How To Set Up A Debian Linux Firewall

The material on this page was prepared using **Sarge** or **Etch**
configured using our Installation and Packages pages.
If you did not use our pages to set up your system, what you
encounter on your system may be different than what is given here.

This page covers using IPTABLES with the 2.4 Linux kernel.
For the page on using IPCHAINS with the 2.2 Linux kernel click here.

A firewall is a system or router that sits between an external network (i.e. the Internet) and an internal network. This internal network can be a large LAN at a business or your networked home PCs. The firewall in it's simplest form is like a one-way street. It allows people on the internal network to access the external network (the Intenet), but it restricts traffic so that no one can use the external network to access the systems or files on the internal network.

If you connect to the Internet using a modem you typically don't have to be too concerned about firewalling your internal network. That's because each time you connect to the Internet you are assigned a different IP address by your ISP, and your dial-up sessions are limited in length. However, if you have a DSL line or a cable modem, you retain the same IP address for much longer periods of time, perhaps months, because you have an "always on" connection to the Internet. In this case a firewall is advisable.

A firewall has two network connections, one for the external network and one for the internal network. Traffic that is allowed to flow between the two networks is internally "bridged" (using a FORWARD rule) between these two connections. Disallowed traffic is not. This decision-based bridging of traffic between two connections is called "routing" or "IP forwarding". What this means is that any firewall, by its very nature, is a router (but not all routers block traffic, so not all routers are firewalls).

You can buy "cable/DSL routers" (proxy-in-a-box) for under $100 which offer integrated firewalling functionality. However, the firewalling is typically limited in nature and may not be able to be upgraded as new threats are discovered.

So why use a Linux system as a firewall if these $100 boxes are available? One reason is flexibility. You could use your Linux firewall system to simultaneously act as a Web (possibly with a Web cam) and/or e-mail server (more about this can be found on the Internet Servers and Web Cam pages). Plus you can set up very fine controls about who can access what on the Internet. Many organizations only want a few select individuals to have Internet access. In this case you could statically assign them addresses in a range outside of the DHCP scope and only allow that address range out. (If you have your own internal DNS server be sure to allow it outbound UDP port 53 so it can access the root hints servers.) But the main reason for setting up a Linux firewall is because it will help you learn Linux.

The Firewall Script

If you haven't already done so, please read the Proxy/NAT page. Much of the information for proxy servers and firewalls is the same and we won't be repeating it here. As a matter of fact, if you created the proxy server on the previous page, all you have to do is add a few more IPTABLES commands to enhance the firewalling functionality of the system.

All we're going to do is take the proxy server shell script from the Proxy/NAT page and add some more rules to it. Whereas the proxy script only had specific rules related to forwarding, the modified script will have all three types of rules (input, ouput, and forwarding). To set this script up you'll need to:

Section A
- Enter your internal interface designation (`INTIF`)
- Enter your internal **network** address (`INTNET`)
- Enter your internal **interface IP** address (`INTIP`)
- Enter your external interface designation (`EXTIF`)

Section B
- If your external interface uses a **static** IP address
  - Uncomment the `EXTIP` line and enter your static IP address

Section C
- If your external interface uses a **dynamic** IP address
  - Uncomment the `EXTIP` line

Optional
- If you plan to simultaneously use your firewall system as a Web server uncomment the two `OPTIONAL:` lines (**echo**

and **iptables**) in the `INPUT` section.

The comments in the script give a little more information on what values to enter and what lines need to be uncommented for your situation. If you want to have a Web server but don't feel comfortable using your firewall system to act as one, we show you how to set up the firewall to forward traffic to a separate Web server that's behind the firewall in the **DMZ** section below.

```
#!/bin/sh

#  IPTABLES  FIREWALL  script for the Linux 2.4 kernel.
#  This script is a derivitive of the script presented in
#  the IP Masquerade HOWTO page at:
#  www.tldp.org/HOWTO/IP-Masquerade-HOWTO/stronger-firewall-examples.html
#  It was simplified to coincide with the configuration of
#  the sample system presented in the Guides section of
#  www.aboutdebian.com
#
#  This script is presented as an example for testing ONLY
#  and should not be used on a production firewall server.
#
#    PLEASE SET THE USER VARIABLES
#    IN SECTIONS A AND B OR C

echo -e "\n\nSETTING UP IPTABLES FIREWALL..."


# === SECTION A
# -----------   FOR EVERYONE

# SET THE INTERFACE DESIGNATION AND ADDRESS AND NETWORK ADDRESS
# FOR THE NIC CONNECTED TO YOUR _INTERNAL_ NETWORK
#   The default value below is for "eth0".  This value
#   could also be "eth1" if you have TWO NICs in your system.
#   You can use the ifconfig command to list the interfaces
#   on your system.  The internal interface will likely have
#   have an address that is in one of the private IP address
#   ranges.
#       Note that this is an interface DESIGNATION - not
#       the IP address of the interface.

# Enter the designation for the Internal Interface's
INTIF="eth0"

# Enter the NETWORK address the Internal Interface is on
INTNET="192.168.1.0/24"

# Enter the IP address of the Internal Interface
INTIP="192.168.1.1/24"



# SET THE INTERFACE DESIGNATION FOR YOUR "EXTERNAL" (INTERNET) CONNECTION
#   The default value below is "ppp0" which is appropriate
#   for a MODEM connection.
#   If you have two NICs in your system change this value
#   to "eth0" or "eth1" (whichever is opposite of the value
#   set for INTIF above).  This would be the NIC connected
#   to your cable or DSL modem (WITHOUT a cable/DSL router).
#       Note that this is an interface DESIGNATION - not
#       the IP address of the interface.
#   Enter the external interface's designation for the
#   EXTIF variable:

EXTIF="ppp0"


# ! ! ! ! !  Use ONLY Section B  *OR*  Section C depending on
#  ! ! ! !   the type of Internet connection you have.
# ! ! ! ! !  Uncomment ONLY ONE of the EXTIP statements.


# === SECTION B
# -----------   FOR THOSE WITH STATIC PUBLIC IP ADDRESSES

   # SET YOUR EXTERNAL IP ADDRESS
   #   If you specified a NIC (i.e. "eth0" or "eth1" for
   #   the external interface (EXTIF) variable above,
   #   AND if that external NIC is configured with a
   #   static, public IP address (assigned by your ISP),
   #   UNCOMMENT the following EXTIP line and enter the
   #   IP address for the EXTIP variable:

#EXTIP="your.static.IP.address"
```

```
# === SECTION C
# ----------   DIAL-UP MODEM, AND RESIDENTIAL CABLE-MODEM/DSL (Dynamic IP) USERS


# SET YOUR EXTERNAL INTERFACE FOR DYNAMIC IP ADDRESSING
#   If you get your IP address dynamically from SLIP, PPP,
#   BOOTP, or DHCP, UNCOMMENT the command below.
#   (No values have to be entered.)
#         Note that if you are uncommenting these lines then
#         the EXTIP line in Section B must be commented out.

#EXTIP="`/sbin/ifconfig ppp0 | grep 'inet addr' | awk '{print $2}' | sed -e 's/.*://'`"


# --------  No more variable setting beyond this point  --------


echo "Loading required stateful/NAT kernel modules..."

/sbin/depmod -a
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe ip_conntrack_irc
/sbin/modprobe iptable_nat
/sbin/modprobe ip_nat_ftp
/sbin/modprobe ip_nat_irc

echo "    Enabling IP forwarding..."
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

echo "    External interface: $EXTIF"
echo "       External interface IP address is: $EXTIP"
echo "    Loading firewall server rules..."

UNIVERSE="0.0.0.0/0"

# Clear any existing rules and setting default policy to DROP
iptables -P INPUT DROP
iptables -F INPUT
iptables -P OUTPUT DROP
iptables -F OUTPUT
iptables -P FORWARD DROP
iptables -F FORWARD
iptables -F -t nat

# Flush the user chain.. if it exists
if [ "`iptables -L | grep drop-and-log-it`" ]; then
   iptables -F drop-and-log-it
fi

# Delete all User-specified chains
iptables -X

# Reset all IPTABLES counters
iptables -Z

# Creating a DROP chain
iptables -N drop-and-log-it
iptables -A drop-and-log-it -j LOG --log-level info
iptables -A drop-and-log-it -j REJECT

echo -e "     - Loading INPUT rulesets"

######################################################################
# INPUT: Incoming traffic from various interfaces.  All rulesets are
#        already flushed and set to a default policy of DROP.
#

# loopback interfaces are valid.
iptables -A INPUT -i lo -s $UNIVERSE -d $UNIVERSE -j ACCEPT

# local interface, local machines, going anywhere is valid
iptables -A INPUT -i $INTIF -s $INTNET -d $UNIVERSE -j ACCEPT

# remote interface, claiming to be local machines, IP spoofing, get lost
iptables -A INPUT -i $EXTIF -s $INTNET -d $UNIVERSE -j drop-and-log-it

# remote interface, any source, going to permanent PPP address is valid
iptables -A INPUT -i $EXTIF -s $UNIVERSE -d $EXTIP -j ACCEPT

# Allow any related traffic coming back to the MASQ server in
iptables -A INPUT -i $EXTIF -s $UNIVERSE -d $EXTIP -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
#  OPTIONAL:  Uncomment the following two commands if plan on running
#             an Apache Web site on the firewall server itself
#
#echo -e "      - Allowing EXTERNAL access to the WWW server"
#iptables -A INPUT -i $EXTIF -m state --state NEW,ESTABLISHED,RELATED -p tcp -s $UNIVERSE -d $EXTIP --dport 80 -j ACCEPT


# Catch all rule, all other incoming is denied and logged.
iptables -A INPUT -s $UNIVERSE -d $UNIVERSE -j drop-and-log-it


echo -e "     - Loading OUTPUT rulesets"

##################################################################
# OUTPUT: Outgoing traffic from various interfaces.  All rulesets are
#         already flushed and set to a default policy of DROP.
#

# loopback interface is valid.
iptables -A OUTPUT -o lo -s $UNIVERSE -d $UNIVERSE -j ACCEPT

# local interfaces, any source going to local net is valid
iptables -A OUTPUT -o $INTIF -s $EXTIP -d $INTNET -j ACCEPT

# local interface, any source going to local net is valid
iptables -A OUTPUT -o $INTIF -s $INTIP -d $INTNET -j ACCEPT

# outgoing to local net on remote interface, stuffed routing, deny
iptables -A OUTPUT -o $EXTIF -s $UNIVERSE -d $INTNET -j drop-and-log-it

# anything else outgoing on remote interface is valid
iptables -A OUTPUT -o $EXTIF -s $EXTIP -d $UNIVERSE -j ACCEPT

# Catch all rule, all other outgoing is denied and logged.
iptables -A OUTPUT -s $UNIVERSE -d $UNIVERSE -j drop-and-log-it


echo -e "     - Loading FORWARD rulesets"

##################################################################
# FORWARD: Enable Forwarding and thus IPMASQ
#          Allow all connections OUT and only existing/related IN

iptables -A FORWARD -i $EXTIF -o $INTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT

# Catch all rule, all other forwarding is denied and logged.
iptables -A FORWARD -j drop-and-log-it

# Enable SNAT (MASQUERADE) functionality on $EXTIF
iptables -t nat -A POSTROUTING -o $EXTIF -j SNAT --to $EXTIP

echo -e "    Firewall server rule loading complete\n\n"
```

Not the 'drop-and-log-it' action in the 'catch all' rules. IPTABLES log messages are written to the **`/var/log/messages`** file and also to the 'console' (screen). These messages include the source and destintation address and interface information of **dropped** packets. This is useful in troubleshooting. If your firewall isn't acting the way you thought, you can see which packets are being dropped.

The **`UNIVERSE="0.0.0.0/0"`** line just means "any address".

If you read the script comments you saw there's a commented-out command that you can uncomment if you want to also have your firewall act as a Web server (not a real secure idea). But what if you wanted to set up a separate Web server system *behind* your firewall system? There's three statements that you have to add to the script. We show you how to do that in the **Setting Up A DMZ** section below. (You don't have to set up a full-blown DMZ to use these commands to have servers behind your firewall.)

As with the proxy script, you can simply copy/paste the above script into a text editor and make the necessary changes for your system, network, and type of external connection Then save it using the file name **`firewall.txt`** and anonymous FTP it to your Debian system.

> Note that, also like the proxy script, you cannot set this script to run at boot up if you are using a dynamic IP-based modem connection for the external interface unless you add the commands to call the **`pon`** script.

Once the file is transferred, use the following commands to copy/rename it to the appropriate scripts directory and to make it executable for root:

```
cp /home/ftp/pub/incoming/firewall.txt /etc/init.d/firewall.sh
chmod 755 /etc/init.d/firewall.sh
```

Now all you have to do is connect to your ISP and enter the following command to run the script:

```
/etc/init.d/firewall.sh
```

Using IPTABLES sets up a "packet filtering firewall". It inspects packets for source or destination addresses, protocol (tcp, udp, or icmp), and port numbers (which indicate the type of Internet application being used such as 80 for http (Web browsing), 21 for ftp, 23 for telnet, etc.). There are other, more sophisticated types of firewalls. Those that examine the actual data in the packets to see if what's being transferred back and forth is a logical exchange of information are called "stateful" firewalls.

If you created a symbolic link on the Proxy/NAT page so the proxy script would run at bootup, you may want to delete it and recreate one for this script. The following two commands will take care of that:

```
rm /etc/rc2.d/S95proxy
ln -s /etc/init.d/firewall.sh /etc/rc2.d/S95firewall
```

If you added the commands to the proxy script to call the **pon** dialer you may want to add them to the firewall script also.

Where to learn more - *The best of our bookshelves:*

More info...

Linux Firewalls is extensive in its coverage of, not only firewalls, but other aspects of security as well. The book is crammed with examples of `IPTABLES` chains for different firewall types and security scenarios including DMZs. The book starts out with a good explanation of the basics of TCP/IP, packets, and firewalls in general. As a result, when you do get into the later chapters you'll not only find out how to do things, but you'll know why you're doing them as well. The book focuses on setting up Linux-system type firewalls which is appreciated. Those that cover firewalls in general tend to blur the lines between commercial products and their non-commercial counterparts. However, within this focus the book runs wide as well as deep. I doubt there's anything you'd need to know about firewall-based security that's not in this book. The book also covers IPCHAINS so it provides a good opportunity to compare the syntaxes of the IPTABLES and IPCHAINS commands.

If you don't want a firewall system doing anything but firewalling any open ports represent a potential hole in your firewall. You can check to see if any ports are actively listening for traffic using the command:

```
netstat -an | more
```

If you see any entries with the word **LISTEN** next to them you've got open ports. Typically you'll see them listed with an address like:

```
0.0.0.0:80
```

which would indicate that the server is listening on port 80 (HTTP for a Web server). There are a lot of Web pages out there that list what port numbers are used by which services. You'll want to shut down any services that have ports open on a firewall system. You'll likely have to do this by renaming some of the links (so they don't run) in whichever **/etc/rcX.d** directory matches your default runlevel and then rebooting.

Debian also has an option that helps prevent spoofing. Check your **/etc/network/options** file to make sure it has the line:

```
spoofprotect=yes
```

If it doesn't, edit the file to add it in. Removing any unnecessary applications (packages) is also a good idea. We get into this more on the Securing Servers page. You may have also noticed in the above file the line:

```
ip_forward=no
```

This is because IP forwarding is disabled by default. The line in our firewall script:

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

turns IP forwarding on when the script is run (to allow the proxy function).

If you just want to keep punks off your home or SOHO network a single firewall system is fine. If you want to protect Internet servers you'll want to use a couple IPTABLES firewall systems to set up a DMZ (covered below). However, if your critical intellectual-property or sensitive data is stored on servers on your internal network you'd better look for something more secure. You can do packet filtering with a Cisco router, yet companies will spend tens of thousands of dollars on a Cisco PIX firewall product because it offers greater protection. For Linux systems, you may want to check out the NetMax Firewall Suite or VPN Suite (www.netmax.com). Their products include the Linux software so it's not an application you would install on a Debian system. With NetMax, you remove all existing partitions to create a "bare" hard-drive and pop in the CD to install everything.

Note that it's not advisable to have a serious firewall system also serve up Web pages. Due to it's critical role in the security policy of an

organization, a firewall should be one thing and one thing only, a firewall. This is less of an issue for most home networks since home systems typically aren't powered up 24/7 and the external IP address (assigned by your ISP) isn't static.

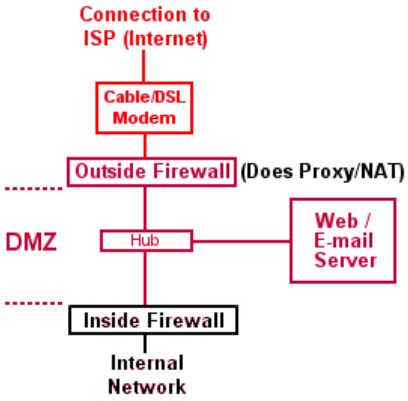
If you want to get an overall view of your firewall, once you run the firewall script, use the command:

```
iptables -L
```

The `L` stands for **L**ist. It gives a comprehensive listing of your rules which is easier to understand than simply looking at the rules themselves.

Setting Up A DMZ

A DMZ allows you place systems that you want the public to access in an "isolation network" between two firewall systems or routers. Recall the DMZ diagram from the Networking page.



The outside firewall is set up to do the proxy/NAT stuff for your internal network, but it also offers two more benefits:

- It allows you to set up some rules to help protect your publically-available systems
- It allows you to use one public IP address to provide access to several different Internet servers

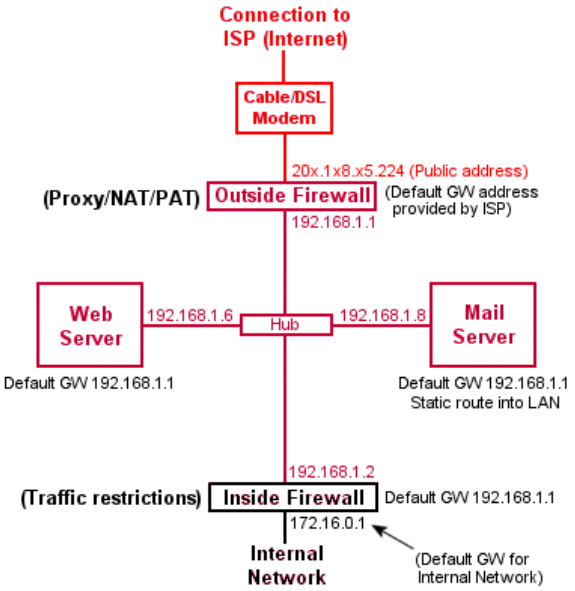
This functionality is called PAT (Port Address Translation). It allows you to specify that any HTTP traffic (port 80) coming into the outside firewall from the Internet be directed to a specific system (Web server) in the DMZ. Any incoming SMTP traffic (port 25) can be directed to a different (mail server) system. The same for FTP (ports 20 and 21), etc.

Why not run all those services on just one box? A number of reasons. Better security (only one service needs to be configured per system), load balancing, finer control over who accesses what from your internal LAN, or you may already have a mail server but you'd like to add other servers along with implementing a DMZ.

If you just want to make a Web or e-mail server that's running on your internal LAN publically accessible, you don't have to set up a full-blown DMZ. Just set up the "outside firewall" system so you have port forwarding. This, however, leaves your LAN systems more vulnerable to a sophisticated hacker than a full-blown DMZ does.

The Outside Firewall

Let's expand the DMZ diagram a bit and see how we would set up the addresses, etc. As you can see, the DMZ is simply set up as a small private network with two gateways.



So how do you accomplish PAT using IPTABLES? You have to use add a few commands to the above firewall script so that the last section of the script looks like this:

```
#####################################################################
# FORWARD: Enable Forwarding and thus IPMASQ
#          Allow all connections OUT and only existing/related IN

iptables -A FORWARD -i $EXTIF -o $INTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT

# Allow forwarding of incoming Port 80 traffic to DMZ Web server
iptables -A FORWARD -i $EXTIF -o $INTIF -d 192.168.1.6 -p tcp --dport 80 -j ACCEPT

# Catch all rule, all other forwarding is denied and logged.
iptables -A FORWARD -j drop-and-log-it

# Enable SNAT (MASQUERADE) functionality on $EXTIF
iptables -t nat -A POSTROUTING -o $EXTIF -j SNAT --to $EXTIP

# Enable DNAT port translation to DMZ Web server
iptables -t nat -A PREROUTING -i $EXTIF -d $EXTIP -p tcp --dport 80 -j DNAT --to 192.168.1.6

echo -e "       Firewall server rule loading complete\n\n"
```

You can use PAT in this manner if you also have Citrix or VPN boxes set up in your DMZ for remote access by employees. That way they can just point their remote client to your domain name (providing it resolves to the public IP address on the external interface of the outside firewall) and you can have the outside firewall forward the traffic onto the appropriate box based on the port number in the incoming packets.

QUIZ! You'll also need to add a static route to your outside firewall. Looking at the above diagram can you think of what it would be needed for? And if you know, what would the correct `route` command be? When you think you've got the answers, drag your mouse over the blank area below to see if you're right:

Note that if you have problems accessing an FTP server that you have set up in the DMZ, make sure your FTP client or browser is set to use PASSIVE FTP. In IE6 there's a "Use passive FTP" checkbox under Tools/Options/Advanced. With the WS_FTP client there's a checkbox under the Advanced tab of the "Session Properties" window.

The Inside Firewall

While the outside firewall is mainly for PAT and masquerading, the inside firewall is where you do most of your traffic restriction. You want to restrict **outgoing** traffic to only those services (port numbers) you want users on the internal network to have, i.e. Web (80), POP (110) to the mail server in the DMZ, FTP (20/21), etc.

Incoming traffic should be restricted to only "established" traffic. This is accomplished in our above script using the command:

```
iptables -A FORWARD -i $EXTIF -o $INTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Remember that this is for the **external** interface of the **inside** firewall system (the interface that connects to the DMZ).

You may also want to come up with some **INPUT** rules for the **internal** interface (the interface that connects to the internal network) which would restrict outgoing user traffic from the internal network to only certain ports or networks. Also remember that there are no NAT rules on this inside firewall system. All the NAT stuff is done on the outside firewall system.

Getting the right rules on the right boxes is a balancing act. It all depends how you want to filter things. You can put traffic restrictions on your outside firewall too (limiting incoming traffic to Web, mail, and FTP traffic for instance). However, you wouldn't want to put the above established-traffic rule on the external interface of your oustide firewall or no one (or no system) on the Internet could ever create a connection to your Web or mail or other DMZ-located systems.

Troubleshooting

If something isn't getting through our firewall you have to find out if the firewall is stopping it or you have some other configuration issue. The fact that logging to the console of dropped packets is enabled by the script you can tell right away if the firewall is stopping the traffic. Packet information will be displayed on the screen. This indicates that your rules aren't correct for what you're trying to accomplish because the firewall is dropping the packets you want to go through.

If traffic isn't getting through but you're not getting any packet information displayed on the screen, it's likely something with the configuration of the system behind the firewall. **Don't assume!** I beat my brains out for three hours trying to figure out why I wasn't getting a response from my Web server located behind the firewall. No packet information was being displayed on the screen (i.e. the firewall wasn't dropping anything).

It ended up that I did not have the firewall's internal NIC address entered as the default gateway on the **Web server system**, not the firewall. When the Web server gets a request from a browser on the Internet, it will try and respond to the address that was given as the source address in the request. This source address will be that of the system that's trying to browse to your server over the Internet. This address won't be on the DMZ network, and without a valid gateway address specified in the `/etc/network/interfaces` file, the Web server won't know how to get to the non-local system.

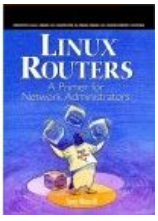
Linux As A Router

Cisco's claim to fame is not that they have routers that will route TCP/IP. It's that they have routers that will simultaneously route TCP/IP, IPX/SPX (Novell), AppleTalk, and a few others. These are the protocols that computers use to transfer data between each other and these are the protocols that routers route. These are called rout**ed** protocols. Another big benefit of Cisco routers is that they will "talk" to one another using rout**ing** protocols. Routing protocols enable routers to update each others routing information so they can automatically adjust to changes in the network (when links go down for example). Using routing protocols to automatically adjust for network changes is known as "dynamic routing".

However, if your internal network isn't very complex and you're strictly using TCP/IP you don't need all the features that Cisco routers offer. You can set up a Linux system with two or three or even four network cards and set it up to be a router rather than spending $5,000 on a Cisco router. In cases where a network isn't complex, there's really no need to run any rout**ing** protocols. Any necessary changes made to the network's architecture are rare and can be compensated for by manually changing the routing information. When referring to Cisco routers, manually maintaining the routing information is done using "static routes".

Linux has a `route` command which lets you enter static routes also. However, no checking of the packets is done. Whatever comes in gets sent out. The benefit of using IPTABLES commands on a per-interface basis to do your packet forwarding is that you can do rules-based routing.

Where to learn more - *The best of our bookshelves:*



More info...

The title of **Linux Routers - A Primer for Network Administrators** is misleading. It's way more than a "primer". It gives detailed listings of `route` and `IPTABLES` commands for setting up LAN, Internet, extranet, and satellite office routers. While it doesn't focus on any one distrbution, luckily Debian is the author's distro of choice so the examples in the book are based on it. Not only does the book give details on setting up frame relay and ISDN routers, but the author gives vendor information on where to obtain frame relay and ISDN capable PCI and ISA cards. He also covers how to use LRP (the Linux Router Project software that allows you to fit everything on a floppy disk) and has a very detailed chapter on masquerading.

In the firewall script above notice that the IPTABLES commands had referenced the different interface designations. The ability to apply rules to specific interfaces is the same thing that Cisco does with the IOS (Internetwork Operating System) software that runs on its routers. Packet filtering is a two step process with the Cisco IOS. You set up ACLs (Access Control Lists) and then apply them to the appropriate router interface(s).

If you set up a Linux box with three NICs their designations would be eth0, eth1, and eth2. Each interface (NIC) would be connected to a different subnet. You would then use IPTABLES commands to allow forwarding or denying of traffic between the NICs (and their

connected subnetworks). Yes, it would be quite a sophisticated script, but it would be easier to learn IPTABLES and figure out what rules would be needed to accomplish your goals than it would be to learn the Cisco IOS. And once you did learn the Cisco IOS, and pay big bucks for a router, you'd still have to develop similarly-sophisticated ACLs.

For those that may be well-versed in routers, and would like to play around with Linux in a more complex routing environment, there is a server application known as **gated** that allows a Linux system to work with dynamic routing protocols like RIP and OSPF.

SECURITY WARNING

Do NOT plan to use the system you will create using these guide pages as a "production" (real) server. It will NOT be secure!

There are many steps involved in creating a secure Internet or LAN server. While we do refer to some things you can do to make your system more secure, there are many other measures related to system security that also need to be taken into consideration and they are not covered on these pages.

These guide pages are meant as a learning tool only. The knowledge gained on these pages will help you understand the material covered in security-related publications when you are ready to consider setting up a production server.