goes wrong, you can then revert to the old configuration, knowing it will rapidly recover within minutes rather than days.

Set up your server in house using a different domain, for example www.my-site-test.com. Also set the TTL on this domain to 1 minute.

Ask your existing web hosting provider to add a DNS entry for your new server in the my-site.com domain. Now your server will be a part of both my-site.com and my-site-test.com.

Test your applications using server.my-site.com

Test mail to users @my-site-test.com

Test web traffic to www.my-site-test.com

Once testing is completed, convert all the server configuration files to reference my-site.com and not my-site-test.com Restart all the relevant applications.

Coordinate with your web hosting provider to simultaneously update you domain registration's DNS records to point to your new DNS server.

As both TTLs were set to 1 minute previously, you'll be able to see results of the migration within minutes.

Once complete, you can set the TTL back to 3 days to help reduce the volume of DNS query traffic hitting your DNS server.

Remember, you don't have to host DNS or mail in-house, this could be left in the hands of your service provider. You can migrate these services in-house later as your confidence in hosting becomes greater.

Finally, if you have concerns that your service provider won't co-operate then you could explain to them that you want to test their failover capabilities to a duplicate server that you host in-house. You can then decide whether the change will be permanent once you have failed over back and forth a few times.

DHCP Considerations For DNS

If you have a DHCP server on your network, you'll need to make it assign the IP address of the Linux box as the DNS server it tells the DHCP clients to use. If your Linux box is the DHCP server, then you may need to refer to the DHCP server chapter.

Chapter 7

Dynamic DNS

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

What Is DNS?

As explained on the introduction to networking chapter, DNS is the way in which a URL or domain like www.linuxhomenetworking.com is converted to an IP address.

If you want to host a website at home you have two DNS options:

Static DNS: This is used when your ISP provides you with unchanging “fixed” or “static” Internet IP addresses. Your DNS server acts as the authoritative source of information for your my-site.com domain. You can consider static DNS as the “traditional” or “regular” form of DNS.

Dynamic DNS: Used when you get a changing “dynamic” Internet IP addresses via DHCP from your ISP. You will have to use the services of a third party DNS provider to provide DNS information for your my-site.com domain.

What Is Dynamic DNS?

In many home networking environments, the DSL IP address is provided by DHCP and therefore changes from time to time. Dynamic DNS (DDNS) allows you to host a website such as www.my-site.com in which the IP address is dynamically assigned.

Before considering using a dynamic DNS solution for hosting a website at home with dynamic IPs:

- you must make sure your DSL provider will allow inbound connections, specifically HTTP, or else it will not work
- be prepared for slower response times for your home based site than if you were using a static IP and a regular DNS service.

register your domain name and read your DDNS provider's instructions on how to use their name servers.

DDNS works by having webmasters register their DDNS sites on the DDNS provider's servers. The web masters then register their domains with companies such as Verisign and RegisterFree and tells these registrars to direct queries to www.my-site.com to the servers of the DDNS provider.

The webserver itself then has a DDNS client program running that updates the DDNS providers name servers with the most current DHCP IP address of the site.

This chapter describes how to configure the most popular Linux based DDNS software ez-ipupdate and DDclient in the following two configurations:

on a Linux box directly connected to the Internet

on a Linux box when protected by a NAT router / firewall

Remember that unlike DSL, most cable modem providers may not allow you to host sites at home. dynDNS.org offers a service to overcome this limitation.

Dynamic DNS And NAT Router/Firewalls

As discussed in the introduction to networking chapter, in order to conserve the limited number of IP addresses available for internet purposes, most home router / firewalls will use Network Address translation (NAT) to map a single public DHCP obtained IP addresses to the many private IP addresses within your network.

NAT can fool the operation of some DDNS client software. In these cases, the software can only report the true IP address of the Linux box's NIC interface. If the Linux box is being protected behind a NAT router / firewall then the NIC will report in its data stream to the DDNS provider a private IP address which no one can reach directly via the Internet. The reported value is therefore invalid.

Some DDNS providers use more intelligent clients such as DDclient which can be configured to let the DDNS provider record the public IP address from which the data stream is originating. Once this is done, you'll have to also configure your router / firewall to do port forwarding to make all HTTP traffic destined for the IP address of the router / firewall to be exclusively NAT-ed and forwarded to a single server on your home network. An example of port forwarding with a Cisco PIX firewall is given in both the Cisco PIX firewall chapter and Net-Filter chapters.

Dynamic DNS Prerequisites

Sign Up With A DDNS Provider

First you'll have to register with a DDNS provider, some of which are listed on the Bibliography. This chapter focuses on the services of miniDNS and DynDNS.org. Most DDNS providers assume you are going to create a sub domain of their main domain. For example miniDNS.net will default to a domain such as machine-name.minidns.net.

If you want to create your own domain such as my-site.com, you'll have to do a little extra work. You'll have to register your domain with a DNS registrar such as www.registerfree.com or www.verisign.com. The cost is about US$20 per year.

The miniDNS registration for your own domain requires you to use the "add DNS Record" link on the registration page to create your own domain.

With dynDNS.org you'll have to go with their paid service to get a customized domain name. They call it Custom DNS and it doesn't support ez-ipupdate, you'll need DDclient in this case.

First you add your domain such as my-site.com. Then you must add a host record. You can give your machine's name or you can name the machine "www" to create a combined domain-subdomain of www.my-site.com which would be more intuitive to use.

Update Your DNS Registration

If you have your own domain, you'll have to return to www.registerfree.com or www.verisign.com and update the nameserver entries for your domain to point to the name servers of your DDNS provider. DNS queries for my-site.com will eventually query RegisterFree or Verisign which will then refer the query to your DDNS providers name servers which will have the most current IP address of your site because of the DDNS client software you are running at your home site.

Installing And Using ez-ipupdate

Download the tar/gzip file to your server's **/tmp** directory from the ez-ipupdate site listed in the Bibliography. Use the following commands to extract the contents into a new subdirectory.

```
[root@bigboy]tmp]# gunzip zip-tar-filename
[root@bigboy]tmp]# tar -xvf tar-filename
[root@bigboy]tmp]# cd /tmp/filename
```

Follow the install instructions for doing the "make" or program compilation. The ez-ipupdate installation will put the executable file in **/usr/local/bin** and all the files in the **/tmp/filename** directory will become extraneous.

The /etc/ez-ipupdate.conf File

ez-ipupdate uses a configuration file named **/etc/ez-ipupdate.conf** in which you must specify:

Your registration username and password

The host name you have selected for your Linux box

The NIC interface which is connected to your DSL line.

Here is a sample:

```
service-type=justlinux
user=registration-username:registration-password
host=servername.my-site.com
interface=eth0
```

Note: The service-type line is specific to your dynamic DNS provider which will often provide a customized **/etc/ez-ipupdate.conf** file for you to use.

ez-ipupdate And NAT

The ez-ipupdate software runs as a daemon in memory continuously checking the IP address of your NIC. If your Linux server is protected behind a firewall using NAT then the IP address of the NIC won't match that of the public IP address of the firewall and DDNS won't work properly, you'll have to use a client like DDclient which doesn't have this limitation.

Installing And Using DDclient

Another highly used solution is DDclient. The developer of DDclient has recognized the limitations of using ez-ipupdate with NAT. DDclient has a simple "web" update mode which tells your DDNS provider to use the source IP address of the data stream used to update your DDNS record. In most Home / SOHO environments this will be the same as that of the firewalls external NAT IP address.

In cases where "web" mode doesn't work, the DDclient script can also log in and parse out the external IP address of the router. It then communicates this information to your dynamic DNS provider. DDclient claims to offers support for a wide variety of routers from different manufacturers.

Remember, some routers such as the netgear line may provide automatic DDNS service and you may not have to download the software.

Before installing DDclient, read the README file to give you an idea of what to do. Check the Bibliography for the DDclient URL. Here is an example of the steps used to install it.

```
[root@bigboy tmp]# gunzip ddclient.tar.gz
[root@bigboy tmp]# tar -xvf ddclient.tar
ddclient-3.6.2/
ddclient-3.6.2/COPYRIGHT
ddclient-3.6.2/COPYING
…
…
…
[root@bigboy tmp]# cd dd*
[root@bigboy ddclient-3.6.2]# ll
-rw-r--r-- 1 root root 18007 Jan 3 2002 COPYING
-rw-r--r-- 1 root root 869   Jan 3 2002 COPYRIGHT
…
…
…
```

```
 [root@bigboy ddclient-3.6.2]# cp sample-
etc_rc.d_init.d_ddclient.redhat /etc/rc.d/init.d/ddclient
[root@bigboy ddclient-3.6.2]# /sbin/chkconfig --add ddclient
[root@bigboy ddclient-3.6.2]# cp ddclient /usr/sbin/
```

The /etc/ddclient.conf File

DDclient uses a configuration file named **/etc/ddclient.conf** in which you must specify:

Your registration username and password

The host name you have selected for your Linux box. This is referenced on the line labeled “server”.

The NIC interface which is connected to your DSL line.

Here is a sample using interface eth0:

```
## dyndns.org custom addresses
##
## (supports variables: wildcard,mx,backupmx)
##
use=if, if=eth0 # via interfaces
login=your-login # default login
password=your-password # default password
custom=yes \
server=members.dyndns.org, \
protocol=dyndns2 \
your-domain.top-level,your-other-domain.top-level # Your domains
here
```

Before updating the file you can use DDclient with the "-query" option to tell you which is the best mode to use. Here is an example.

```
[root@bigboy ddclient-3.6.2]# ddclient -daemon=0 -query
use=if, if=lo address is 127.0.0.1
use=if, if=wlan0 address is 192.168.1.100
use=web, web=dyndns address is 97.158.253.26
[root@bigboy ddclient-3.6.2]#
```

In this case, the simple web mode provides an acceptable value for your external IP address. You can then configure your **/etc/ddclient.conf** file to use "web"

```
#use=if, if=eth0 # via interfaces
use=web # via web
```

Testing Your Dynamic DNS

You can test your dynamic DNS by:

Looking at the status page of your DNS provider and making sure the IP address that matches your "www" site is the same as your router / firewall's public IP address.

Using the nslookup www.my-site.com command from your Linux command prompt and see whether you are getting a valid response. If you failed to add your host record, you will get an error message like this:

```
[root@bigboy tmp]# nslookup www.my-site.com

Server: 127.0.0.1
Address: 127.0.0.1#53

** server can't find www.my-site.com: NXDOMAIN
```

Note: This error could also be due to the fact that your domain hasn't propagated fully throughout the Internet. You can test to make sure everything is OK by forcing NS lookup to query the nameservers directly. The example below queries the miniDNS name server:

```
[root@bigboy tmp]# nslookup
> server ns1.minidns.net
Default server: ns1.minidns.net
Address: 202.64.51.214#53
> www.my-site.com
Server: ns1.minidns.net
Address: 202.64.51.214#53

Name: www.my-site.com
Address: 12.235.194.96
>
```

Testing Port Forwarding

Remember to read the configuration manual of your router / firewall to activate port forwarding. Test it by asking a friend to access your web server by pointing their browser to the external IP address of your router / firewall.

Chapter 8

The Apache Web Server

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

This is page outlines how to create multiple websites using a single IP address for a basic home configuration. The Apache online documentation gets a little complicated.

Download and Install The Apache Package

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail. It is best to use the latest version of Apache.

For example, the RedHat 9.0 RPM as of this writing was:

```
httpd-2.0.40-21.5.rpm
```

Install the package using the **rpm** command

```
[root@bigboy tmp]# rpm -Uvh httpd-2.0.40-21.5.i386.rpm
```

By default, Apache expects its HTML files to be located in the **/var/www/html** directory

How To Get Apache Started

Use the chkconfig command to configure Apache to start at boot:

```
[root@bigboy tmp]# chkconfig --level 35 httpd on
```

Use the httpd init script in the **/etc/init.d** directory to start/stop/restart Apache after booting

```
[root@bigboy tmp]# /etc/init.d/httpd start
[root@bigboy tmp]# /etc/init.d/httpd stop
[root@bigboy tmp]# /etc/init.d/httpd restart
```

You can test whether the Apache process is running with the following command, you should get a response of plain old process ID numbers:

```
[root@bigboy tmp]# pgrep httpd
```

Configuring DNS For Apache

Remember that you will never receive the correct traffic unless you have configured DNS for your domain to make your new Linux box web server the target of the DNS domain's www entry. See either the Static DNS or Dynamic DNS pages on how to do this.

General Configuration Steps

The configuration file used by Apache is **/etc/httpd/conf/httpd.conf**. Examples of this will follow.

Named Virtual Hosting

You can make your web server host more than one site per IP address by using Apache's "named virtual hosting" feature. The **NameVirtualHost** directive in the **/etc/httpd/conf/httpd.conf** file is used to tell Apache the IP addresses which will participate in this feature. Here is the format:

```
NameVirtualHost 97.158.253.26
```

The **<VirtualHost>** sections in the file then tell Apache where it should look for the web pages used on each web site. You must specify the IP address for which each **<VirtualHost>** section applies. Here is the format:

```
<VirtualHost 97.158.253.26>
    Directives for site #1
</VirtualHost>
```

```
<VirtualHost 97.158.253.26>
    Directives for site #2
</VirtualHost>
```

Within each **<VirtualHost>** section you then specify the primary website domain name for that IP address with the **ServerName** directive. The directory where the index page for that site is located is defined with the **DocumentRoot** directive.

You can also list secondary domain names which will serve the same content as the primary **ServerName** using the **ServerAlias** directive.

As explained on the apache website: "When a request arrives, the server will first check if it is using an IP address that matches the **NameVirtualHost**. If it is, then it will look at each **<VirtualHost>** section with a matching IP address and try to find one where the **ServerName** or **ServerAlias** matches the requested hostname. If it finds one, then it uses the configuration for that server. If no matching virtual host is found, then **the first listed virtual host** that matches the IP address will be used."

IP Based Virtual Hosting

The other virtual hosting option is to have one IP address per website which is also known as IP based virtual hosting. In this case you will not have a **NameVirtualHost** directive for the IP address, and you must only have a single **<VirtualHost>** section per IP address.

A Note On Virtual Hosting And SSL

It is common for system administrators to replace the IP address in the **<VirtualHost>** and **NameVirtualHost** directives with the "*" (all IP addresses) wildcard character. This makes configuration easier.

If you installed Apache with support for secure HTTPS / SSL, which is used frequently in credit card and shopping cart web pages, then wild cards won't work. The Apache SSL module demands at least one explicit **<VirtualHost>** directive for IP based virtual hosting. When you use wild cards, Apache interprets it as an overlap of name based and IP based **<VirtualHost>** directives and will give errors like this because it can't make up its mind about which method to use:

```
Starting httpd: [Sat Oct 12 21:21:49 2002] [error] VirtualHost
_default_:443 -- mixing * ports and non-* ports with a
NameVirtualHost address is not supported, proceeding with
undefined results
```

If you try to load any webpage on your web server you'll also notice an error like this:

```
Bad request!

Your browser (or proxy) sent a request that this server could not
understand.
If you think this is a server error, please contact the webmaster
```

You have two options to overcome this problem.

Continue using wildcards and disable SSL.

Run Apache with more careful use of wildcards

Disabling SSL – (Not Recommended)

If you wish to host a basic home SOHO website in which secure connections for credit card payments are unnecessary then you have the option of disabling SSL altogether. This can be done by not loading all the modules from the **/etc/httpd/conf.d** directory. By default, all the modules in this directory are loaded with the following directive in the **/etc/httpd/conf/httpd.conf** file:

```
Include conf.d/*.conf
```

You can therefore do a listing of all the files in this directory and specifically load all except **ssl**. In this case we load only the **php** and **perl** modules.

```
Include conf.d/perl.conf
Include conf.d/php.conf
```

You will then have to restart Apache for the changes to take effect.

Use Wild Cards Sparingly

The other choice is not to use virtual hosting statements with wild cards. The only exception would be the very first **<VirtualHost>** directive which defines the web pages to be displayed when matches to the other **<VirtualHost>** directives cannot be found.

A Note on Home Pages or Index Pages

By default, Apache will search the **DocumentRoot** directory for an index or "home" page named **index.html**. So for example, if you have a **VirtualHost** site of www.my-site.com with a **DocumentRoot** directory of /home/www/site1/, Apache will display the contents of the file /home/www/site1/index.html when you enter http://www.my-site.com in your browser.

Some editors like Microsoft FrontPage will create files with an "**.htm**", not "**.html**" extension. This isn't usually a problem if all your HTML files have hyperlinks pointing to files ending in "**.htm**" as FrontPage does. The problem occurs with Apache not recognizing the topmost **index.htm** page. The easiest solution is to create a symbolic link ("shortcut" for Windows users) called **index.html** pointing to the file **index.htm**. This will then allow you to edit/copy the file index.htm with index.html being updated automatically. You'll almost never have to worry about index.html and Apache again!

```
[root@bigboy tmp]# cd /home/www/site1
[root@bigboy site1]# ln -s index.htm index.html
[root@bigboy site1]# ll index.*
-rw-rw-r--    1 root     home        48590 Jun 18 23:43 index.htm
lrwxrwxrwx    1 root     root            9 Jun 21 18:05 index.html
-> index.htm
[root@bigboy site1]#
```

The "`l`" at the very beginning of the index.html entry signifies a link and the "->" the link target.

Configuration – Multiple Sites And IP Addresses

What follows are snippets of the section of the **/etc/httpd/conf/httpd.conf** file you'll need to edit. In this scenario:

The systems administrator for the server has previously created DNS entries for www.my-site.com, my-site.com, www.my-cool-site.com, www.default-site.com and www.test-site.com to map to an IP address 97.158.253.26 on this web server. The domain www.my-other-site.com was also configured to point to alias IP address 97.158.253.27.

Traffic to www.my-site.com, my-site.com, www.my-cool-site.com must get content from sub-directory site2. Hitting these URLs will cause Apache to display the contents of file index.html in this directory.

Traffic to www.test-site.com must get content from sub-directory site3.

Named virtual hosting will be required for 97.158.253.26 as in this case we have a single IP address serving different content for a variety of domains. A **NameVirtualHost** directive for 97.158.253.26 is therefore required.

Traffic going to www.my-other-site.com will get content from directory site4.

There is no **ServerName** directive for www.default-site.com and so traffic going to this domain

All other domains pointing to this server that don't have a matching **ServerName** directive will get web pages from the directory defined in the very first **<VirtualHost>** section. In this case is directory site1. Site www.default-site.com falls in this category.

Web Hosting Scenario Summary

Domain	IP address	Directory	Type of Virtual Hosting
www.my-site.com my-site.com www.my-cool-site.com	97.158.253.26	Site2	Name Based
www.test-site.com	97.158.253.26	Site3	Name Based
www.my-other-site.com All other domains	97.158.253.27	Site1	IP Based
www.default-site.com	97.158.253.26	Site1	Name Based

A sample snippet or a working **httpd.conf** file is listed below. The statements listed would normally be found at the very bottom of the file where virtual hosting statements normally reside.

The last section of this configuration snippet has some additional statements to ensure read-only access to your web pages with the exception of web based forms using POSTs (pages with “submit” buttons). Remember to restart Apache every time you update the conf file for the changes to take effect on the running process.

```
ServerName localhost
NameVirtualHost 97.158.253.26

#
# Match a webpage directory with each website
#
<VirtualHost *>
   DocumentRoot /var/www/html/site1
</VirtualHost>

<VirtualHost 97.158.253.26>
   DocumentRoot /var/www/html/site2
   ServerName www.my-site.com
   ServerAlias my-site.com, www.my-cool-site.com
</VirtualHost>

<VirtualHost 97.158.253.26>
   DocumentRoot /var/www/html/site3
   ServerName www.test-site.com
</VirtualHost>

<VirtualHost 97.158.253.27>
   DocumentRoot /var/www/html/site4
   ServerName www.my-other-site.com
</VirtualHost>


#
# Make sure the directories specified above
# have restricted access to read-only.
#
<Directory "/var/www/html/*">
   Order allow,deny
   Allow from all

   AllowOverride FileInfo AuthConfig Limit
   Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
   <Limit GET POST OPTIONS>
         Order allow,deny
         Allow from all
   </Limit>

   <LimitExcept GET POST OPTIONS>
         Order deny,allow
         Deny from all
   </LimitExcept>

</Directory>
```

A Note On Virtual Hosting And DNS

You will have to configure your DNS server to point to the correct IP address used for each of the websites you host. The chapter on static DNS shows you how to configure multiple domains such as my-site.com and my-other-site.com on your DNS server.

Disabling Directory Listings

Be careful if you create subdirectories under your **DocumentRoot** directory. Apache will run as expected if you link to files in the subdirectory. Be careful if you have designed your site without index.html pages in each subdirectory.

Say for example we create a subdirectory named /home/www/site1/example under www.my-site.com's **DocumentRoot** of /home/www/site1/. Now we'll be able to view the contents of the file my-example.html in this subdirectory if we point our browser to:

```
http://www.my-site.com/example/my-example.html
```

If a curious surfer decides to see what the index page is for www.my-site.com/example, they would type the link:

```
http://www.my-site.com/example
```

Apache will list all the contents of the files in the "example" directory if it can't find the index.html file. You can disable the directory listing by using a "**-Indexes**" option in the <Directory> directive for the **DocumentRoot** like this:

```
<Directory "/home/www/*">
    …
    …
    …
    Options MultiViews -Indexes SymLinksIfOwnerMatch
    IncludesNoExec
```

Remember to restart Apache after the changes. Users attempting to access the nonexistent index page will now get a "403 Access denied" message.

Handling Missing Pages

You can tell Apache to display a pre-defined HTML file whenever a surfer attempts to access a non-index page that doesn't exist. You can place this statement in the httpd.conf file which will make Apache display the contents of missing.htm instead of a generic "404 file Not Found" message.

```
ErrorDocument 404 /missing.htm
```

Remember to put a file with this name in each DocumentRoot directory. You can see the missing.htm file I use by clicking the non-existent link below. You'll notice that this gives the same output as http://www.linuxhomenetworking.com/missing.htm.

http://www.linuxhomenetworking.com/bogus-file.htm

Using Data Compression On Web Pages

Apache also has the ability to dynamically compress static web pages into gzip format and then send the result to the remote web surfers' web browser. Most current web browsers support this format and will transparently uncompress the data and present it on the screen. This can significantly reduce bandwidth charges if you are paying for internet access by the megabyte.

First you need to load Apache version 2's **deflate** module in your **httpd.conf** file and then use **Location** directives to specify what type of files to compress. After making these modifications and restarting Apache you will be able to verify from your **/var/log/httpd/access_log** file that the sizes of the transmitted HTML pages has shrunk.

Here is a comparison of the file sizes in the Apache logs and the document directory, 78,350 bytes shrunk to 15,190 bytes, almost 80% compression.

Log File

```
67.119.25.115 - - [15/Feb/2003:23:06:51 -0800] "GET /dns-static.htm
HTTP/1.1" 200 15190 "http://www.siliconvalleyccie.com/sendmail.htm"
"Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; AT&T CSM6.0;
YComp 5.0.2.6)"
```

Corresponding Directory Listing

```
[root@ bigboy tmp]# ll /web-dir/dns-static.htm
-rw-r--r--    1 user      group       78350 Feb 15 00:53
/home/www/ccie/dns-static.htm
[root@bigboy tmp]#
```

Compression Configuration Example

You can insert these statements just before your virtual hosting section of your **httpd.conf** file to activate the compression of static pages. Remember to restart Apache when you do.

```
LoadModule deflate_module modules/mod_deflate.so

<Location />

     # Insert filter
     SetOutputFilter DEFLATE

     # Netscape 4.x has some problems...
     BrowserMatch ^Mozilla/4 gzip-only-text/html

     # Netscape 4.06-4.08 have some more problems
     BrowserMatch ^Mozilla/4\.0[678] no-gzip

     # MSIE masquerades as Netscape, but it is fine
```

```
        BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

        # Don't compress images
        SetEnvIfNoCase Request_URI \
          \.(?:gif|jpe?g|png)$ no-gzip dont-vary


        # Make sure proxies don't deliver the wrong content
        Header append Vary User-Agent env=!dont-vary

    </Location>
```

Apache Running On A Server Behind A Firewall

If your webserver is behind a firewall, and you are logged on a machine behind the firewall as well, then you may find problems when trying to access www.mysite.com of www.my-other-site.com. The reason for this is that due to NAT (Network Address translation), firewalls frequently won't allow access from their protected network to IP addresses that they masquerade on the outside.

For example, in this case, Linux web server bigboy has an internal IP address of 192.168.1.100, but the firewall presents it to the world with an external IP address of 97.158.253.26 via NAT/masquerading. If you are on the inside, 192.168.1.X network, you may find it impossible to hit URLs that resolve in DNS to 97.158.253.26.

The solution to this can also be solved with virtual hosting. You can configure Apache to serve the correct content when accessing www.mysite.com or www.my-other-site.com from the outside, and also when accessing the specific IP address 192.168.1.100 from the inside. Fortunately Apache allows you to specify multiple IP addresses in the **<VirtualHost>** statements to help you overcome this problem. Here is an example:

```
NameVirtualHost 192.168.1.100
NameVirtualHost 97.158.253.26

<VirtualHost 192.168.1.100 97.158.253.26>
    DocumentRoot /www/server1
    ServerName www.my-site.com
    ServerAlias bigboy, www.my-site-192-168-1-100.com
</VirtualHost>
```

File Permissions And Apache

Remember that if you get a "permissions" error in your web browser after trying to browse your newly configured website, then you need to ensure that you allow "others" to have read access to the directory all the way from the root directory "/" to the target sub-directory.

The appendix has a short script that you can use to recursively set the file permissions in a directory to match those expected by Apache.

You may also have to use the "Directory" directive to make Apache serve the pages once the file permissions have been correctly set. If you have your files in the default **/var/www/html** directory then this second step becomes unnecessary.

How To Protect Web Page Directories With Passwords

You can password protect content in both the main and sub-directories of your **DocumentRoot** fairly easily. I know of cases where persons will allow normal access to their regular web pages, but require passwords for directories / pages that show MRTG or Webalizer data. In this example we'll show how to password protect the **/var/www/html directory**.

Apache has a password utility called "**htpasswd**" which can create "username password" combinations independent of your system login password for web page access. You have to specify the location of the password file, and if it doesn't yet exist, you'll have to include a "**-c**" or "create" switch on the command line. I recommend placing the file in your **/etc/httpd/conf** directory, away from the **DocumentRoot** tree where web users could possibly view it. Here is an example for a first user named "peter" and a second named "paul":

```
[root@bigboy tmp]# htpasswd -c /etc/httpd/conf/.htpasswd peter
New password:
Re-type new password:
Adding password for user peter
[root@bigboy tmp]#

[root@bigboy tmp]# htpasswd /etc/httpd/conf/.htpasswd paul
New password:
Re-type new password:
Adding password for user paul
[root@bigboy tmp]#
```

Make the **.htpasswd** file readable by all users.

```
[root@bigboy tmp]# chmod 644 /etc/httpd/conf/.htpasswd
```

Create a **.htaccess** file in the directory to which you want password control with the following entries. Remember this will password protect this directory and all its sub directories.

```
AuthUserFile /etc/httpd/conf/.htpasswd
AuthGroupFile /dev/null
AuthName EnterPassword
AuthType Basic
require user peter
```

- The AuthUserFile tells Apache to use the "**.htpasswd"** file
- The "require user" tells Apache that only user "peter" in the "**.htpasswd"** file should have access. If you wanted all "**.htpasswd"** users to have access then you'd replace this line with require valid-user

- "AuthType Basic" instructs Apache to accept basic unencrypted passwords from the remote users web browser.

Set the correct file protections on your new **.htaccess** file in the directory **/var/www/html**.

```
[root@bigboy tmp]# chmod 644 /var/www/html/.htaccess
```

Make sure your **/etc/httpd/conf/http.conf** file has an **AllowOverride** statement in a **<Directory>** directive for any directory in the tree **above** **/var/www/html**. In the example below, we want all directories below **/var/www/** to require password authorization.

```
<Directory /var/www/html/*>
        AllowOverride AuthConfig
</Directory>
```

You must also ensure that you have a **<VirtualHost>** directive that defines access to **/var/www/html** or another directory higher up in the tree.

```
<VirtualHost *>
     ServerName 97.158.253.26
     DocumentRoot /var/www/html
</VirtualHost>
```

Restart Apache. Try accessing the web site and you'll be prompted for a password.

Issues When Upgrading To Apache 2.0

Incompatible /etc/httpd/conf/http.conf files

Your old configuration files will be incompatible when upgrading from Apache version 1.3 to Apache 2.X. The new version 2.X default configuration file is stored in **/etc/httpd/conf/httpd.conf.rpmnew**. For the simple virtual hosting example above, it would be easiest to:

Save the old **httpd.conf** file with another name, **httpd.conf-version-1.x** for example. Copy the **ServerName**, **NameVirtualHost**, and **VirtualHost** sections from the old file and place them in the new file **httpd.conf.rpmnew**

Copy the **httpd.conf.rpmnew** file an name it **httpd.conf**

Restart Apache

Chapter 9

Configuring Linux Mail Servers

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

This chapter will help to show you how to set up a mail server for your home network. It covers Sendmail which is responsible for relaying your mail to a remote user's mailbox and also POP mail which is used to retrieve the mail from the mail box to your local PC via a mail client such as outlook Express.

Configuring Sendmail

An Overview Of How Sendmail Works

Sendmail is the most popular Linux program for processing mail. Once the mail arrives on the mail server it can be read in a number of ways:

Linux users logged into the mail server can read their mail directly using a text based client such as "mail" or a GUI client such as Evolution.

Windows users can use an email client such as "Outlook" or "Outlook Express" to download the mail to their local PC via POP. Windows users also have the option of either keeping or deleting the mail on the mail server after it has been downloaded.

The process is different when sending mail via the mail server:

If the mail is destined for a local user then sendmail will place the message in that person's mailbox so that they can retrieve it using one of the methods above.

If the mail isn't destined for the mailbox of a local user, then sendmail will attempt to relay it to the appropriate destination mail server via the Simple Mail Transport Protocol or SMTP. One of the main advantages of mail relaying is that when a PC user "A" sends mail to another user "B" on the Internet, the PC of user "A" can delegate the SMTP processing to the mail server.

Note: If mail relaying is not configured properly then your mail server could end up relaying SPAM. Simple sendmail security is outlined on this page.

Configuring DNS

Remember that you will never receive mail unless you have configured DNS for your domain to make your new Linux box mail server the target of the DNS domain's MX record. See either the Static DNS or Dynamic DNS pages on how to do this.

Installing And Starting Sendmail

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail.

It is best to use the latest version of sendmail as older versions have had a number of security holes. As of this writing the latest version of the sendmail suite was version 8.12.8-9.90. Install all the packages in this order:

```
[root@bigboy tmp]# rpm -Uvh sendmail-cf-8.12.8-9.90.i386.rpm
[root@bigboy tmp]# rpm -Uvh sendmail-8.12.8-9.90.i386.rpm
```

You'll also need to make sure the M4 macro package is also installed.

```
[root@bigboy tmp]# rpm -Uvh m4-1.4.1-13.i386.rpm
```

You can use the **chkconfig** command to get Sendmail configured to start at boot:

```
[root@bigboy tmp]# chkconfig --level 35 sendmail on
```

To start/stop/restart sendmail after booting

```
[root@bigboy tmp]# /etc/init.d/sendmail start
[root@bigboy tmp]# /etc/init.d/sendmail stop
[root@bigboy tmp]# /etc/init.d/sendmail restart
```

Remember to restart the sendmail process every time you make a change to the configuration files for the changes to take effect on the running process. You can also test whether the sendmail process is running with the pgrep command, you should get a response of plain old process ID numbers:

```
[root@bigboy tmp]# pgrep sendmail
```

Restart Sendmail After Editing Your Configuration Files

In this chapter we'll see that Sendmail uses a variety of configuration files which require different treatments in order for their commands to take effect. This little script encapsulates all the required post configuration steps.

```
#!/bin/bash
cd /etc/mail
make
m4 /etc/mail/sendmail.mc > /etc/sendmail.cf         # RH Ver 7.3-
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf    # RH Ver 8.0+
newaliases
/etc/init.d/sendmail restart
```

Use this command to make the script executable.

```
chmod 700 filename
```

You'll need to run the script each time you change any of the sendmail configuration files described in the sections to follow.

The line in the script that restarts sendmail is only needed if you have made changes to the **/etc/mail/sendmail.mc** file, but it has been included so that you don't forget. This may not be a good idea in a production system. Delete the appropriate "m4" line depending on your version of RedHat.

Both the **newaliases** and **m4** commands depend on the sendmail-cf RPM package. This must be installed, if not, you'll get errors like this when running the script:

Errors With The Newaliases Command

```
[root@bigboy mail]# newaliases
Warning: .cf file is out of date: sendmail 8.12.5 supports
version 10, .cf file is version 0
No local mailer defined
QueueDirectory (Q) option must be set
[root@bigboy mail]#
```

Errors With The m4 Command

```
[root@bigboy mail]# m4 /etc/mail/sendmail.mc >
/etc/mail/sendmail.cf
/etc/mail/sendmail.mc:8: m4: Cannot open /usr/share/sendmail-
cf/m4/cf.m4: No such file or directory
[root@bigboy mail]#
```

Errors When Restarting sendmail

```
[root@bigboy mail]# /etc/init.d/sendmail restart
Shutting down sendmail: [ OK ]
Shutting down sm-client: [FAILED]
Starting sendmail: 554 5.0.0 No local mailer defined
554 5.0.0 QueueDirectory (Q) option must be set
[FAILED]
Starting sm-client: [ OK ]
[root@bigboy mail]#
```

The /var/log/maillog File

Sendmail throws all its status messages in the **/var/log/maillog** file. It is always good to monitor this file whenever you are doing changes. Open two telnet, SSH or console windows. Work in one of them and monitor the sendmail status output in the other using the command

```
[root@bigboy tmp]# tail -f /var/log/maillog
```

The /etc/mail/sendmail.mc File

Most of sendmail's configuration parameters are set in this file with the exception of mailing list and mail relay security features. It is often viewed as an intimidating file with its series of structured "directive" statements that get the job done. Fortunately in most cases you won't have to edit this file very often.

The two most basic steps in configuring a Sendmail server are to modify this file to enable Sendmail to listen on the NIC interface and to make Sendmail to accept mail from valid web domains.

Why Sendmail Only Listens On The Loopback Interface By Default

All Linux systems have a virtual loopback interface that only lives in memory with an IP address 127.0.0.1. As mail must be sent to a target IP address even when there is no NIC in the box, Sendmail therefore uses the loopback address to send mail to users on the local box. To become a server, and not a client, Sendmail needs to be also configured to listen for messages on the NIC interface.

We can verify that sendmail is running by first using the pgrep command which will return the sendmail process ID number once sendmail is running. If it isn't running, then the return value will be blank.

```
[root@bigboy tmp]# pgrep sendmail
22131
[root@bigboy tmp]#

We can also see the interfaces on which Sendmail is listening
with the "netstat" command.  Sendmail listens on TCP port 25, so
we use "netstat" and "grep" for "25" to see a default
configuration listening only on IP address 127.0.0.1 (loopback).

[root@bigboy tmp]# netstat -an | grep :25 | grep tcp
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN
[root@bigboy tmp]#
```

Edit /etc/mail/sendmail.mc To Make Sendmail Listen On NICs Too

To correct this you'll have to comment out the daemon_options line in the **/etc/mail/sendmail.mc** file with "dnl" statements. It is also good practice to take precautions against SPAM by not accepting mail from domains that don't exist by

commenting out the "**accept_unresolvable_domains**" feature too. See the ***italicized*** lines in the example below.

```
dnl This changes sendmail to only listen on the loopback device
127.0.0.1
dnl and not on any other network devices. Comment this out if
you want
dnl to accept email over the network.
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
dnl NOTE: binding both IPv4 and IPv6 daemon to the same port
requires
dnl a kernel patch
dnl DAEMON_OPTIONS(`port=smtp,Addr=::1, Name=MTA-v6,
Family=inet6')
dnl We strongly recommend to comment this one out if you want to
protect
dnl yourself from spam. However, the laptop and users on
computers that do
dnl not have 24x7 DNS do need this.
dnl FEATURE(`accept_unresolvable_domains')dnl
dnl FEATURE(`relay_based_on_MX')dnl
```

You need to be careful with the **accept_unresolvable_names** feature. In our sample network, bigboy the mail server will not accept email relayed from any of the other PCs on your network if they are not in DNS. The chapter on DNS shows how to create your own internal domain just for this purpose.

Regenerate The sendmail.cf File

Once finished editing the file, we have to regenerate a new sendmail.cf file and restart sendmail.

Note: When sendmail starts, it reads the file sendmail.cf for its configuration. **sendmail.mc** is a more user friendly configuration file and really is much easier to fool around with without getting burned. The **sendmail.cf** file is located in different directories dependent on the version of RedHat you use. **/etc/sendmail.cf** for versions up to 7.3, and **/etc/mail/sendmail.cf** for versions 8.0 and higher.

Redhat versions up to 7.3

```
[root@bigboy tmp]# m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
```

Redhat versions 8.0+

```
[root@bigboy tmp]# m4 /etc/mail/sendmail.mc >
/etc/mail/sendmail.cf
```

Restart sendmail to load the new configuration

```
[root@bigboy tmp]# /etc/init.d/sendmail restart
Shutting down sendmail: [ OK ]
Starting sendmail: [ OK ]
[root@bigboy tmp]#
```

Now Make Sure Sendmail Is Listening On All Interfaces

Sendmail should start listening on all interfaces (0.0.0.0)

```
[root@bigboy tmp]# netstat -an | grep :25 | grep tcp
tcp 0 0 0.0.0.0:25 0.0.0.0:* LISTEN
[root@bigboy tmp]#
```

A General Guide To Using The sendmail.mc File

The sendmail.mc file can seem jumbled. To make it less cluttered I usually create two easily identifiable sections in it with all the custom commands I've ever added.

The first section is near the top where the FEATURE statements usually are, and the second section is at the very bottom.

Sometimes sendmail will archive this file when you do a version upgrade. Having easily identifiable modifications in this file will make post upgrade reconfiguration much easier. Here is a sample:

```
dnl ***** Customised section 1 start *****
dnl
dnl
FEATURE(delay_checks)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_entire_domain)dnl
dnl
dnl
dnl ***** Customised section 1 end *****
```

The /etc/hosts File

It is very important to have a correctly configured /etc/hosts file. Here is a brief example:

```
127.0.0.1       localhost.localdomain localhost
192.168.1.100   bigboy.my-site.com    bigboy    mail    www
```

Here the IP address is followed by the hostname.domain (bigboy.my-site.com) followed by the hostname and all the DNS CNAMEs assigned to the server's IP address.

Sendmail uses this file to determine:

The system name

The domains it is responsible for relaying

Sendmail looks for the IP address of your NIC in **/etc/hosts** and then assumes the first name after it is the fully qualified domain name of the server such as **bigboy.my-site.com**. If bigboy had an entry like this:

```
192.168.1.100     my-site.com        (Wrong!!!)
```

Sendmail would assume the server's name was my-site and that the domain was all of ".com". The server would therefore be open to relay all mail from any ".com" domain and would ignore the security features of the access and relay-domains files we'll describe below.

If you fail to put the IP address of your NIC in the **/etc/hosts** file altogether, then you run the risk of having all your mail appear to come from localhost.localdomain and not bigboy.my-site.com.

Symptoms Of A Bad /etc/hosts File

As discussed above, a poorly configured **/etc/hosts** file can make mail sent from your server to the outside world appear as if it came from users at localhost.localdomain and not bigboy.my-site.com.

Use the sendmail program to send a sample email to someone in verbose mode. Enter some text after issuing the command and end your message with a single "." all by itself on the last line.

```
[root@bigboy tmp]# sendmail -v example@another-site.com
test text
test text
.
example@another-site.com... Connecting to mail.another-site.com.
via esmtp...
220 ltmail.another-site.com LiteMail v3.02(BFLITEMAIL4A); Sat,
05 Oct 2002 06:48:44 -0400
>>> EHLO localhost.localdomain
250-mx.another-site.com Hello [67.120.221.106], pleased to meet
you
250 HELP
>>> MAIL From:<root@localhost.localdomain>
250 <root@localhost.localdomain>... Sender Ok
>>> RCPT To:<example@another-site.com>
250 <example@another-site.com>... Recipient Ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 Message accepted for delivery
example@another-site.com... Sent (Message accepted for delivery)
Closing connection to mail.another-site.com.
>>> QUIT
[root@bigboy tmp]#
```

Localhost.localdomain is the domain that all computers use to refer to themselves, it is therefore an illegal internet domain. If mail sent from computer PC1 to PC2 appears to come from a user at localhost.localdomain on PC1 and is rejected, the rejected email will be returned to localhost.localdomain. PC2 will see that the mail originated from localhost.localdomain and will think that the rejected email should be sent to a user on PC2 that may not exist. You will probably get an error like this in **/var/log/maillog** if this happens:

```
Oct 16 10:20:04 bigboy sendmail[2500]: g9GHK3iQ002500:
SYSERR(root): savemail: cannot save rejected email anywhere
```

```
Oct 16 10:20:04 bigboy sendmail[2500]: g9GHK3iQ002500:
Losing ./qfg9GHK3iQ002500: savemail panic
```

Note: You may also get this error if you are using a SPAM prevention program, for example a script based on the PERL module Mail::Audit. An error in the script could cause this type of message too.

Another set of tell tale errors caused by the same problem can be generated when trying to send mail to a user , in this example "root", or creating a new alias database file. (The newalias command will be explained later):

```
[root@bigboy tmp]# sendmail -v  root
WARNING: local host name (bigboy) is not qualified; fix $j in
config file
[root@bigboy tmp]# newaliases
WARNING: local host name (bigboy) is not qualified; fix $j in
config file
[root@bigboy tmp]#
```

With the accompanying error in **/var/log/maillog** log file that looks like this:

```
Oct 16 10:23:58 bigboy sendmail[2582]: My unqualified host name
(bigboy) unknown; sleeping for retry
```

The /etc/mail/relay-domains File

The **/etc/mail/relay-domains** file is used to determine domains from which it will relay mail. The contents of the relay-domains file should be limited to those domains that can be trusted not to originate spam. By default, this file does not exist in a standard RedHat install. In this case, all mail sent from **my-super-duper-site.com** and not destined for this mail server will be forwarded.

```
my-super-duper-site.com
```

One disadvantage of this file is that it can only control mail based on the source domain which can be spoofed by SPAM email servers. The **/etc/mail/access** file has more capabilities, such as restricting relaying by IP address or network range and is more commonly used. If you delete **/etc/mail/relay-domains**, then relay access is fully determined by the **/etc/mail/access** file.

Sendmail has to be restarted after editing this file for the changes to take effect.

The /etc/mail/access File

You can make sure that only trusted PCs on your network have the ability to relay mail via your mail server by using the **/etc/mail/access** file. That is to say, the mail server will only relay mail for those PCs on your network that have their email clients configured to use the mail server as their "outgoing SMTP mail server". (In Outlook Express you set this using: Tools Menu -> Accounts -> Properties -> Servers)

If you don't take the precaution of using this feature, you may find your server being used to relay mail for SPAM email sites. Configuring the **/etc/mail/access** file will not stop SPAM coming to you, only spam flowing through you.

The **/etc/mail/access** file has two columns. The first lists IP addresses and domains from which the mail is coming or going. The second lists the type of action to be taken when mail from these sources / destinations is received. Keywords include RELAY, REJECT, OK (not ACCEPT) and DISCARD. There is no third column to state whether the IP address or domain is the source or destination of the mail, Sendmail assumes it could be either and tries to match both. Sendmail will REJECT all other attempted relayed mail that doesn't match any of the entries in the **/etc/mail/access** file. Despite this, my experience has been that control on a per email address basis is much more intuitive via the **/etc/mail/virtusertable** file.

In the sample file below, we allow relaying for only the server itself (127.0.0.1, localhost), two client PCs on your home 192.168.1.X network, everyone on your 192.168.2.X network and everyone passing email through the mail server from servers belonging to my-site.com. Remember that a server will only be considered a part of my-site.com if its IP address can be found in a DNS reverse zone file:

```
localhost.localdomain           RELAY
localhost                       RELAY
127.0.0.1                       RELAY
192.168.1.16                    RELAY
192.168.1.17                    RELAY
192.168.2                       RELAY
my-site.com                     RELAY
```

You'll then have to convert this text file into a Sendmail readable database file named **/etc/mail/access.db**. Here are the commands to do that:

```
[root@bigboy tmp]# cd /etc/mail
[root@bigboy mail]# make
```

Remember that the relay security features of this file may not work if you don't have a correctly configured /etc/hosts file.

The /etc/mail/local-host-names File

When sendmail receives mail, it needs a way of determining whether it is responsible for the mail it receives. It uses the **/etc/mail/local-host-names** file to do this. This file has a list of hostnames and domains for which sendmail will accept responsibility. For example, if this mail server was to accept mail for the domains my-site.com and my-other-site.com and the host server.my-site.com then the file would look like this:

```
my-site.com
my-other-site.com
```

In this case, remember to modify the MX record of the "my-other-site.com" DNS zonefile point to my-site.com. Here is an example (Remember each "." is important):

```
my-other-site.com. MX 10 mail.my-site.com. ; Primary Mail
Exchanger for my-other-site.com
```

Which User Should Really Receive The Mail?

Sendmail uses two different methods to determine who the ultimate mail recipient will be. It checks these methods in this order:

The /etc/mail/virtusertable file

This file has two columns.

The first lists the destination to which the original sender intended to send the mail.

The second column lists the single true destination.

The true destination in the eyes of the mail server could be a local Linux user, a mailing list entry in the **/etc/aliases** file or the email address of someone on some other mail server to which the mail should be automatically forwarded.

The /etc/aliases file

This file has two columns too. It could be viewed as a mailing list file. The first column has the mailing list name (sometimes called a virtual mailbox) and the second column has the members of the mailing list separated by commas.

If the mailing list member doesn't have an "@" in the name, then sendmail assumes the recipient is on the local box.

It will then search the first column of the aliases file to see if the recipient isn't on yet another mailing list.

If it doesn't find a duplicate, it assumes the recipient is a local user.

If the recipient is a mailing list, then it goes through the process all over again to determine each individual in the mailing list and when it is all finished, they will all get a copy of the email message.

The /etc/mail/virtusertable file

This file contains a set of simple instructions on what to do with received mail. The first column lists the target email address and the second column lists the local user's mail box or remote email address to which the email should be forwarded.

In the example below; mail sent to:

webmaster@my-other-site.com will go to local user (or mailing list) "webmasters", all other mail to my-other-site.com will go to local user "marc".

"sales" at my-site.com will go to the sales department at my-othersite.com.

"paul" and "finance" at my-site.com goes to local user (or mailing list) "paul"

all other users at my-site.com receive a "bounce back" message stating "User unknown"

```
webmaster@my-other-site.com   webmasters
@my-other-site.com            marc
sales@my-site.com             sales@my-other-site.com
paul@my-site.com              paul
finance@my-site.com           paul
@my-site.com                  error:nouser User unknown
```

After editing this file you'll have to convert it into a sendmail readable database file named **/etc/mail/virtusertable.db**. Here are the commands to do that:

```
[root@bigboy tmp]# cd /etc/mail
[root@bigboy mail]# make
```

The /etc/aliases File

This file is really a list of email aliases for local users. It contains a list of virtual mail boxes (or mailing lists) in the first column, and members of the mailing lists in the second column. In the example below, you can see that mail sent to users "bin", "daemon", "lp", "shutdown", "apache", "named"... etc by system processes will all be sent to user (or mailing list) "root". In this case "root" is actually an alias for a mailing list consisting of user "marc" and webmaster@my-site.com.

Note: The default **/etc/aliases** file installed with RedHat has the last line of this sample commented out with a "#", you may want to delete the comment and change user "marc" to another user.

```
# Basic system aliases -- these MUST be present.
mailer-daemon:        postmaster
postmaster:           root

# General redirections for pseudo accounts.
bin:                  root
daemon:               root
lp:                   root
shutdown:             root
mail:                 root
apache:               root
named:                root
system:               root
manager:              root
abuse:                root

# trap decode to catch security attacks
decode:               root

# Person who should get root's mail
root:                 marc,webmaster@my-site.com
```

Notice that there are no spaces between the mailing list entries for “**root**”. This is important as you will get errors if you add spaces.

After editing this file you'll have to convert it into a sendmail readable database file named **/etc/aliases.db**. Here is the command to do that:

```
[root@bigboy tmp]# newaliases
```

Simple Mailing Lists Using Aliases

In the simple mailing list example above, mail sent to "root" actually goes to user account "marc" and webmaster@my-site.com. Here are a few more list examples for your /etc/aliases file.

Mail to "directors@my-site.com" goes to users "peter", "paul" and "mary".

```
# Directors of my SOHO company
directors:      peter,paul,mary
```

Mail sent to "family@my-site.com" goes to users "grandma", "brother" and "sister"

```
# My family
family:         grandma,brother,sister
```

Mail sent to admin-list gets sent to all the users listed in the file **/home/mailings/admin-list**. The advantage of using mailing list files is that the admin-list file can be a file that trusted users can edit, user “**root**” is only needed update the aliases file. Despite this, there are some problems with mail reflectors. One is that bounce messages from failed attempts to broadcast goes to all users. Another is that all subscriptions and unsubscriptions have to be done manually by the mailing list administrator. If either of these are a problem for you, then consider using a mailing list manager like majordomo.

```
# My mailing list file
admin-list:     ":include:/home/mailings/admin-list"
```

After editing this file, you'll have to convert it into a sendmail readable database file named **/etc/aliases.db**. Here is the command to do that:

```
[root@bigboy tmp]# newaliases
```

An Important Note About The /etc/aliases File

By default your system uses sendmail to mail system messages to local user "root". When sendmail sends email to a local user, it will have no "to:" in the email header. If you then use a mail client like Outlook Express with a spam mail filtering rule to reject mail with no to: in the header, you may find yourself dumping legitimate mail.

To get around this, try making root have an alias for a user with a fully qualified domain name, this will force sendmail to insert the correct fields in the header. Here is an example:

```
# Person who should get root's mail
root:                   webmaster@my-site.com
```

Sendmail Masquerading Explained

If you want your mail to appear to come from user@mysite.com and not user@bigboy.mysite.com then you have two choices:

Configure your email client, such as Outlook Express, to set your email address to user@mysite.com. This explained later in this chapter in the POP Mail section.

Set up masquerading to modify the domain name of all traffic originating from and passing trough your mail server.

Configuring masquerading

In the DNS configuration, we made bigboy the mailserver for the domain my-site.com. You now have to tell bigboy in the sendmail configuration file sendmail.mc that all outgoing mail originating on bigboy should appear to be coming from my-site.com, if not, based on our settings in the **/etc/hosts file**, it will appear to come from mail.my-site.com. This isn't terrible, but you may not want your website site to be remembered with the word "mail" in front of it. In other words you may want your mail server to handle all email by assigning a consistent return address to all outgoing mail, no matter which server originated the email.

This can be solved by editing your **sendmail.mc** configuration file and adding some masquerading commands and directives. These are explained below:

```
FEATURE(always_add_domain)dnl
FEATURE(`masquerade_entire_domain')dnl
FEATURE(`masquerade_envelope')dnl
FEATURE(`allmasquerade')dnl
MASQUERADE_AS(`my-site.com')dnl
MASQUERADE_DOMAIN(`my-site.com.')dnl
MASQUERADE_DOMAIN(localhost)dnl
MASQUERADE_DOMAIN(localhost.localdomain)dnl
```

- The **MASQUERADE_AS** directive will make all mail originating on bigboy appear to come from a server within the domain my-site.com by rewriting the email header.
- The **MASQUERADE_DOMAIN** directive will make mail relayed via bigboy from all machines in the my-other-site.com and localdomain domains appear to come from the **MASQUERADE_AS** domain of my-site.com. Sendmail uses DNS to check the domain name associated with the IP address of the mail relay client sending the mail to help it determine whether it should do masquerading or not.
- Feature "**masquerade_entire_domain**" makes sendmail masquerade servers named *my-site.com, and *my-other-site.com as my-site.com. In other words, mail from sales.my-site.com would be masqueraded as my-site.com. If this wasn't selected, then only servers named my-site.com and my-othersite.com would be masqueraded. Use this with caution when you are sure you have the authority to do this.

- Feature "**allmasquerade**" will make sendmail rewrite both recipient addresses and sender addresses relative to the local machine. If you cc: yourself on an outgoing mail, the other recipient will see a cc: to an address he knows instead of one on localhost.localdomain.
- Feature "**always_add_domain**" will always masquerade email addresses, even if the mail is sent from a user on the mail server to another user on the same mail server.
- Feature "**masquerade_envelope**" will rewrite the email envelope just as "**MASQUERADE_AS**" rewrote the header.

The email header is what email clients, such as Outlook Express, say the "to:" and "from:" should be. The "to:" and "from:" in the header is what is used when you use Outlook Express to do a "reply" or "reply all". It is easy to fake the header, as Spammers often do, it is detrimental to email delivery to fake the envelope.

The email envelope contains the "to:" and "from:" used by mailservers for protocol negotiation. It is the envelope's "from:" which is used when email rejection messages are sent between mail servers.

Testing Masquerading

The best way of testing masquerading from the Linux command line is to use the "**mail** *-v username*" command. I have noticed that "**sendmail** *-v username*" ignores masquerading altogether. You should also tail the **/var/log/maillog** file to verify that the masquerading is operating correctly and check the envelope and header of test email received by test email accounts.

Other Masquerading Notes

By default, user "**root**" will not be masqueraded. This is achieved with the:

```
EXPOSED_USER(`root')dnl
```

command in **/etc/mail/sendmail.mc**. You can comment this out if you like with a "dnl" at the beginning of the line and recompiling / restarting sendmail

Using Sendmail to Change the Sender's Email Address

Sometimes masquerading isn't enough. There are times when you may need to change not only the domain of the sender but also the username portion of the sender's email address.

For example, the need to do this could occur after buying a program for your SOHO office that sends out notifications to your staff, but the sender's address inserted by the program is its own, and not that of the IT person.

Sendmail allows you to change both the domain and username on a case by case basis using the genericstable feature.

To do this you need to:

Add these statements to your **/etc/mail/sendmail.mc** file to activate the feature:

```
FEATURE(`genericstable',`hash -o
/etc/mail/genericstable.db')dnl
GENERICS_DOMAIN_FILE(`/etc/mail/generics-domains')dnl
```

Create a **/etc/mail/generics-domains** file that is just a list of all the domains that should be inspected. This should include your server's canonical domain name which can be obtained by using the command:

```
sendmail -bt -d0.1 </dev/null
```

Here is a sample **/etc/mail/generics-domains** file:

```
my-site.com
my-other-site.com
bigboy.my-site.com
```

Create your **/etc/mail/genericstable** file. First sendmail searches the **/etc/mail/generics-domains** file mentioned in the previous step for a list of domains to reverse map. It then looks at the **/etc/mail/genericstable** file for an individual email address from a matching domain. The format of the file is

```
linux-username        username@new-domain.com
```

Here is an example:

```
alert           security-alert@my-site.com
peter           urgent-message@my-site.com
```

Run the sendmail restart script I mentioned at the beginning of the chapter and then test.

Using Public SPAM Blacklists With Sendmail

There are many publicly available lists of known open mail relay servers and spam generating mail servers on the Internet. Some are maintained by volunteers, others are managed by public companies, but in all cases they rely heavily on complaints from spam victims. Some spam blacklists simply try to determine whether the email is coming from a legitimate IP address.

The IP addresses of offenders usually remain on the list for six months to two years. In some cases, to provide additional pressure on the spammers, the blacklists will include not only the offending IP address but also the entire subnet or network block to which it belongs. This prevents the spammers from easily switching their server's IP address to the next available one on their network. Also, if the spammer uses a public datacenter, it is possible that their activities would also cause the IP addresses of legitimate emailers to be black listed too. It is hoped that these legitimate users will pressure the datacenter's management to evict the spamming customer.

You can configure sendmail to use its `**dnsbl**' feature to both query these lists and reject the mail if a match is found. Here are some sample entries you can add to your **/etc/sendmail.mc** file, they should all be on one line. You can visit the URLs listed to learn more about the individual services.

RFC-Ignorant

Valid IP address checker.

```
FEATURE(`dnsbl', `ipwhois.rfc-ignorant.org',`"550 Mail from "
$&{client_addr} " refused. Rejected for bad WHOIS info on IP of
your SMTP server - see http://www.rfc-ignorant.org/"')
```

Easynet

Open proxy list.

```
FEATURE(`dnsbl', `proxies.blackholes.easynet.nl', `"550 5.7.1
ACCESS DENIED to OPEN PROXY SERVER "$&{client_name}" by
easynet.nl
DNSBL  (http://proxies.blackholes.easynet.nl/errors.html)"',
`')dnl
```

The Open Relay Database

Open mail relay list.

```
FEATURE(`dnsbl', `relays.ordb.org', `"550 Email rejected due to
sending server misconfiguration - see
http://www.ordb.org/faq/\#why_rejected"')dnl
```

Spamcop

Spammer blacklist.

```
FEATURE(`dnsbl', `bl.spamcop.net', `"450 Mail from "
$`'&{client_addr} " refused - see http://spamcop.net/bl.shtml"')
```

Spamhaus

Spammer blacklist.

```
FEATURE(`dnsbl',`sbl.spamhaus.org',`Rejected - see
http://spamhaus.org/')dnl
```

A Simple PERL Script To Help Stop SPAM

Blacklists won't stop everything.

It is possible to limit the amount of unsolicited commercial email (UCE or spam) you receive by writing a small script to intercept your mail before it is written to your mailbox.

This is fairly simple to do as sendmail always checks the "**.forward**" file in your home directory for the name of this script. Sendmail then looks for the filename in the directory **/etc/smrsh** and executes it.

By default, PERL doesn't come with modules that are able to check email headers and envelopes so you will have to download them from CPAN (www.cpan.org). The most important modules are:

MailTools

IO-Stringy

MIME-tools

Mail-Audit

I have written a script called mail-filter.pl that effectively filters out SPAM email for my home system. There are a few steps required to make the script work:

Install PERL and the PERL modules listed above.

Place an executable version of the script in your home directory and modify the script's $FILEPATH variable point to your home directory

Update the two configuration files:

mail-filter.accept, which specifies the subjects and email addresses to accept,

mail-filter.reject that specifies those that you should reject.

Update your "**.forward**" file and place an entry in **/etc/smrsh**

Mail-filter will first reject all email based on the "reject" file and will then accept all mail found in the "accept" file. It will then deny everything else.

I have included a simple script with instructions on how to install the PERL modules in the Appendix.

Configuring Your POP Mail Server

Sendmail will just handle mail sent to your "my-site.com" domain. Each user on your Linux box will get mail sent to their account's mail folder. If you want to retrieve this mail from your Linux box's user account, using a mail client such as Microsoft Outlook or Outlook Express, then you have a few more steps. You'll also have to make your Linux box a POP mail server.

Installing Your POP Mail Server

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail.

The IMAP/POP mail suite comes standard with the RedHat installation CDs. You can install the RPM with this command:

```
[root@bigboy tmp]# rpm -Uvh imap-2001a-15.i386.rpm
```

POP mail is started by xinetd. Therefore to get POP mail configured to start at boot you have to use the chkconfig command to make sure xinetd starts up on booting.

```
[root@bigboy tmp]# chkconfig --level 35 xinetd on
```

To start/stop/restart POP mail after booting you can use the xinetd init script located in the directory **/etc/init.d** like this:

```
[root@bigboy tmp]# /etc/init.d/xinetd start
[root@bigboy tmp]# /etc/init.d/xinetd stop
[root@bigboy tmp]# /etc/init.d/xinetd restart
```

Remember to restart the POP mail process every time you make a change to the configuration files for the changes to take effect on the running process

Configuring Your POP Mail Server

The starting and stopping of POP Mail is controlled by xinetd via the **/etc/xinetd.d/ipop3** file. POP Mail is deactivated by default, so you'll have to edit this file to start the program. Make sure the contents look like this. The disable feature must be set to "no" to accept connections.

Follow the steps below and set the "**disable**" parameter to "no".

```
[root@bigboy tmp]# cd /etc/xinetd.d
[root@bigboy xinetd.d]# vi ipop3

# default: off
# description: The POP3 service allows remote users
# to access their mail \
# using an POP3 client such as Netscape Communicator, mutt, \
# or fetchmail.
service pop3
{
   socket_type = stream
   wait = no
   user = root
   server = /usr/sbin/ipop3d
   log_on_success += HOST DURATION
   log_on_failure += HOST
   disable = no
}
```

You will then have to restart xinetd for these changes to take effect using the startup script in the **/etc/init.d** directory.

Naturally, to disable POP Mail once again, you'll have to edit the /etc/xinetd.d/ipop3 file, set "disable" to "yes" and restart xinetd.

How To Configure Your Windows Mail Programs

All your POP email accounts are really only regular Linux user accounts in which Sendmail has deposited mail. You can now configure your email client such as Outlook Express to use your use your new POP / SMTP Mail Server quite easily. Here's how:

POP Mail

Set your POP mail server to be the IP address of your Linux mail server. Use your Linux user username and password when prompted.

SMTP

Set your SMTP mail server to be the IP address / domain name of your Linux mail server.

How to handle overlapping email addresses.

If you have a user overlap, eg. John Smith (john@my-site.com) and John Brown (john@my-other-site.com), by default, both users will get sent to the Linux user account "john". You have two choices:

Make the user part of the email address is different. For example: john1@my-site.com and john2@my-other-site.com. Create Linux accounts "john1" and "john2". If the users insist on overlapping names then you may need to modify your virtusertable file.

Create the user accounts "john1" and "john2". Have virtusertable entries for john@my-site.com pointing to account "john1" and john@my-other-site.com pointing to account "john2". The POP configuration in Outlook Express for each user should POP using "john1" and "john2" respectively.

Chapter 10

Monitoring Server Performance

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

Monitoring your system's web performance can be done quite easily with a number of graphical tools available for Linux. These include MRTG for raw network traffic which is based on SNMP and Webalizer that monitors web site hits.

SNMP

What is SNMP?

Most routers and firewalls keep their operational statistics in Management Information Blocks (MIBs). Each statistic has an Object Identifier (OID) and can be remotely retrieved from the MIB via the Simple Network Management Protocol (SNMP). However, as a security measure, you need to know the SNMP password or "community string" to do so. There are a number of types of community strings, the most commonly used ones are the "Read Only" community string that only provides access for viewing statistics and system parameters. In many cases the "Read Only" community string or password is set to "public". There is also a "Read Write" community string for not only viewing statistics and system parameters but also for updating the parameters too.

Doing SNMP Queries

If you intend to use your Linux box to query your network devices, other servers or even itself using MRTG or any other tool, you will need to have the SNMP tools package **net-snmp-utils** installed.

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail.

- For example, the RedHat 9.0 RPM as of this writing was:

```
net-snmp-utils-5.0.6-17.i386.rpm
```

- Install the package using the **rpm** command

```
[root@bigboy tmp]# rpm -Uvh net-snmp-utils-5.0.6-17.i386.rpm
```

SNMP on a Linux Server

By default, RedHat Linux has the NetSNMP server package installed to provide SNMP services. NetSNMP uses a configuration file **/etc/snmp/snmpd.conf** in which the community strings may be set. The version of the configuration file that comes with Net-SNMP is quite complicated. I suggest archiving it and using a much simpler version with only a single line containing the keyword "**rocommunity**" followed by the community string. Here is an example of how to do that.

- Save the old configuration file

```
[root@bigboy snmp]# cd /etc/snmp/
[root@bigboy snmp]# mv snmpd.conf snmpd.conf.old
[root@bigboy snmp]# vi snmpd.conf
```

- Enter the following line in the new configuration file to set the Read Only community string to "craz33guy"

```
rocommunity craz33guy
```

- Configure Linux to start SNMP services on each reboot with the chkconfig command:

```
[root@bigboy root]# chkconfig --level 345 snmpd on
[root@bigboy root]#
```

- You can then start SNMP to load the current configuration file.

```
[root@bigboy root]# /etc/init.d/snmpd start
Starting snmpd: [ OK ]
[root@bigboy root]#
```

- Test whether SNMP can read the "system" and "interface" information MIB

```
[root@bigboy snmp]# snmpwalk -v 1 -c craz33guy localhost
system
SNMPv2-MIB::sysDescr.0 = STRING: Linux bigboy 2.4.18-14 #1
Wed Sep 4 11:57:57 EDT 2002 i586
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-
MIB::netSnmpAgentOIDs.10
SNMPv2-MIB::sysUpTime.0 = Timeticks: (425) 0:00:04.25
SNMPv2-MIB::sysContact.0 = STRING: root@localhost
SNMPv2-MIB::sysName.0 = STRING: bigboy
...
...
...
[root@bigboy snmp]# snmpwalk -v 1 -c craz33guy localhost
interface
IF-MIB::ifNumber.0 = INTEGER: 3
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
```

```
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: wlan0
IF-MIB::ifDescr.3 = STRING: eth0
...
...
...
[root@bigboy snmp]#
```

Now that we know SNMP is working correctly on your Linux server, we can configure a SNMP statistics gathering software package such as MRTG to create online graphs of your traffic flows.

SNMP On Other Devices

In case above we were polling localhost. You can poll any SNMP aware network device that has SNMP enabled. All you need is the IP address and SNMP read only string and you'll be able to get similar results.

There are currently three versions of SNMP; versions 1, 2 and 3. The Linux **snmpwalk** and **snmpget** commands have "**–v 1**", "**–v 2c**" and "**–v 3**" switches for specifying the SNMP version to be used for queries. Always make sure you are using the correct one.

MRTG

What is MRTG?

MRTG (Multi Router Traffic Grapher) is a public domain package for producing graphs of various types of router statistics via a web page. You can easily create graphs of traffic flow statistics through your home network's firewall / router or even your Linux box's NIC cards using MRTG. The product is available from the MRTG website and also on your distribution CDs.

Download and Install The MRTG Packages

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail. The latest version of the RPM for RedHat 9.0 is:

```
mrtg-2.9.17-13.i386.rpm
```

You can install the package like this:

```
[root@bigboy tmp]# rpm -Uvh mrtg-2.9.17-13.i386.rpm
```

You will also need to have a webserver package installed for MRTG to work. RedHat Linux usually comes with the Apache webserver software preinstalled. The easiest way to tell if Apache is installed is to see if the file /etc/init.d/httpd exists on your server. If not you can refer to the Apache chapter on how to install it. By default Apache expects the

HTML files for your website to be located in **/var/www/html**. MRTG will place its HTML files in **/var/www/html/mrtg**.

Note: Make sure you have the SNMP utilities installed before proceeding. They will help a lot in testing to see whether MRTG is installed correctly.

Configuring MRTG

By default, MRTG will map the inbound and outbound data throughput rates on the device it is polling. There are ways to specify other OIDs such as CPU and memory usage, but this is beyond the scope of this book. We'll be discussing the default configuration.

When the MRTG RPM is installed it creates a directory called **/etc/mrtg** in which all future configuration files are stored. Here are the steps you need to go through to create new configuration files.

First of all, take a look at the file **/etc/sysconfig/i18n**. This governs the default character set used by Linux. MRTG doesn't seem to like the default "en_US.UTF-8" language setting, and prefers plain "en_US". Just replace this line:

```
LANG="en_US.UTF-8"
```

with

```
LANG="en_US"
```

In this example we'll use MRTG's **cfgmaker** command to create a configuration file named **localhost.cfg** for the server "bigboy" using a read only community string of **craz33guy**. All data files will be placed in the directory **/var/www/html/mrtg/stats**.

```
[root@bigboy tmp]# cfgmaker --output=/etc/mrtg/localhost.cfg \
-ifref=ip --global "workdir: /var/www/html/mrtg/stats" \
craz33guy@localhost


--base: Get Device Info on craz33guy@localhost:
--base: Vendor Id:
--base: Populating confcache
--snpo: confcache craz33guy@localhost: Descr lo --> 1
--snpo: confcache craz33guy@localhost: Descr wlan0 --> 2
--snpo: confcache craz33guy@localhost: Descr eth0 --> 3
--snpo: confcache craz33guy@localhost: Ip 0.0.0.0 --> 3
--snpo: confcache craz33guy@localhost: Ip 127.0.0.1 --> 1
--snpo: confcache craz33guy@localhost: Ip 192.168.1.100 --> 2
--snpo: confcache craz33guy@localhost: Type 24 --> 1
--snpo: confcache craz33guy@localhost: Type 6 --> 2
--snpo: confcache craz33guy@localhost: Type 6 --> 3 (duplicate)
--snpo: confcache craz33guy@localhost: Eth --> 1
--snpo: confcache craz33guy@localhost: Eth 00-06-25-09-6a-b5 --
> 2
--snpo: confcache craz33guy@localhost: Eth 00-08-c7-10-74-a8 --
> 3
--base: Get Interface Info
--base: Walking ifIndex
```

```
--base: Walking ifType
--base: Walking ifSpeed
--base: Walking ifAdminStatus
--base: Walking ifOperStatus
--base: Writing /etc/mrtg/localhost.cfg
[root@bigboy tmp]#
```

Troubleshooting Tip: As explained in the SNMP section, there are different versions of SNMP. If your query doesn't work, check to make sure you are using the required version and then other SNMP configuration parameters on the target device. You can specify MRTG's SNMP query version with the **--snmp-options cfgmaker** option. Here is an example of **cfgmaker** using an SNMP version 2 query of a router with an IP address of 192.168.1.3. The **--snmp-options** option's five colons before the number "2" are important.

```
[root@bigboy tmp]# cfgmaker --output=/etc/mrtg/192.168.1.3.cfg
\
-ifref=ip --global "workdir: /var/www/html/mrtg/stats" \
--snmp-options=:::::2 craz33guy@192.168.1.3
```

Next create the **/var/www/html/mrtg/stats** directory and copy all of MRTG's standard "**.png**" image files into it.

```
[root@bigboy mrtg]# mkdir /var/www/html/mrtg/stats
[root@bigboy mrtg]# cp
/var/www/html/mrtg/*.png  /var/www/html/mrtg/stats
[root@bigboy mrtg]#
```

Edit **/etc/mrtg/localhost.cfg** and remove the sections related to interfaces you don't need to monitor. This would most likely include the loopback interface **L0**: with the IP address of 127.0.0.1

When the MRTG RPM is installed it places an entry in the **/etc/crontab** file to make MRTG run every 5 minutes using the default **/etc/mrtg/mrtg.cfg** configuration file. Add a new line referring to **/etc/mrtg/localhost.cfg** and comment out the one pointing to **mrtg.cfg**.

```
# 0-59/5 * * * * root /usr/bin/mrtg /etc/mrtg/mrtg.cfg
0-59/5 * * * * root /usr/bin/mrtg /etc/mrtg/localhost.cfg
```

Run MRTG using **/etc/mrtg/localhost.cfg** as your argument three times. You'll get an error the two times as MRTG tries to move old data files, and naturally, the first time it is run, MRTG has no data files to move.

```
[root@bigboy mrtg]# mrtg /etc/mrtg/localhost.cfg
Rateup WARNING: /usr/bin/rateup could not read the primary log
file for localhost_192.168.1.100
Rateup WARNING: /usr/bin/rateup The backup log file for
localhost_192.168.1.100 was invalid as well
Rateup WARNING: /usr/bin/rateup Can't remove
localhost_192.168.1.100.old updating log file
Rateup WARNING: /usr/bin/rateup Can't rename
localhost_192.168.1.100.log to localhost_192.168.1.100.old
```

```
updating log file
[root@bigboy mrtg]# mrtg /etc/mrtg/localhost.cfg
Rateup WARNING: /usr/bin/rateup Can't remove
localhost_192.168.1.100.old updating log file
[root@bigboy mrtg]# mrtg /etc/mrtg/localhost.cfg
[root@bigboy mrtg]#
```

You'll then want to use MRTG's **indexmaker** command to create a combined index page to see all the graphs defined in all the various ".cfg" files in your **/etc/mrtg** directory. Once this is done, you can point your browser to **http://ip-address/mrtg/** to get a graphical listing of all the monitored interfaces.

Note: The indexmaker command creates a very generic index page which is very similar to the MRTG home page, don't be fooled, you will find your devices at the very bottom. The format of the command is:

```
indexmaker  --output=filename device1.cfg device2.cfg etc
```

RedHat Version 9.0 / 8.0 and Indexmaker

RedHat versions 8 and 9 give an error like this when running indexmaker.

```
[root@bigboy mrtg]# indexmaker --output=index.html
/etc/mrtg/localhost.cfg
Can't locate package $VERSION for @MRTG_lib::ISA at
/usr/bin/indexmaker line 49
main::BEGIN() called at /usr/bin/../lib/mrtg2/MRTG_lib.pm line
49
eval {...} called at /usr/bin/../lib/mrtg2/MRTG_lib.pm line 49
[root@bigboy mrtg]#
```

This is caused by an incompatibility between MRTG and PERL 5.8 which MRTG uses to generate files. The MRTG site claims this was fixed in version 2.9.22, but this version of MRTG seems to fail under RedHat.

The fix is simple:

- Edit the file /usr/lib/mrtg2/MRTG_lib.pm
- Replace the line:

  ```
  @ISA = qw(Exporter $VERSION);
  ```

  with

  ```
  @ISA = qw(Exporter);
  ```

- Run indexmaker again.

Using MRTG To Monitor Other Subsystems

MRTG will generate HTML pages with daily, weekly, monthly and yearly statistics for your interfaces. By default MRTG provides only network interface statistics. The MRTG website www.mrtg.org has links to other sites that show you how to monitor other sub-systems on a variety of devices and operating systems.

Webalizer

What Is Webalizer?

Webalizer is a web server log file analysis tool that comes installed by default on RedHat Linux. Each night, Webalizer reads your Apache log files and creates a set of web pages that allow you to view websurfer statistics for your site. The information provided includes a list of your web site's most popular pages sorted by "hits" along with traffic graphs showing the times of day when your site is most popular.

How To View Your Webalizer Statistics

By default webalizer places its index page in the directory **/var/www/html/usage**, so if you have a default Apache installation you'll be able to view your data by visiting http://www.my-site.com/usage

The Webalizer Configuration File

Webalizer stores its configuration in the file **/etc/webalizer.conf**. The default settings should be sufficient for your web server, but you may want to adjust the directory in which Webalizer places your graphic statistics. This can be adjusted with the **OutputDir** directive in the file.

Make Webalizer run in Quiet Mode

Webalizer has a tendency to create this message in your logs which according to the Webalizer site's documentation is non-critical

Error: Unable to open DNS cache file /var/lib/webalizer/dns_cache.db

You can make the software run in quite mode by editing the **/etc/cron.daily/00webalizer** script file and adding the **–Q** (Quiet) switch to the webalizer command like this:

```
#! /bin/bash
# update access statistics for the web site

if [ -s /var/log/httpd/access_log ] ; then
   /usr/bin/webalizer -Q
fi
exit 0
```

Once you've done this, Webalizer will function with few annoyances, however be aware that running in quiet mode could hide deeper problems that could occur in future.

TOP

You can monitor the amount of memory and CPU resources your system is using the "top" command. In the case below, the CPU usage is under 1.0% and 14% of memory (57536K) is

free. The amount of free memory may appear low, but in this case, the server doesn't seem to be "swapping" idle processes from memory to the swap disk partition as it isn't being used at all.

```
[root@bigboy tmp]# top

  3:04pm  up 25 days, 23:23,  2 users,  load average: 0.00, 0.02, 0.00
78 processes: 76 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  0.9% user,  0.5% system,  0.0% nice,  0.8% idle
Mem:   384716K av,  327180K used,   57536K free,       0K shrd,  101544K
buff
Swap:  779112K av,       0K used,  779112K free                  130776K
cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
27191 root      15   0  1012 1012   780 R     5.6  0.2   0:00 top
 4545 root      16   0  5892 5888  4956 S     0.9  1.5 169:26 magicdev
    1 root      15   0   476  476   432 S     0.0  0.1   0:05 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:00 keventd
    5 root      15   0     0    0     0 SW    0.0  0.0   0:41 kswapd
    6 root      25   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    8 root      25   0     0    0     0 SW    0.0  0.0   0:00 mdrecoveryd

[root@bigboy tmp]#
```

Excessive swapping can cause your system to slow down dramatically, the simplest ways to avoid this is to add more RAM and / or reduce the number of processes or users that are active on your system. Further discussion of system tuning is beyond the scope of this book.

If your system seems slow but the CPU and memory usage is low, then start looking at networking problems such as poor duplex negotiation, bad cables and network congestion due to excessive traffic.

VMSTAT

You can also determine memory and swap usage with the "vmstat" command which provides a summary of what "top" produces. In the case below, memory is still 14% free and swap isn't being used at all.

```
[root@rahtid named]# vmstat
   procs                  memory    swap          io     system
cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy
 id
 0  0  0      0  57452 101584
130780   0   0     0     4   18     1   3   1   1
[root@rahtid named]#
```

Chapter 11

The NTP Server

= =

In This Chapter

© Peter Harrison, www.linuxhomenetworking.com

= =

You can keep accurate time under Linux by synchronizing your system clock with a Network Time Protocol (NTP) Server. A list of available servers may be found at:

http://www.eecis.udel.edu/~mills/ntp/servers.html

What is NTP?

Network Time Protocol (NTP) is a protocol used to help synchronize your system clock with an accurate time source. There are a number of "Stratum 1" (NTP sites using an atomic clock for timing) and "Stratum 2" (NTP sites with slightly less accurate time sources) sites that allow the general public to synchronize with them. It is good practice to have at least one server on your network be the local time server for all your other devices, this makes the correlation of system events on different systems much easier.

There are a number of freely available NTP client programs for Windows. You can use them to practice with your new NTP server.

Download and Install The NTP Package

Most RedHat Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard. If you need a refresher, the chapter on RPMs covers how to do this in detail.

- The latest version of the RPM for RedHat 9.0 is:

```
ntp-4.1.2-0.rc1.2.i386.rpm
```

- Install the package using the following command:

```
[root@bigboy tmp]# rpm -Uvh ntp-4.1.2-0.rc1.2.i386.rpm
```

The /etc/ntp.conf File

This is the main configuration file for Linux NTP in which you place the IP addresses of the stratum 1 and stratum 2 servers you want to use. Here is a sample of a home configuration using a pair of sample Internet based NTP servers:

- First we specify the servers we're interested in:

```
server  otherntp.server.org     # A stratum 1 server at server.org
server  ntp.research.gov        # A stratum 2 server at
research.gov
```

- Then we restrict the type of access you allow these servers. In this example we're not allowing them to modify or query our Linux NTP server.

```
restrict otherntp.server.org   mask 255.255.255.255 nomodify
notrap noquery
restrict ntp.research.gov      mask 255.255.255.255 nomodify
notrap noquery
```

The **mask** statement 255.255.255.255 is really a subnet mask limiting access to the single IP address of the remote NTP servers.

- Now list the NTP clients on our home network which should be querying our server for the time (notice that the **noquery** has been removed):

```
restrict 192.168.1.0 mask 255.255.255.0 notrust nomodify notrap
```

In this case the **mask** statement has been expanded to include all 255 possible IP addresses on our local network.

- We also want to make sure that localhost (The universal IP address used to refer to a Linux server itself) has full access without any restricting keywords:

```
restrict 127.0.0.1
```

- Last, but most importantly, you need to make sure the default restrict statement is removed. It will override all other restrict statements and you'll find your NTP server will only be communicating properly with itself. If the line is there, comment it out like this:

```
#restrict default ignore
```

- Save the file
- Do the following commands twice for each new server added to **/etc/ntp.conf**

```
[root@bigboy tmp]# ntpdate otherntp.research.gov
24 Mar 18:16:36 ntpdate[10254]: step time server 200.100.20.10
offset -15.266188 sec
[root@bigboy tmp]# ntpdate otherntp.research.gov
24 Mar 18:16:43 ntpdate[10255]: adjust time server 200.100.20.10
offset -0.000267 sec
```

How To Get NTP Started

- To get NTP configured to start at boot:

```
[root@bigboy tmp]# chkconfig --level 35 ntpd on
```

- To start/stop/restart NTP after booting:

```
[root@bigboy tmp]# /etc/init.d/ntpd start
[root@bigboy tmp]# /etc/init.d/ntpd stop
[root@bigboy tmp]# /etc/init.d/ntpd restart
```

Remember to restart the NTP process every time you make a change to the conf file for the changes to take effect on the running process

- You can test whether the NTP process is running with the following command, you should get a response of plain old process ID numbers:

```
[root@bigboy tmp]# pgrep ntpd
```

Determining If NTP Is Synchronized Properly

Use the following command to see the servers with which you are synchronized:

```
[root@bigboy tmp]# ntpq -p
```

Sample output:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
-jj.cs.umb.edu   gandalf.sigmaso  3 u   95 1024  377   31.681  -18.549    1.572
milo.mcs.anl.go  ntp0.mcs.anl.go  2 u  818 1024  125   41.993  -15.264    1.392
-mailer1.psc.edu ntp1.usno.navy.  2 u  972 1024  377   38.206   19.589   28.028
-dr-zaius.cs.wis ben.cs.wisc.edu  2 u  502 1024  357   55.098    3.979    0.333
+taylor.cs.wisc. ben.cs.wisc.edu  2 u  454 1024  347   54.127    3.379    0.047
-ntp0.cis.strath harris.cc.strat  3 u  507 1024  377  115.274   -5.025    1.642
*clock.via.net   .GPS.            1 u  426 1024  377  107.424   -3.018    2.534
ntp1.conectiv.c  0.0.0.0         16 u    - 1024    0    0.000    0.000 4000.00
```

A telltale sign that you haven't got proper synchronization is when all the remote servers have jitters of 4000 with delay and reach values of zero.

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)         10 l    -   64    7    0.000    0.000   0.008
 ntp-cup.externa 0.0.0.0          16 u    -   64    0    0.000    0.000 4000.00
 snvl-smtp1.trim 0.0.0.0          16 u    -   64    0    0.000    0.000 4000.00
 nist1.aol-ca.tr 0.0.0.0          16 u    -   64    0    0.000    0.000 4000.00
```

This could be caused by:

- The **restrict default ignore** statement in the **/etc/ntp.conf** file not being commented out.
- A firewall blocking access to your Stratum 1 and 2 NTP servers

Configuring Cisco Devices To Use An NTP Server

Cisco IOS

Here are the commands you would use to make your router synchronize with NTP servers with IP addresses 192.168.1.100 and 192.168.1.201. An explanation of the commands used follows.

```
ciscorouter> enable
password: *********
ciscorouter# config t
ciscorouter(config)# ntp update-calendar
ciscorouter(config)# ntp server 192.168.1.100
ciscorouter(config)# ntp server 192.168.1.201
ciscorouter(config)# exit
ciscorouter# wr mem
```

- ntp server: Forms a server association with another system.
- ntp update-calendar: Configures the system to update its hardware clock from the software clock at periodic intervals.

CAT OS

Here are the commands you would use to make your router synchronize with NTP servers with IP addresses 192.168.1.100 and 192.168.1.201. An explanation of the commands used follows.

```
ciscoswitch> enable
password: *********
ciscoswitch# set ntp client enable
```

```
ciscoswitch# ntp server 192.168.1.100
ciscoswitch# ntp server 192.168.1.100
ciscoswitch# exit
```

- ntp server: Forms a server association with another system.
- set ntp client enable: Activate the NTP client

Firewalls and NTP

NTP servers communicate with one another using UDP with a destination port of 123. Unlike most UDP protocols, the source port isn't a high port (ie. greater than 1023), but 123 also. You'll have to allow UDP traffic on source/destination port 123 between your server and the Stratum 1/2 server with which you are synchronizing.

A sample Linux iptables firewall script snippet is in the Appendix.