How To Set Up A Debian Linux Proxy Server

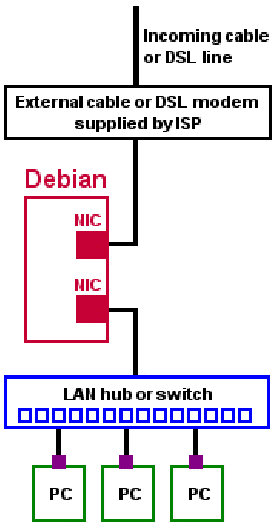
The material on this page was prepared using **Sarge** or **Etch**
configured using our Installation and Packages pages.
If you did not use our pages to set up your system, what you
encounter on your system may be different than what is given here.

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.

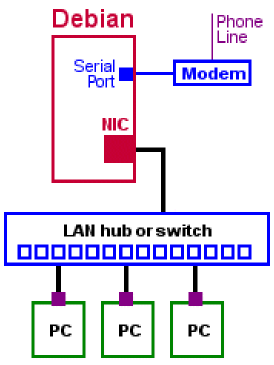
Back on the Networking page we covered the basics of the "what", "where", and "why" of a proxy server, and the reason NAT ("masquerading" in Linux-ese) is needed, as well as how to configure the Linux networking files to get your system operational on a network. This page will show you how to turn your networked system into a proxy server.

Recall also from the Networking page that a proxy server is a "dual-homed" system. In other words, it needs two network interfaces. The "internal" interface (a NIC card) connects to the internal LAN and the "external" interface connects to the outside network (typically the Internet). The external interface can be a NIC card which connects to a cable or DSL modem, or you can simply use a dial-up modem as your external interface. (We showed you how to get a dial-up modem working back on the Modems page.) The script below will work in either instance.

Sharing Your Broadband Connection


Incoming cable or DSL line
External cable or DSL modem supplied by ISP
Debian
NIC
NIC
LAN hub or switch
PC
PC
PC

Sharing a Modem Connection



Naturally, the dial-up modem shown in the diagram above could also be an internal PCI or ISA slot modem. In any event, be sure you can access Internet resources from the Debian system itself using your cable, DSL, or dial-up modem *before* proceeding to add proxy server functionality to the system.

***** WARNING ***** A proxy server is fine if you plan to share a dial-up (modem) connection. However, if you have an "always on" DSL or cable connection a proxy server can put your systems at risk because **it offers very little protection against outside intrusion into your network.**

Because of this, you should consider this material on proxy servers an introduction, not a solution. **It's very easy to go from a proxy server to a firewall system that also does NAT.** All you need to do is simply add a few more IPTABLES commands to the `proxy.sh` script to turn it into a `firewall.sh` script. We'll do just that on the Firewall page.

In addition to being more secure, the firewall script can be modified to limit access based on protocol (TCP/UDP), ports (port 80 for Web and 25 for mail), the address of the user's machine, or the address of a Web site you wish to block access to. It also has a command for those who wish to have the firewall system simultaneously act as a Web server to host a family Web site or serve up

Web cam images.

> **Note:** You may see the "Squid proxy server" product mentioned in various Web pages, books, and articles about Linux proxy servers. You don't need Squid or any other product to set up a Linux proxy server. The advantage of Squid is that it's a *caching* proxy server product. The caching function will store cached copies of frequently visited Web pages on the proxy server's local hard-drive and serve those up to requesting browsers rather than bringing a fresh copy over the Internet. You'd need quite a large number of users on your network to see any benefit of using a caching proxy server.

The Kernel and NAT

It is the Linux kernel that inspects the packets and modifies the addresses in the packet headers (the NAT function) before forwarding them on to the final destination system (whether that is an Internet server for outgoing packets or a network workstation for incoming packets).

We need a way to tell the kernel how to handle packets. With the Linux 2.2 kernel (Potato and Woody) we used the IPCHAINS command to do that. With the Linux 2.4 kernel (Sarge) IPTABLES is used. We issue a series of IPTABLES commands with each command establishing a "rule" which dictates how packets should be handled (which should be forwarded and which should be dropped). The series of rules that is built using a series of IPTABLES commands is called a "ruleset" or "chain".

A ruleset only exists in memory so when you reboot the system it disappears and has to be recreated. If you had to enter the necessary IPTABLES commands manually every time you rebooted your system it would be a lot of repetitious work. Instead, we put the commands in a shell script. We can optionally set the script file to get executed automatically at boot up if this is going to be a full-time proxy server. We'll see how to set that up later in this page.

The following is a shell script that was derived from the Linux 2.4 kernel script in the IP Masquerade HOWTO. If you're viewing this page on a Windows PC the simplest way to get the below script is to drag over it to select it, copy it to your clipboard, and paste it into Notepad. You can then save it and ftp it to your server.

At first glance the script looks long and hideos but that's only because we've gone overboard with the comments. Most shell scripts aren't this heavily commented. To set this script up you'll need to:

Section A
- Enter your internal interface designation (`INTIF`)
- 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

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.

```
#!/bin/sh

#  IPTABLES  PROXY  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/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 proxy server.
#
#    PLEASE SET THE USER VARIABLES
#    IN SECTIONS A AND B OR C

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


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

# SET THE INTERFACE DESIGNATION 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 internal interface's designation for the
#   INTIF variable:

INTIF="eth0"


# 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.


# === 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 proxy server rules..."

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

# FWD: Allow all connections OUT and only existing and related ones IN
iptables -A FORWARD -i $EXTIF -o $INTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT

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

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

The `ESTABLISHED` keyword in the rule that fowards packets from the External (Internet) interface to the Internal (LAN) interface limits incoming traffic to that which is a "response" to a previously sent outgoing request (a Web page coming back from a browser request for example). The rule with the `MASQUERADE` keyword is the rule that causes the actual NAT translation.

Once you've pasted the script into a text editor be sure to read through the file's comments and make any necessary changes to the file. Once you've done that, save the file as **`proxy.txt`** to your local hard-drive. (Using a `.txt` extension avoids hassles when saving the file with NotePad and will help ensure that your ftp program transfers the file using ASCII mode rather than binary. We'll change the extension later.)

> One note of interest. Recall that back on the Linux Basics page we mentioned how you can use "piping" to pass the output of one command into another. In the above script the long command immediately above the line
> `# -------- No more variable setting beyond this point --------`
> has three pipes which passes output among four different commands (ifconfig, grep, awk and sed). This series of commands will extract the IP address assigned to a modem (or modem-like device) so that it can be assigned to the `EXTIP` shell script variable.

If we were to remove all of the comments from the above script we find that the actual script itself isn't very long. And most of what remains are the commands which echo imformational messages to the screen which indicate the progress of the script. If we were to remove those also, the final script for those who use an external interface that receives a dynamically-assigned IP address would only be:

```
#!/bin/sh
INTIF="eth0"
EXTIF="ppp0"
EXTIP="`/sbin/ifconfig ppp0 | grep 'inet addr' | awk '{print $2}' | sed -e 's/.*://'`"
/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
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
iptables -P INPUT ACCEPT
iptables -F INPUT
iptables -P OUTPUT ACCEPT
iptables -F OUTPUT
iptables -P FORWARD DROP
iptables -F FORWARD
iptables -t nat -F
iptables -A FORWARD -i $EXTIF -o $INTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT
iptables -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE
```

Once you make the necessary changes and save the file you can use an ftp program to transfer the script file to your Debian system. Be sure to use the **ASCII** transfer mode. (We showed you how to set up ftp server functionality back on the Packages page.) Use anonymous ftp to transfer the file (where the username is "anonymous" and the password is an e-mail address). This is puts the file in **`/home/ftp/pub/incoming`** directory on the Debian system so it's easy to find.

After connecting to your Debian system, use the ftp client to navigate into the **pub** directory and then into the **incoming** directory. This is the directory where anonymous ftp visitors have write access. When you transfer the file (using ASCII mode), it may not show up in the list of files on the "remote" (Debian) system, but it is there. (You can verify that by trying to transfer it a second time. You should get an error saying you don't have overwrite permission.)

Once the file is transferred, enter the following commands on your debian system to copy/rename the file to the appropriate scripts directory and to make it executable for root:

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

The `.sh` part of the new file name just indicates that it's a shell script. Not all shell scripts use an extension but this makes it easier to identify your scripts.

Testing Your Proxy Server

If you set the script to use the `ppp0` as the external interface (i.e. set the `EXTIF` variable equal to `ppp0`), use the `pon` command to connect the modem to your ISP. You must have an active connection for the `ppp0` interface to exist, and this interface must exist before running the script.

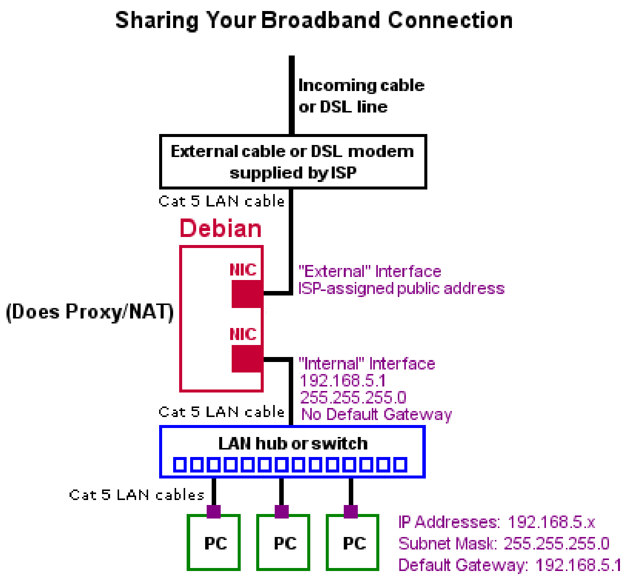
> **Note:** You cannot set this script to run at boot up if you are using a dynamic IP-based modem connection for the external interface such as with a dial-up modem or residential DSL or cable service without some modification (which we'll see below).

Once connected to the Internet, run the script by entering the command:

```
/etc/init.d/proxy.sh
```

You should see the script messages echoed to the screen as it executes. Congratulations! You now have a proxy server.

Try it out. All you have to do is go to one of the other systems on your internal LAN and change it's "default gateway" value to the IP address of the Debian system's **internal** interface. The following diagram from the Networking page shows you what's happening when you do. (The "cable or DSL modem" in the diagram could be your dial-up modem connected to a serial port if that's what you're using.)


Sharing Your Broadband Connection
Incoming cable or DSL line
External cable or DSL modem supplied by ISP
Cat 5 LAN cable
Debian
NIC
"External" Interface
ISP-assigned public address
(Does Proxy/NAT)
NIC
"Internal" Interface
192.168.5.1
255.255.255.0
No Default Gateway
Cat 5 LAN cable
LAN hub or switch
Cat 5 LAN cables
PC
PC
PC
IP Addresses: 192.168.5.x
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.5.1

In the above diagram, you'd set the default gateway on your network system to 192.168.5.1 which is the IP address of the internal interface on the Linux proxy server. If you're using a Windows PC, you would go into Start/Settings/Control Panel/Network and open the TCP/IP properties to change the default gateway value. Also make sure the DNS server setting points to your ISP's DNS server. You may have to reboot your Windows system after making these changes.

Once your Windows system has been changed and rebooted, open up a DOS window and try a trace route by entering the following command:

```
tracert www.debian.org
```

The first thing you should see shows you that the `www.debian.org` got resolved to an IP address which means you were able to access your ISP's DNS server to resolve the domain name. Next your should see a series of trace responses. Look at the IP address on the right end of the **first** trace response. It should be the IP address of your Debian system's internal interface. Here's the trace output from my system:

```
Tracing route to www.debian.org [194.109.137.218] over a maximum of 30 hops:

 1    <10 ms   <10 ms   <10 ms   sarge 192.168.5.1
 2    150 ms   140 ms   151 ms   as29.nwbl0.myisp.net
 3    150 ms   141 ms   140 ms   vl15.rsm0.myisp.net
 4    150 ms   141 ms   140 ms   3.ge3-0-0.rtr1.myisp.net
 5    151 ms   140 ms   140 ms   0.rtr0.chcg0.il.myisp.net
 6    150 ms   150 ms   140 ms   pvc-von.225io.myisp.net
 7    140 ms   140 ms   150 ms   206.220.243.189
 8    211 ms    200 ms    200 ms  oc3-pos0-0.gsr12012.sjc.he.net
 9    180 ms    190 ms    191 ms  gige-g0-0.gsr12008.pao.he.net
10    190 ms    191 ms    190 ms  fe-1-1-0.pao.via.net
11    191 ms    200 ms    190 ms  s6-0.border1-7206.valinux.com
12    200 ms    190 ms    190 ms  fe0-0.dist5-3662.vasoftware.com
13    190 ms    180 ms    201 ms  e2-1.community8-bi8000.vasoftware.com
14    211 ms    200 ms    200 ms  klecker.debian.org

Trace complete.
```

If the trace doesn't work, try bypassing DNS by tracing to the debian.org server using the IP address:

```
tracert 194.109.137.218
```

If this works, and the above didn't, it indicates your DNS settings (on your Windows system) are not correct. In any case, the IP address of the internal interface on your Debian system should be the first line in any trace response. If it's not, check your TCP/IP properties again. If you don't get any response, see if you can ping the internal interface of your Debian system with the command:

```
ping 192.168.5.1
```

If you can't ping that, see if you can ping any other systems on your LAN (you'll have to use the `ipconfig` or `winipcfg` command on another Windows system to find out what its IP address is in order to ping it). If you can ping a different system on your LAN, something is wrong with the network configuration on your Debian box. If you can't ping any other system, something is wrong with the network configuration (likely the TCP/IP properties) on the system you're using to do the pinging.

With the above script, there is no easy way to "turn off" the proxy function once you use the `proxy.sh` script to turn it on. Simply reboot the system to stop it from acting as a proxy server.

In the next section you'll see how to get the proxy/firewall functionality to start up automatically when you boot the system.

Automatic Startup

Remember that you cannot run a proxy server or firewall script at system startup if your Internet connection needs to be dialed. That's because the script looks for the IP address your ISP has assigned to the external interface, and the external interface won't have an IP address until after a connection is made. (However, there may be a way around this.)

In the case of a real, full-time proxy server, firewall, or router, you'll want to set things up so the appropriate script gets run automatically when you boot the system. Back on the Linux Basics page we covered the Debian startup process. Recall from that discussion that we not only need to put the scripts in the `/etc/init.d` directory, which we have already done with the `proxy.sh` script, but we also need to create the appropriate links to the scripts.

There's no way you should ever have NFS file sharing enabled on an Internet-connected system so runlevel 2, which is Debian's default, is appropriate. As such, we'll need to create a symbolic link in the `/etc/rc2.d` subdirectory. Recall that the name of this link needs to start with an upper-case **S** which needs to be followed by a two-digit number. In order to do this we just need to know how to create a symbolic link. For that we use the `ln` command.

If we do a:

```
man ln
```

to pull up the man page for the `ln` command we see the syntax is:

```
ln [OPTIONS] TARGET LINK_NAME
```

We want to use the `-s` option to create a **s**ymbolic link. As a result, our command would be:

```
ln -s /etc/init.d/proxy.sh /etc/rc2.d/S95proxy
```

The **95** part of the link name ensures that the `proxy.sh` script won't be run until near the end of the startup process. If you go into the `/etc/rc2.d` subdirectory and do an `ls` you will see that 95 wasn't already being used. Also, take note of the `S91apache` link. This was created when we installed the Apache package.

Auto-Dialing

So what if all we have is a dial-up connection to the Internet? Are we sunk? No. If you ever worked with DOS batch files you may recall the PAUSE command. You could put the PAUSE command in a batch file followed by a number to get the execution of the program to idle for the number of seconds equal to the number after the command. Naturally Linux has an equivalent. It's the `sleep` command.

Rather than run the `pon` dialer script manually, we can "call" it near the beginning of an proxy or firewall script and then follow that call with a `sleep` statement to give the modem a chance to connect. The modifications would look like this (in blue):

```
#     PLEASE SET THE USER VARIABLES
#     IN SECTIONS A AND B OR C

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


echo "    Dialing Internet connection..."
/usr/bin/pon
sleep 15


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

Naturally, if you don't get a connection for some reason the setup of the proxy server or firewall will fail so it's best to monitor the bootup messages to make sure everything initializes OK.

Since most ISPs don't allow "camping" on a modem line (staying connected for long periods of time) this setup isn't something you'd want to use permanently. However, it will let you simulate having an "always on" connection to do a little playing around.

In order to stop the auto-dialing at boot-up, simply delete the symbolic link with the command:

```
rm /etc/rc2.d/S95proxy
```

You can also delete the dialing-related statements out of the proxy or firewall script if you want, but having them auto-dial for you is a convenience even when you do want to run the scripts manually.

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.