Home−Network−mini−HOWTO

Table of Contents

Red Hat Linux 6.X as an Internet Gateway for a Home Network

Paul Ramsey

January 22, 2000

A simple tutorial on configuring Red Hat 6 and related variants to operate as an internet gateway to a small home or office network. Topics covered include masquerading, DNS, DHCP, and basic security.

5.Problems

1.Introduction

This page contains a simple cookbook for setting up Red Hat 6.X as an internet gateway for a home network or small office network. The instructions are very simplified: no special cases will be discussed, and some assumptions will be made about which network addresses are to be used. The most important assumptions are:

* You have a fulltime Cable or ADSL connection to the Internet.
* You can successfully install Red Hat 6.X on at least one of your computers. Note that these directions are also valid for Red Hat derivatives, such as Mandrake 6.X which is distributed by MacMillan Publishing under a variety of labels.
* Your Linux computer has two network cards installed in it and both are compatible with Linux.
* You have an ethernet hub if you are networking more than one computer or a cross−over cable if you are only networking one computer.
* You know how to edit text files on your Linux machine.
* You can log into your machine as `root`. You know how to install RPM packages from your Linux CDROM.

If you do not meet any of these assumptions, then this document probably isn't for you.

There is nothing special that you have to do during the installation process. Simply choose an installation which makes sense for you and go for it. This document gives directions on installing everything to do with networking from scratch, to avoid making any assumptions about what was installed or configured during installation. To ensure that things work and there is no confusion about what information goes where, all the configuration will be done by directly editing the system configuration files rather than using the GUI configuration tools provided with Red Hat. On the one hand, this might be a little harder than it has to be; on the other hand, your knowledge will be a good deal more transferable to different distributions and situations (like, what if X doesn't work, or you are setting up a headless server).

1.1 Versions

The latest version of this document should always be available at http://www.coastnet.com/~pramsey/linux/homenet.html for the HTML version and http://www.coastnet.com/~pramsey/linux/homenet.sgml for the SGML version.

* December 21, 1999 : First version.
* January 2, 2000 : Incorporated suggestions from John Mellor on outside networking quirks.
* January 22, 2000 : Minor update about identical network cards and info on IP aliasing from Chris Lea.

1.2 Copyright

2.Plugging Things In

Depending on whether you are using a hub or not, your network topolgy will differ slightly. I am only covering networking with RJ45 cabling (the stuff that looks like phone cables on steroids) and not covering thin coax. With thin coax you can network multiple machines without requiring a hub, but have to be more careful about terminating connections and so on. If you know networking already, these instructions will be largely redundant.

2.1 With a Hub

If you have a hub, your network will look like this.

Connect the `eth0` card on the Linux box to the cable modem or ADSL box using the cable supplied by the service provider during their install (or one you know works in that configuration. This is important because sometimes cable modems like to be connected with a crossover and sometimes they like a straight−through through cable: the one the company gives you is the one you want to use.

Connect the `eth1` card on the Linux box to the hub with a straight−through cable. Connect all your other computers to the hub with straight−through cables.

2.2 Without a Hub

If you do not have a hub, you can still connect one computer to your Linux box, using a crossover cable. Your topology will look like this.

Connect the `eth0` card on the Linux box to the cable modem or ADSL box using the cable supplied by the service provider. Connect the `eth1` card on the Linux box to the other computer with a crossover cable.

2.3 With Only One Network Card

This is not a recommended configuration (in this configuration your internal and external networks are on the same physical address, and are therefor theoretically more susceptible to cracking; in reality, the risk is probably very low), but it *can* be done. Your mileage may vary.

The Linux kernel includes support for "IP aliasing", which allows an ethernet card to service two different IP addresses simultaneously. The stock kernels shipped with Red Hat and Mandrake do include support for IP aliasing by default. To set up your gateway with only one ethernet card, in all the subsequent code examples, simply replace `eth1` with `eth0:0`.

In a single−card situation, running a DHCP server is not *recommended.*

Plug all your machines and your cable model or ADSL box into the hub. Cross your fingers and continue.

3.Configuring Networking

OK, by now you have installed Linux on your gateway computer. You may have even configured one of your networking cards, and set up connectivity to the Internet. However, we are going to start from scratch and pretend that nothing is configured at all.

Log in as `root`. All the instructions given in this document assume you are logged in as root.

The Linux kernel refers to your two ethernet cards as `eth0` and `eth1`, so that is how I'll be referring to them from now on too. The trouble is, which one is which? Here's a "simple" way of figuring out, guaranteed to work at least 50% of the time: lay your computer on the desk with the motherboard horizontal and the back panel facing you (as you would if you were going to open it and do some work on it). The leftmost card is `eth0` −− you might want to label it with some masking tape. Now, write down on a piece of paper the make and model of both `eth0` and `eth1`.

OK, let's see if `eth0` and `eth1` are recognized automatically by the kernel. Type `ifconfig eth0` and `ifconfig eth1`. In both cases, if the kernel is recognizing your card, you should see something like this (bearing in mind that the numbers and whatnot will be different):

```
eth0   Link encap: Ethernet   HWaddr 00:60:67:4A:02:0A
       inet addr:0.0.0.0  Bcast:0.0.0.0  Mask:255.255.255.255
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:466 errors:0 dropped:0 overruns:0 frame:0
       TX packets:448 errors:0 dropped:0 overruns:0 carrier:0
       collisions:85 txqueuelen:100
       Interrupt:10 Base address:0xe400
```

If the kernel is not recognizing your network card you will see something like this:

```
eth0: error fetching interface information: Device not found.
```

3.1 Configuring a Network Driver

If both of your cards were found, skip to the next section. Otherwise, read this section.

OK, so one or both of your cards are not recognized by the kernel. This is not a problem, really. What we're going to have to do is tell the kernel more explicitly how to find your cards. There are lots of twists and turns here, and I'm not going to cover all of them. Remember, when the going gets tough, the tough turn to the Ethernet HOWTO. Here's some summary advice:

- *You have a PCI network card.* You are probably sitting pretty, assuming it is not so new and cutting edge that no drivers exist. You can often find out a great deal about your network cards (and other things) by reading through `/proc/pci` and noting down makes and models.
- *You have an ISA network card.* It is possible you will have to know the IO base address and the IRQ the card is operating on. You have manuals, right? Right? If not, this would be a good time to surf to the manufacturer's web site and see if they have any online references. Or if you have an old DOS configuration diskette, boot to DOS and see if there is a setup program which will read and set the address and IRQ.
- *You have an ISA Plug'n'Play card.* You'll have to learn how to configure it first −− read the Plug'n'Play HOWTO. Fortunately, once you've configured your card you will know exactly what the IO base and IRQ are.

Now, since you know what the make and model of `eth0` and `eth1` are you can go to the compatibility page of the Ethernet HOWTO and look up your card. Take note of the recommended driver, and any information about special options your card may require. Write it down.

It's time to edit a configuration file! The file we will be editing is `/etc/conf.modules`. Open this file up in the text editor of your choice. Because there are so many possibilities and combinations of things which can go in this file, I'm going to give my own gateway as an example. I have a PCI 10/100Mb card based on the VIA Rhine chip, and a plain−jane 10Mb NE2000 ISA clone. I use the 100Mb card for the internal network and the 10Mb card for the external connection. My `/etc/conf.modules` file looks like this:

```
alias parport_lowlevel parport_pc
alias eth0 ne
options ne io=0x300 irq=10
alias eth1 via−rhine
```

My `conf.modules` file is laid out as follows:

- The first line is there to configure my parallel port for printing. You probably have a similar line. Leave it alone.
- The second line (`alias eth0 ne`) tells the kernel to use the ne driver for the `eth0` device.
- The third line (`options ne io=0x300 irq=10`) tells the ne driver at which io address and irq interrupt it will find the ISA card at. If you have ISA cards you will probably have to use this kind of directive, just replace the driver, io and irq directives with the correct information for your card.
- The fourth line (`alias eth1 via−rhine`) tells the kernel to use the via−rhine driver for `eth1`. Because my `eth1` card is a PCI card, I do not need to provide io or irq information: the PCI subsystem configures the device automatically.

You will want to ensure that you have alias entries in `conf.modules` for both your cards, and correct options lines for all your ISA cards. You may already have lines in `conf.modules` for any ethernet cards you configured during installation.

When you have finished editing `conf.modules`, try `ifconfig eth0` and `ifconfig eth1` again. You may have to apply some trial and error if you are messing with IO addresses and IRQs without a manufacturers manual.

Two Identical Network Cards

So, you were really really smart, bought two identical network cards for your Linux gateway, and now you cannot get them to work together? Do not worry, getting them to coexist is just a matter of using the correct syntax in `/etc/conf.modules`. For this example, the addresses and IRQ numbers are made up, and I will assume that you have bought a matched pair of NE2000 clones (a common choice). Your `/etc/conf.modules` file should look like this:

```
alias eth0 ne
alias eth1 ne
options ne io=0x330,0x360 irq=7,9
```

The addressing options are all given on the same line, and the first number for each addressing type is for `eth0`, the second number for `eth1`.

3.2 Configuring the Inside Network

The "inside network" is the network which all your home/office machines will talk on. The "outside network" is the big scary internet on the other side of the Linux box. By and large, the inside network will be completely insulated from the outside network by the Linux box, which will operate as a medium strength firewall.

The Network Device

Now that your drivers are working and you can see both `eth0` and `eth1` in `ifconfig` it is time to set up the internal home network. I am assuming that you are going to put your internal network on `eth1` and your external device on `eth0`.

Your internal network is going to be a private network and will therefor be on a special network reserved for internal networking: `192.168.1.0`. This is a "private Class C network", in case you want to impress your friends.

First we need to make sure networking is turned on. Edit the file `/etc/sysconfig/network` and make sure the following lines exist:

```
NETWORKING=yes
FORWARD_IPV4=yes
```

The first line tells Linux that we want the network devices brought up at boot time. The second line tells Linux to enable IP forwarding. This is required when we start configuring masquerading in Section 4.

All the network interface settings for Red Hat and Red Hat derivatives are contained in files in the `/etc/sysconfig/network−scripts` directory. Enter that directory, and create a new file `ifcfg−eth1`. Put the following into the `ifcfg−eth1` file:

```
DEVICE=eth1
IPADDR=192.168.1.1
ONBOOT=yes
```

This code tells the networking scripts to configure eth1 at boot time and to give it a particular IP address. Activate your network with the new settings with the following command:
`/etc/rc.d/init.d/network restart`

The DHCP Server

A DHCP server will automatically configure devices on your internal home network with IP addresses. This is very useful for people with laptops: they can simply plug their machines in and be immediately properly configured. If you do not want a DHCP server on your internal network, just skip to the next section.

First you need to be sure you have the DHCP server installed. Mount your Linux CD and install the `dhcp` RPM. Now edit the `/etc/dhcpd.conf` file and put the following (and only the following) in it:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.2 192.168.1.60;
  default−lease−time 86400;
  max−lease−time 86400;
  option routers 192.168.1.1;
  option ip−forwarding off;
  option broadcast−address 192.168.1.255;
  option subnet−mask 255.255.255.0;
}
```

If you are going to set up your Linux box as a caching domain name server, insert the following option:

```
option domain−name−servers 192.168.1.1;
```

If you know your outside DNS addresses and you are *not* going to use the Linux box for DNS, insert the following option, where x.x.x.x and y.y.y.y are IP numbers of the DNS servers:

```
option domain−name−servers x.x.x.x, y.y.y.y;
```

If you are going to run Samba file sharing on the Linux box for your Windows computers, add the following options to use the Linux box as the default WINS and browsing server:

```
option netbios-name-servers 192.168.1.1;
option netbios-dd-server 192.168.1.1;
option netbios-node-type 8;
option netbios-scope "";
```

Configuring Samba and WINS is well beyond the scope of this document. If you need some pointers, start with the SMB HOWTO and go on from there.

There are still a few more steps. Next, edit the `/etc/rc.d/rc.local` file and add the following lines at the end:

```
# Set up the broadcast device to support DHCP
/sbin/route add -host 255.255.255.255 dev eth1
```

Windows DHCP clients require a particular broadcast address in DHCP responses, and this command forces the Linux TCP/IP stack to produce it.

The next step is to alter the `/etc/rc.d/init.d/dhcpd` file to use `eth1` as the default device. Replace the line:

```
daemon /usr/sbin/dhcpd
```

With:

```
daemon /usr/sbin/dhcpd eth1
```

OK, now we are ready to start up DHCP. First start the DHCP server with the command: `/etc/rc.d/init.d/dhcpd start`. Now add the special broadcast address by invoking `/sbin/route add -host 255.255.255.255 dev eth1`.

Finally, we have to make sure that the DHCP server will start at re−boot time. Some RPM packages of the DHCP server do not include directives to ensure the server starts every time, so we'll make sure it gets started by invoking the command `chkconfig dhcpd on`.

This command causes RedHat to add the dhcp startup script to the various runlevel directories under `/etc/rc.d`. In runlevels 3 and 5 (multiuser console and multiuser X) the DHCP server is started. In runlevels 0, 1 and 6 (shutdown, single user and reboot) the DHCP server is stopped.

The Client Computers

If you have set DHCP up, configuring your client computers is very easy: just enable DHCP configuration. For Windows computers, this involves opening the "Control Panel" and then the "Networking" option. Find the "TCP/IP" protocol and opt to "Configure" it. Check the box that says to "Configure TCP/IP address automatically", apply your changes, and reboot.

Before you reboot, you might want to type the following command: `tail -f /var/log/messages`.

This will watch the Linux system log continuously. If all goes well, when you reboot your Windows computer, you will see it request an IP address and see the DHCP server respond. Control−C exits the `tail −f` command.

If you have not set up DHCP, configuration is still fairly easy. Again, open the "Networking" option from the "Control Panel", and choose to configure the TCP/IP protocol. You can assign your client computers any address in the 192.168.1.0 network except 192.168.1.0 (the network address), 192.168.1.255 (the broadcast address) or 192.168.1.1 (your Linux server). Never give two computers the same IP address. Set the "Gateway" address to 192.168.1.1, so that outgoing traffic is routed through your Linux gateway.

The IP Masquerading HOWTO has very detailed information on client configuration in the Configuration Section.

The DNS Server

Setting up your Linux box as a caching DNS server will (slightly) improve your netsurfing speed, because commonly used DNS addresses will get cached inside your network and not have to be retrieved from the outside.

If you are interesting in doing full blown DNS, there is a great deal of complexity to be learned. There is a DNS HOWTO available, and the book DNS and BIND is a good (and very comprehensive) paper reference.

In order for your client machines to take advantage of the caching server, they must be configured to use the Linux gateway as their primary DNS server. The DHCP directives given in section 3.2.2 are one way to accomplish this. If you are configuring your client computers by hand, you can change the DNS configurations in the same control tabs you used to set the IP address of the machine.

To install the DNS server, first install the `bind` RPM, then install the `caching−nameserver` RPM. At this point, you are almost ready.

As installed, the caching server will work fine, but if you know the IP addresses of the internet providers DNS servers you can improve performance slightly by editing the `/etc/named.conf` file and adding the following line after the `directory` line (where x.x.x.x and y.y.y.y are the primary and secondary DNS servers):

```
forwarders { x.x.x.x; y.y.y.y; };
```

This change makes your DNS server first query the ISPs DNS servers before traversing the internet in search of a given address. The ISPs servers often have a rich cache of DNS information and can provide a much faster answer than your server could.

OK, now you can start your DNS server: `/etc/rc.d/init.d/named start`

Testing the Inside Network

Until we configure the outside network, the DNS service will not work (since it has to communicate with other DNS servers on the internet), but we can test out the basic internal connectivity with the `ping` program.

On one of your client computers, open up a terminal (MSDOS) window, and type `ping 192.168.1.1`. This will send out packets to your Linux computer at regular intervals, and your Linux computer will reflect the packets back. If things are working right, you should see a set of packet return times.

3.3 Configuring the Outside Network

Now we're ready to configure the outside network. Sometimes this will be difficult, depending on how well your internet provider supports Linux. If you have difficulty, there is an ADSL mini−HOWTO which covers ADSL issues in some detail. If I can find a Cable Modem HOWTO, I will link to it also.

The main problem with most outside connections is *getting an IP address*. Some internet providers hand out static IP addresses to cable or ADSL subscribers, and in that case configuration is easy. However, most providers have now moved to dynamic configuration via (you guessed it) DHCP. This means that your Linux computer will likely be a DHCP *server* on your `eth1` interface, and a DHCP *client* on your `eth0` interface.

Additionally, many providers have taken to providing their services in specialized non−standard ways which assume their customers will be using Windows. Some of those cases will be discussed at the end of section 3.3.2.

With a Static IP

If your internet provider has assigned you a static IP address, you are sitting pretty. First, create a new interface configuration file, `/etc/sysconfig/network-scripts/ifcfg-eth0` and put the following in it:

```
DEVICE=eth0
IPADDR=x.x.x.x
NETMASK=y.y.y.y
ONBOOT=yes
```

Just fill in x.x.x.x and y.y.y.y with the values given by your internet provider. Now edit the `/etc/resolv.conf` file and enter the following information:

```
search provider_domain_here
nameserver n.n.n.n
nameserver m.m.m.m
```

The provider_domain should be supplied by your internet provider. Also enter the primary and secondary DNS servers in the n.n.n.n and m.m.m.m lines. If you have set up the Linux box as a DNS server, you can add a line before the other nameserver entries: `nameserver 127.0.0.1`. This will make your Linux server use the caching server before asking the outside servers for DNS information.

With DHCP

If your internet provider uses DHCP configuration, you need to create a new interface configuration file, `/etc/sysconfig/network-scripts/ifcfg-eth0`and put the following in it:

```
DEVICE=eth0

BOOTPROTO=dhcp

ONBOOT=yes
```

Now make sure that the dhcpcd client daemon is installed on your system. Go to your Linux CD and install the `dhcpcd` RPM package.

It's time to test your new network configuration. Just use the command `/etc/rc.d/init.d/network restart`. Now test your outside connection with ping. Ping a computer on the internet, like `www.yahoo.com` and see if anything comes back.

Quirks and Anomalies

Your situation may differ from the very simple situations described above. Here are some short remarks on the various difficulties and links to more authoritative resources and addressing them. Thanks to John Mellor for supplying the links and impetus for adding this section.

PPP Over Ethernet (PPPoE)

Several ADSL providers (Bell Atlantic, for example) are now insisting that their new customers connect to the service using the "PPP over Ethernet" protocol (PPPoE). To this end, they provide a Windows client program: not very useful for Linux users. Fortunately, PPPoE is a simple protocol and several efforts are underway to support it under Linux, although all are in the early stages.

- PPPoE on Linux for Bell Sympatico
- PPPoE on Linux for Sympatico (General Info) (Linux Info)

Stupid DHCP Tricks

One of the favorite tricks network providers play is to tie your service to a unique hostname, or even a unique network interface card. This is presumably to keep you from plugging multiple computers into your ethernet port using a hub (of course, by using Linux and Masquerading you're getting the same effect with better security and the cable company has no way of knowing!).

If the provider has given you a hostname and insisted that you set your Windows box with that name in order you use their service, then you'll have to make sure that your Linux box sends in that hostname when requesting an address from the DHCP server.

The Red Hat DHCP client is called when you set the BOOTPROTO to dhcp in the interface configuration file, but it is called without reference to a hostname. To call the program with a hostname, in Red Hat 6.1, edit the `/etc/sysconfig/network` file, and change the line:

```
HOSTNAME=
```

To read this:

```
HOSTNAME=your_isp_assigned_name
```

This may not work in some of the Red Hat variants. If it does not work, check the `/sbin/ifup` script and see if the call to dhcpcd and pump include a −h $HOSTNAME switch. If they do not, add them, so the calls look like `/sbin/dhcpcd -i $DEVICE -h $HOSTNAME` and `/sbin/pump -i $DEVICE -h $HOSTNAME`.

Road Runner

The Road Runner cable service has a special login process which much be run before the server can be used. Fortunately, a detailed Linux Road Runner HOWTO is available.

Looking at the Network Entries

Now you can admire your work. Type `ifconfig` to see all your configured devices. On my gateway computer, it looks like this:

```
eth0  Link encap:Ethernet  HWaddr 00:60:67:4A:02:0A
      inet addr:24.65.182.43  Bcast:24.65.182.255  Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
      RX packets:487167 errors:0 dropped:0 overruns:0 frame:0
      TX packets:467064 errors:0 dropped:0 overruns:0 carrier:0
      collisions:89 txqueuelen:100
      Interrupt:10 Base address:0xe400
eth1  Link encap:Ethernet  HWaddr 00:80:C8:D3:30:2C
      inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
      RX packets:284112 errors:0 dropped:0 overruns:0 frame:1
      TX packets:311533 errors:0 dropped:0 overruns:0 carrier:0
      collisions:37938 txqueuelen:100
      Interrupt:5 Base address:0xe800
lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      UP LOOPBACK RUNNING  MTU:3924  Metric:1
      RX packets:12598 errors:0 dropped:0 overruns:0 frame:0
      TX packets:12598 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
```

Note that the `eth0` interface has a fancy outside IP address, and the `eth1` address has a private internal address.

You can look at the network routes by typing the `route` command. On my gateway computer it looks like this:

```
  Kernel IP routing table
  Destination     Gateway      Genmask         Flags Metric Ref Use Iface
  255.255.255.255 *            255.255.255.255 UH    0      0     0 eth1
  192.168.1.0     *            255.255.255.0   U     0      0     0 eth1
  24.65.182.0     *            255.255.255.0   U     0      0     0 eth0
  127.0.0.0       *            255.0.0.0       U     0      0     0 lo
  default         24.65.182.1  0.0.0.0         UG    0      0     0 eth0
```

Here we can see the outside network is set up, the inside network is set up, the local device is set up, the special 255.255.255.255 broadcast address is set up, and the default route is set up to point to the internet providers gateway. Perfect!

Now you have the outside, and the inside. All the remains is to open the door between the two. First though, we have to make sure no monsters can get in from the outside.

3.4 Security

One of the drawbacks of being permanently connected to the internet via ADSL or cable is that your computer is exposed to potential security threats 24 hours a day, 7 days a week. Using Linux as a gateway reduces the risks, because it hides all your other computers: as far as the rest of the internet is concerned, only your Linux box is available for connections. This means that your network is only as secure as your Linux box, so at this point I'll give a few basic tips to make your box more secure.

First, you need to shut out all the bad guys. To do this, edit the file `/etc/hosts.deny` and make sure it looks just like this:

```
#
# hosts.deny  This file describes the names of the hosts which are
#             *not* allowed to use the local INET services, as decided
#             by the '/usr/sbin/tcpd' server.
#
#             The portmap line is redundant, but it is left to remind you that
#             the new secure portmap uses hosts.deny and hosts.allow. In particular
#             you should know that NFS uses portmap!
ALL: ALL
```

This tells the "TCP wrappers" −− which control 95% of incoming connections −− to deny all connections from all hosts. That's a pretty good rule! But, it will also keep you from connecting to your Linux box from inside your home network, which is annoying, so we will make one exception. Edit the file `/etc/hosts.allow` and make sure it looks just like this:

```
#
# hosts.allow  This file describes the names of the hosts which are
#              allowed to use the local INET services, as decided
#              by the '/usr/sbin/tcpd' server.
#
ALL: 127.0.0.1
ALL: 192.168.1.
```

This tells the "TCP wrappers" that they can allow connections to all services from the local device (127.0.0.1) and from your home network (192.168.1.).

You have now locked the monsters outside, with a strong padlock. If you want to put up bars and alarm systems, you will have to be alot more sophisticated. The Security HOWTO is a good place to start if you want to learn more about securing your Linux box.

4.Configuring Masquerading

All right! The preliminaries are over, this is where the magic begins. IP masquerading is one of the truly magical services Linux provides. There are commercial products for Windows which do the same thing, but not nearly as efficiently: an ancient 386 can merrily provide IP masquerading services to a whole medium sized office, but cannot even run Windows 95, let alone the add on masquerading package.

Linux has an extremely versatile firewalling capability, and we are going to be using it in the simplest and crudest possible manner. If you want to learn how to do firewalling like an expert, you should read both the Firewalling HOWTO for an understanding of the theory and the IPChains HOWTO for instructions on the new `ipchains` firewalling tool which ships with the Linux 2.2.X kernel (and by extension Red Hat 6.X). There is also now a very good IP Masquerading HOWTO available which has more details on masquerading tweaks.

Configuring simple masquerading is very very easy once your internal and external networking is operational. Edit the `/etc/rc.d/rc.local` file and add the following lines to the bottom:

```
# 1) Flush the rule tables.
/sbin/ipchains -F input
/sbin/ipchains -F forward
/sbin/ipchains -F output
# 2) Set the MASQ timings and allow packets in for DHCP configuration.
/sbin/ipchains -M -S 7200 10 60
/sbin/ipchains -A input -j ACCEPT -i eth0 -s 0/0 68 -d 0/0 67 -p udp
# 3) Deny all forwarding packets except those from local network.
#    Masquerage those.
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -s 192.168.1.0/24 -j MASQ
# 4) Load forwarding modules for special services.
/sbin/modprobe ip_masq_ftp
/sbin/modprobe ip_masq_raudio
```

The last two lines insert kernel modules which allow FTP and RealAudio to work for computers on the inside network. There are other modules for special services which you can tack on if you need them:

- CUSeeMe (`/sbin/modprobe ip_masq_cuseeme`)
- Internet Relay Chat (`/sbin/modprobe ip_masq_irc`)
- Quake (`/sbin/modprobe ip_masq_quake`)
- VDOLive (`/sbin/modprobe ip_masq_vdolive`)

Now you're ready to try masquerading! Run the `rc.local` script with the command `/etc/rc.d/rc.local` and you are ready to go! Sit down at one of your other computers and try some web surfing. With any luck, everything is now hunky dory.

5.Problems

There are lots and lots of things which can go wrong using a simple document like this, because there are plenty of special cases. The majority of possible problems adhere to the configuration of the internal and external network devices. I will try and respond to people with problems, figure out what went wrong and add links down here so that people with special case problems can track down help. Feel free to contact me at pramsey@refractions.net.

5.1 ICQ Does Not Work

Some portions of ICQ work fine over masquerading. Other portions do not work well at all. There is a beta quality ICQ module under development, however, which addresses some (but not all) of the deficiencies of running ICQ over masquerading. The `README` file in the source code distribution describes how to compile the module. Once you have it compiled and installed, invoke `/sbin/modprobe ip_masq_icq`.