# Linux IP Masquerade HOWTO

# Table of Contents

# Table of Contents

# Table of Contents

# Linux IP Masquerade HOWTO

## David Ranch, `dranch@trinnet.net`

v1.95, November 14, 2000

---

*This document describes how to enable the Linux IP Masquerade feature on a given Linux host. IP Masq is a form of Network Address Translation or NAT that allows internally connected computers that do not have one or more registered Internet IP addresses to have the ability to communicate to the Internet via your Linux box's single Internet IP address.*

---

## 1. Introduction



## 2. Background Knowledge



## 3. Setting Up IP Masquerade



## 4. Configuring the other internal to−be MASQed machines

## 5. Testing IP Masquerade



## 6. Other IP Masquerade Issues and Software Support



## 7. Frequently Asked Questions

## 8. Miscellaneous

# 1. Introduction

## 1.1 Introduction to IP Masquerading or IP MASQ for short

This document describes how to enable the Linux IP Masquerade feature on a given Linux host. IP Masq is a form of Network Address Translation or NAT that allows internally connected computers that do not have one or more registered Internet IP addresses to have the ability to communicate to the Internet via your Linux box's single Internet IP address. It is possible to connect your internal machines to the Linux host with LAN technologies like Ethernet, TokenRing, FDDI, as well as other kinds of connections such as dialup PPP or SLIP links. This document uses Ethernet for the primary example since it is the most common scenario.

> ***This document is intended for users using either of the stable Linux kernels: 2.0.38+ and 2.2.17+ on a IBM−compatible PC. Older kernels such as 1.2.x, 1.3.x, and 2.1.x are NOT covered in this document and, in some kernel versions, can be considered broken. Please upgrade to one of the stable Linux kernels before using IP Masquerading. The new 2.3 and 2.4 kernels with the new NetFilter code aren't covered yet but URLs are provided below. Once the feature set for Netfilter is final, the new code will be covered in this HOWTO.***
>
> ***If you are configuring IP Masq for use on a Macintosh, please email Taro Fukunaga, tarozax@earthlink.net for a copy of his short MkLinux version of this HOWTO.***

## 1.2 Foreword, Feedback & Credits

As a new user, I found it very confusing to setup IP masquerade on Linux kernel, (1.2.x kernel back then). Although there is a FAQ and a mailing list, there was no document that was dedicated to it. There were also some requests on the mailing list for such a HOWTO. So, I decided to write this HOWTO as a starting point for new users and possibly create a building block for other knowledgeable users to use add to in the future. If you have any ideas for this document, corrections, etc., feel free to tell us so that we can make it better.

This document was originally based on the original FAQ by Ken Eves and numerous helpful messages from the IP Masquerade mailing list. A special thanks to Mr. Matthew Driver whose mailing list message inspired me to set up IP Masquerade and eventually writing this. Recently, David Ranch re−wrote the HOWTO and added a substantial number of sections to the HOWTO to make this document as complete as possible.

Please feel free to send any feedback or comments to ambrose@writeme.com and dranch@trinnet.net if you have any corrections or if any information/URLs/etc. is missing. Your invaluable feedback will certainly influence the future of this HOWTO!

**This HOWTO is meant to be a fairly comprehensive guide on getting your Linux IP Masquerading network working in the shortest time possible. David is not a technical writer by trade so you might find the information in this document not as general and/or objective as it could be. The latest news and information regarding this HOWTO and other IP MASQ details can be found at the IP Masquerade Resource web page that we actively maintain. If you have any technical questions on IP Masquerade, please join the IP Masquerade Mailing List instead of sending email to David. Most MASQ problems are common for ALL MASQ users and can be easily solved by someone on the list. In addition to this, the response time of the IP MASQ email list will be much faster than a reply from**

**David.**

The latest version of this document can be found at the following sites which also contains HTML and postscript versions

- http://ipmasq.cjb.net/: The IP Masquerade Resources
- http://ipmasq2.cjb.net/: The IP Masquerade Resources MIRROR
- The Linux Documentation Project
- Dranch's Linux page
- Also refer to IP Masquerade Resource Mirror Sites Listing for other local mirror sites.

## 1.3 Copyright & Disclaimer



Ok, with all this behind us... On with the show..

---

# 2. Background Knowledge

## 2.1 What is IP Masquerade?

IP Masquerade is a networking function in Linux similar to one−to−many NAT (Network Address Translation) found in many commercial firewalls and network routers. For example, if a Linux host is connected to the Internet via PPP, Ethernet, etc., the IP Masquerade feature allows other "internal" computers connected to this Linux box (via PPP, Ethernet, etc.) to also reach the Internet as well. Linux IP Masquerading allows for this functionality even though these internal machines don't have *an officially assigned IP addresses*.

MASQ allows a set of machines to *invisibly* access the Internet via the MASQ gateway. To other machines on the Internet, all this outgoing traffic will appear to be from the IP MASQ Linux server itself. In addition to the added functionality, IP Masquerade provides the foundation to create a VERY secure networking environment. With a well built firewall, breaking the security of a well configured masquerading system and internal LAN should be considerably difficult.

If you would like to know more on how MASQ differs from 1:1 NAT and Proxy solutions, please see the what−is−masq FAQ entry.

## 2.2 Current Status

IP Masquerade has been out for several years now and is fairly mature as Linux enters the 2.2.x kernel stage. Kernels since Linux 1.3.x have had MASQ support built−in. Today many individuals and commercial businesses are using it with excellent results.

Common network uses like Web browsing, TELNET, FTP, PING, TRACEROUTE, etc. work well over IP Masquerade. Other communications such as FTP, IRC, and Real Audio work well with the appropriate IP MASQ modules loaded. Other network−specific programs like streaming audio (MP3s, True Speech, etc) work too. Some fellow users on the mailing list have even had good results with video conferencing software.

It should also be noted that running IP Masquerade with only ONE network card (NIC) to MASQ between internal and external Ethernet networks is NOT recommended. For more details, please see the aliasing FAQ section for full details.

Anyway, please refer to Supported Client Software section for a more complete listing of software supported.

IP Masquerade works well as a server to other 'client machines' running various different OS and hardware platforms. There are successful cases with internal MASQed systems using :

- Unix: Sun Solaris, *BSD, Linux, Digital UNIX, etc.
- Microsoft Windows 2000, NT (3.x and 4.x), 95/98/ME, Windows for Workgroups (with the TCP/IP package)
- IBM OS/2
- Apple Macintosh MacOS machines running either MacTCP or Open Transport
- DOS−based systems with packet drivers and the NCSA Telnet package
- VAXen
- Compaq/Digital Alpha running Linux and NT
- even Amiga computers with AmiTCP or AS225−stack.

The list goes on and on but the point is, if your OS platform talks TCP/IP, it should work with IP Masquerade!

## 2.3 Who Can Benefit From IP Masquerade?

- If you have a Linux host connected to the Internet and
- if you have some computers running TCP/IP connected to a Linux box on a local subnet, and/or
- if your Linux host has more than one modem and acts as a PPP or SLIP server connecting other computers, which
- those **OTHER** machines do not have official or public assigned IP addresses (i.e. addressed with private TCP/IP numbers).
- And of course, if you want those **OTHER** machines to communicate to the Internet without spending extra money to get additional Public / Official TCP/IP addresses from your ISP and either configure Linux to be a router or purchase an external router.

## 2.4 Who Doesn't Need IP Masquerade?

- If your machine is a stand−alone Linux host connected to the Internet (though setting up a firewall is a good idea), or

- if you already have multiple assigned public addresses for your **OTHER** machines, and
- of course, if you don't like the idea of a 'free ride' using Linux and feel more comfortable using expensive commercial tools to do the exact same thing.

## 2.5 How does IP Masquerade Work?

From the original IP Masquerade FAQ by Ken Eves:

```
  Here is a drawing of the most simple setup:

   SLIP/PPP         +------------+                        +-------------+
   to ISP provider  |  Linux     |         SLIP/PPP       | Anybox      |
  <---------- modem1|    #1      |modem2 ----------- modem3|             |
    111.222.121.212 |            |           192.168.0.100 |             |
                    +------------+                        +-------------+

    In the above drawing, a Linux box with IP_MASQUERADING is installed as
  Linux #1 and is connected to the Internet via SLIP/or/PPP using modem1.  It has
  an assigned public IP address of 111.222.121.212.  It also has modem2 connected
  to allow callers to dial-in and start a SLIP/or/PPP connection.

    The second system (which doesn't have to be running Linux) calls into the
  Linux #1 box and starts a SLIP/or/PPP connection.  It does NOT have a publicly
  assigned IP address from the Internet so it uses the private address
  192.168.0.100. (see below for more info)

    With IP Masquerade and the routing configured properly, the machine
  "Anybox" can interact with the Internet as if it was directly connected to the
  Internet (with a few small exceptions).
```

```
Quoting Pauline Middelink:

  Do not forget to mention that the "ANYBOX" machine should have the
  Linux #1 box configured as its gateway (whether is be the default route or just
  a subnet is no matter). If the "ANYBOX" machine can not do this, the Linux
  machine should be configured to support proxy arp for all routed addresses. But,
  the setup and configuration of proxy arp is beyond the scope of the document.
```

```
The following is an excerpt from a previous post on comp.os.linux.networking which
has been edited to match the names used in the above example:

   o I tell machine ANYBOX that my PPP or SLIPed Linux box is its gateway.
   o When a packet comes into the Linux box from ANYBOX, it will assign it
     a new TCP/IP source port number and slap its own IP address in the packet
     header, saving the originals.  The MASQ server will then send the modified
     packet out over the SLIP/PPP interface to the Internet.
   o When a packet returns from the Internet to the Linux box, Linux examines
     if the port number is one of those ports that was assigned above.  If so, the
     MASQ server will get the original port and IP address, put them back in the
     returned packet header, and send the packet to ANYBOX.
   o The host that sent the packet will never know the difference.
```

**Another IP Masquerading Example:**

A typical example is given in the diagram below:

```
    +----------+
    |          |  Ethernet
```

```
| A-box    |::::::
|          |.2   : 192.168.0.x
+----------+     :
                 :      +----------+   PPP
+----------+     :   .1 |  Linux   |   link
|          |     :::::::| Masq-Gate|:::::::::::::::::::// Internet
| B-box    |::::::      |          |  111.222.121.212
|          |.3   :      +----------+
+----------+     :
                 :
+----------+     :
|          |     :
| C-box    |::::::
|          |.4
+----------+

|                       |          |
| <-Internal Network--> |          | <- External Network ---->
|                       |          |
```

In this example, there are (4) computer systems that we are concerned about. There is also presumably something on the far right that your PPP connection to the Internet comes through (terminal server, etc.) and that there is some remote host (very far off to the right of the page) out on the Internet that you are interested communicating with). The Linux system **`Masq-Gate`** is the IP Masquerading gateway for ALL the internal network of machines **`A-box`**, **`B-box`** and **`C-box`** to get to the Internet. The internal network uses one of the several RFC−1918 assigned private network addresses where in this case, the Class−C network 192.168.0.0. The Linux box having the TCP/IP address 192.168.0.1 while the other systems having the addresses:

- A−Box: 192.168.0.2
- B−Box: 192.168.0.3
- C−Box: 192.168.0.4

The three machines, `A-box`, `B-box` and `C-box`, can be running any operating system as long as they can speak TCP/IP. OSes such as **Windows 95**, **Macintosh MacTCP or OpenTransport** or even another **Linux box** can connect to other machines on the Internet. When running, the masquerading system or `MASQ-gate` converts all of these internal connections so that they appear to originate from `masq-gate` itself. MASQ then arranges so that data coming back in to a masqueraded connection is relayed back to the proper originating system. Because of this, the systems on the internal network see a direct route to the internet and are unaware that their data is being masqueraded. This is called a "Transparent" connection.

NOTE: Please see the FAQ for more details on topics such as:

- The differences between NAT, MASQ, and Proxy servers.
- How packet firewalls work

## 2.6 Requirements for IP Masquerade on Linux 2.2.x

**** Please refer to IP Masquerade Resource for the latest information. ****

- Kernel 2.2.x source available from http://www.kernel.org/
  NOTE #1: Linux 2.2.x kernels less than 2.2.16 have a TCP root exploit vunerability and versions less than 2.2.11 have a IPCHAINS fragmentation bug. Because of this, people running strong IPCHAINS

rulesets are open to attack. Please upgrade your kernel to a fixed version.

NOTE #2: Most newer MASQ−supported−Distributions such as Redhat 5.2 might not be Linux 2.2.x ready for your setup. Tools like DHCP, NetUtils, etc. will need to be upgraded. More details can be found in the HOWTO.

- Loadable kernel modules, preferably 2.1.121 or newer available from http://www.pi.se/blox/modutils/index.html or ftp://ftp.ocs.com.au/pub/modutils/
- A running TCP/IP network or LAN covered in Linux NET−3−4 HOWTO and the Network Administrator's Guide
  Also check out the TrinityOS document. TrinityOS is a very comprehensive guide on Linux networking. Including topics like IP MASQ, security, DNS, DHCP, Sendmail, PPP, Diald, NFS, IPSEC−based VPNs, and performance sections to name a few. Over Fifty sections in all!
- Connectivity to Internet for your Linux host covered in Linux ISP Hookup HOWTO, Linux PPP HOWTO, TrinityOS, Linux DHCP mini−HOWTO, Linux Cable Modem mini−HOWTO and http://www.linuxdoc.org/HOWTO/mini/ADSL.html
- IP Chains 1.3.9 or newer available from http://netfilter.filewatcher.org/ipchains/.
  Additional information on version requirements, find the newest IPCHAINS HOWTO, etc is at the Linux IP Chains page
- Know how to configure, compile, and install a new Linux kernel as described in the Linux Kernel HOWTO
- You can download and use various optional IP Masquerade tools to enable other functionality such as:
    - TCP/IP port−forwarders or re−directors:
        - IP PortForwarding (IPMASQADM) − RECOMMENDED or his old mirror.

ICQ MASQ module

- Andrew Deryabin's ICQ MASQ module

PORTFW FTP Solutions:

- There are both 2.2.x and 2.0.x kernel MASQ Module solutions for PORTFWed FTP to a MASQed machine. Please see the Application Page on the IPMASQ WWW site for full details.
- There is a full FTP proxy application from SuSe that will also allow for PORTFWed−like functionality to reach an internal FTP server. For more details, please see the SuSe Proxy URL for more details.

IPROUTE2 for True 1:1 NAT, Policy−based (source) routing, and Traffic Shaping:

- ftp://ftp.inr.ac.ru/ip−routing

- Documentation can be found at http://www.compendium.com.ar/policy−routing.txt
- The Advanced Routing HOWTO
- http://defiant.coinet.com/iproute2/

Some source code mirrors are:

ftp://linux.wauug.org/pub/net ftp://ftp.nc.ras.ru/pub/mirrors/ftp.inr.ac.ru/ip−routing/ ftp://ftp.gts.cz/MIRRORS/ftp.inr.ac.ru/ ftp://ftp.funet.fi/pub/mirrors/ftp.inr.ac.ru/ip−routing/
(STM1 to
USA) ftp://sunsite.icm.edu.pl/pub/Linux/iproute/ ftp://ftp.sunet.se/pub/Linux/ip−routing/ ftp://ftp.nvg.ntnu.no/pub/linux/ip−routing/ ftp://ftp.crc.ca/pub/systems/linux/ip−routing/ ftp://ftp.paname.org
(France) ftp://donlug.ua/pub/mirrors/ip−route/ ftp://omni.rk.tusur.ru/mirrors/ftp.inr.ac.ru/ip−routing/

RPMs are available at ftp://omni.rk.tusur.ru/Tango/ and at
ftp://ftp4.dgtu.donetsk.ua/pub/RedHat/Contrib−Donbass/KAD/

Please see the IP Masquerade Resource page for more information available on these patches and possibly others as well.

## 2.7 Requirements for IP Masquerade on Linux 2.3.x and 2.4.x

**** Please refer to IP Masquerade Resource for the latest information. ****

- The newest 2.3.x and 2.4.x kernels are now using a completely new system called NetFilter (much like the 2.2.x kernels went to IPCHAINS). Fortunately **unlike** the migration to IPCHAINS, the new NetFilter tool has kernel modules that can actually NATIVELY support both IPCHAINS and IPFWADM syntax so re−writing your old script is not required. Now, there might be several benefits to do a re−write (speed, new features, etc) but that is dependant on how good your old rulesets were. Many architectural changes have gone into this new code that will give the user a lot more flexibility, future features, etc.

  Some of the new functionality includes the following pros and cons:

  **PROs:**

  - 
  - Offers TRUE 1:1 NAT functionality for those who have TCP/IP subnets to play with
  - Built−in PORT Forwarding which makes IPMASQADM no longer required
  - The new built−in PORTFWing ability works for both external and internal traffic. This means that users using PORTFW for external traffic and REDIR for internal redirection don't have to use two tools any more!
  - Full Policy−Based routing features (source−based TCP/IP address routing)
  - Compatibly with Linux's FastRoute feature for significantly faster packet forwartding (a.k.a Linux network switching)
  - Fully supports TCP/IP v4, v6, and even DECnet (ack!)
  - Supports wildcard interface names like ppp* for PPP0, PPP1, etc
  - Supports filtering on both input and output INTERFACES
  - Ethernet MAC filtering
  - Denial of Service (DoS) packet rate limiting
  - Very simple and generic Stateful−like inspection functionality
  - Packet REJECTs now have user−selectable return ICMP messages
  - Variable levels of logging (different packets can goto different SYSLOG levels

**CONs:**

- 
- Because Netfilter is an entirely new architecure, most of all the old MASQ kernel modules need to be re−written. Namely, on the FTP module has been updated though the following modules remain to be re−written:

  ip_masq_cuseeme.o ip_masq_icq.o ip_masq_quake.o ip_masq_user.o ip_masq_irc.o ip_masq_raudio.o ip_masq_vdolive.o

  There is documentation on how to do this porting at http://netfilter.kernelnotes.org/unreliable−guides/netfilter−hacking−HOWTO−5.html, If you have the time, you talent would highly appreciated to get these ported over quickly.

As of this version of the HOWTO, Netfilter is NOT covered. Once the feature set of NetFilter is set, it will be added to −this− HOWTO or possibly a new HOWTO. Until then, please see the following links for the available NetFilter documentation. As it stands, the new NetFilter code will share 95% of the same setup and troubleshooting issues that IPCHAINS users have today. Because of this fact, this HOWTO is still very relevant for NetFilter firewall and NAT users.

http://netfilter.filewatcher.org/unreliable−guides/index.html and more specifically
http://netfilter.filewatcher.org/unreliable−guides/NAT−HOWTO.html

Please see the IP Masquerade Resource page for more information available on these patches and possibly others as well.

## 2.8 Requirements for IP Masquerade on Linux 2.0.x

**** Please refer to IP Masquerade Resource for the latest information. ****

- Any decent computer hardware. See the FAQ−Hardware section for more details.

- Kernel 2.0.x source available from http://www.kernel.org/
  (Most modern Linux MASQ−supported−Distributions such as Redhat 5.2 have modular kernels with all the IP Masquerade kernel options compiled in. In such cases, there is no need to compile a new Linux kernel. If you are UPGRADING your kernel, you should be aware of what other programs might be required and/or upgraded (mentioned later in the HOWTO.)

- Loadable kernel modules, preferably 2.1.85 or newer available from http://www.pi.se/blox/modutils/index.html or ftp://ftp.ocs.com.au/pub/modutils/
  (modules−1.3.57 is the minimal requirement)

- A running TCP/IP network or LAN covered in Linux NET−3−4 HOWTO and the Network Administrator's Guide
  Also check out the TrinityOS deocument. TrinityOS is a very comprehensive guide on Linux networking including topics like IP MASQ, security, DNS, DHCP, Sendmail, PPP, Diald, NFS, IPSEC−based VPNs, and performance sections just to name a few. Over Fifty sections in all!

- Connectivity to the Internet for your Linux host covered in Linux ISP Hookup HOWTO, Linux PPP

HOWTO, TrinityOS, Linux DHCP mini−HOWTO, Linux Cable Modem mini−HOWTO and Linux ADSL mini−HOWTO

* Ipfwadm 2.3 or newer available from ftp://ftp.xos.nl/pub/linux/ipfwadm/ipfwadm−2.3.tar.gz More information on version requirement is on the Linux IPFWADM page
    * If you are interested in running IPCHAINS on a 2.0.38+ kernel, see Willy Tarreau's IPCHAINS enabler for 2.0.36 or Rusty's IPCHAINS for 2.0.x kernels
* Know how to configure, compile, and install a new Linux kernel as described in the Linux Kernel HOWTO
* You can also apply various optional IP Masquerade patches to enable other functionality such as:
    * TCP/IP port−forwarders or re−directors: With these tools, you can get some non−MASQ friendly programs to work behind a MASQ server. In addition to this, you can configure a MASQ server to let Internet users contact internal WWW, TELNET, SMTP, FTP (with a patch), etc., servers. See Forwarders section of the HOWTO for more information. Here is a list of IP Masquerading patches for 2.0.x kernels:
        * Steven Clarke's IP PortForwarding (IPPORTFW) − **RECOMMENDED**
        * IP AutoForward and a mirror (IPAUTOFW) − NOT Recommended
        * REDIR for TCP (REDIR) − NOT Recommended
        * UDP redirector (UDPRED) − NOT Recommended

      PORTFWed FTP:

        * If you are going to port forward FTP traffic to an internal FTP server, you might need to download Fred Viles's FTP server patch via HTTP or Fred Viles's FTP server patch via FTP. The reason for "might" is some users have had success while other needs this modified kernel module. Explicit details on this topic can be found in the Forwarders section of the HOWTO.

      X−Windows display forwarders:

        * X−windows forwarding (DXCP)

      ICQ MASQ module

        * Andrew Deryabin's ICQ MASQ module

      PPTP (GRE) and SWAN (IPSEC) VPNs tunneling forwarders:

        * John Hardin's VPN Masquerade forwarders or the old patch for just PPTP Support.

      Game specific patches:

        * Glenn Lamb's LooseUDP for 2.0.36+ patch.

          Please note that some WWW browsers with automatically uncompress this .gz file. To download this file, hold down the SHIFT key as you click on the above URL.

Also check out Dan Kegel's NAT Page for more information. Additional information can be found in the Game−Clients section and the FAQ section.

Please see the IP Masquerade Resource page for more information available on these patches and possibly others as well.

# 3. Setting Up IP Masquerade

**If your private network contains any vital information, think carefully in terms of SECURITY before implementing IP Masquerade. By default, IP MASQ becomes a GATEWAY for you to get to the Internet but it also can allow someone on the Internet to possibly get into your internal network.**

Once you have IP MASQ functioning, it is HIGHLY recommended for the user to implement a STRONG IPFWADM/IPCHAINS firewall ruleset. Please see the Strong−IPFWADM−Rulesets and Strong−IPCHAINS−Rulesets sections below for more details.

## 3.1 Compiling the Kernel for IP Masquerade Support

**If your Linux distribution already has all the required feature support compiled such as:**

- IPFWADM/IPCHAINS
- IP forwarding
- IP masquerading
- IP Firewalling
- etc.

and all MASQ−related modules compiled (most modular kernels will have all you need), then you will NOT need to re−compile the kernel. If you aren't sure if you Linux distribution is MASQ ready, see the MASQ−supported−Distributions section. If you don't trust this list or if your distribution isn't listed, try the following tests:

- Run the command "ls /proc/sys/net/ipv4" while logged into the Linux box.
- See if files such as "ip_forward", "ip_masq_debug", "ip_masq_udp_dloose"(optional), and "ip_always_defrag"(optional) exist.

If they do, your kernel is ready to go.

If you can't find any of the above files or if your distribution does support IP Masquerading by default, ASSUME IT DOESN'T support MASQ by default. If so.. you'll need to compile a kernel but don't worry.. it isn't hard.

Regardless of native support or not, reading this section is still highly recommended as it contains other useful information.

## Linux 2.2.x Kernels

**Please see the 2.2.x−Requirements section for any required software, patches, etc.**

- First of all, you need the kernel source for 2.2.x (preferably the latest kernel version 2.2.16 or above)

  NOTE #1: Linux 2.2.x kernels less than 2.2.16 have a TCP root exploit vunerability and versions less than 2.2.11 have a IPCHAINS fragmentation bug. Because of this, people running strong IPCHAINS rulesets are open to attack. Please upgrade your kernel to a fixed version.

  NOTE #2: As the 2.2.x train as progressed, they keep changing the compile−time options. As of this version, this section reflects the settings for 2.2.15. If you are running a previous kernel version, the dialogs will look different. It is recommended that you update to the newest kernel for all the new features and stability they bring.
- If this is your first time compiling the kernel, don't be scared. In fact, it's rather easy and it's covered in several URLs found in the 2.2.x−Requirements section.
- Unpack the kernel source to `/usr/src/` with a command: `tar xvzf linux-2.2.x.tar.gz -C /usr/src`, where the "x" in 2.2.x is the current Linux 2.2 kernel. Once finished, make sure there is a directory or symbolic link to `/usr/src/linux/`
- Apply any appropriate or optional patches to the kernel source code. As of 2.2.1, IP Masq does not require any specific patching to get everything working. Features like PPTP and Xwindows forwarders are optional. Please refer to the 2.2.x−Requirements section for URLs and the IP Masquerade Resources for up−to−date information and patch URLs.
- Here are the MINIMUM options that are needed to be compiled into the kernel. You will also need to configure the kernel to use your installed network interfaces as well. Refer to the Linux Kernel HOWTO and the README file in the kernel source directory for further instructions on compiling a kernel.

  Please note the ***YES or NO ANSWERS*** to the following. Not all options will be available without the proper kernel patches described later in this HOWTO:

```
* Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n/?]
  - YES: though not required for IP MASQ, this option allows the kernel to create the MASQ modules and enable the option for port forwarding

-- Non-MASQ options skipped --

* Enable loadable module support (CONFIG_MODULES) [Y/n/?]
  - YES: allows you to load kernel IP MASQ modules

-- Non-MASQ options skipped --

* Networking support (CONFIG_NET) [Y/n/?]
  - YES: Enables the network subsystem

-- Non-MASQ options skipped --

* Sysctl support (CONFIG_SYSCTL) [Y/n/?]
  - YES:  Enables the ability to enable disable options such as forwarding,
    dynamic IPs, LooseUDP, etc.
```

```
-- Non-MASQ options skipped --

* Packet socket (CONFIG_PACKET) [Y/m/n/?]
  - YES: Though this is OPTIONAL, this recommended feature will allow you to use TCPDUMP to debug any problems with IP MASQ

* Kernel/User netlink socket (CONFIG_NETLINK) [Y/n/?]
  - YES: Though this is OPTIONAL, this feature will allow the logging of advanced firewall issues such as routing messages, etc

* Routing messages (CONFIG_RTNETLINK) [Y/n/?]
  - NO:  This option does not have anything to do with packet firewall logging

-- Non-MASQ options skipped --

* Network firewalls (CONFIG_FIREWALL) [Y/n/?]
  - YES: Enables the kernel to be comfigured by the IPCHAINS firewall tool

* Socket Filtering (CONFIG_FILTER) [Y/n/?]
  - OPTIONAL:  Though this doesn't have anything do with IPMASQ, if you plan
    on implimenting a DHCP server on the internal network, you WILL need this
    option.

* Unix domain sockets (CONFIG_UNIX) [Y/m/n/?]
  - YES:  This enables the UNIX TCP/IP sockets mechanisms

* TCP/IP networking (CONFIG_INET) [Y/n/?]
  - YES: Enables the TCP/IP protocol

-- Non-MASQ options skipped --

* IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) [Y/n/?]
  - YES:  This will allow you to configure advanced MASQ options farther down

* IP: policy routing (CONFIG_IP_MULTIPLE_TABLES) [N/y/?]
  - NO: Not needed by MASQ though users who need advanced features such as
    TCP/IP source address-based or TOS-enabled routing will need to
    enable this option.

* IP: equal cost multipath (CONFIG_IP_ROUTE_MULTIPATH) [N/y/?]
  - NO: Not needed for normal MASQ functionality

* IP: use TOS value as routing key (CONFIG_IP_ROUTE_TOS) [N/y/?]
  - NO:  Not needed for normal MASQ functionality

* IP: verbose route monitoring (CONFIG_IP_ROUTE_VERBOSE) [Y/n/?]
  - YES: This is useful if you use the routing code to drop IP spoofed packets (highly recommended) and you want to log them.

* IP: large routing tables (CONFIG_IP_ROUTE_LARGE_TABLES) [N/y/?]
  - NO:  Not needed for normal MASQ functionality

* IP: kernel level autoconfiguration (CONFIG_IP_PNP) [N/y/?] ?
  - NO:  Not needed for normal MASQ functionality

* IP: firewalling (CONFIG_IP_FIREWALL) [Y/n/?]
  - YES: Enable the firewalling feature

* IP: firewall packet netlink device (CONFIG_IP_FIREWALL_NETLINK) [Y/n/?]
  - OPTIONAL: Though this is OPTIONAL, this feature will allow IPCHAINS to copy some packets to UserSpace tools for additional checks

* IP: transparent proxy support (CONFIG_IP_TRANSPARENT_PROXY) [N/y/?]
  - NO:  Not needed for normal MASQ functionality
```

```
* IP: masquerading (CONFIG_IP_MASQUERADE) [Y/n/?]
  - YES: Enable IP Masquerade to re-address specific internal to external TCP/IP packets

* IP: ICMP masquerading (CONFIG_IP_MASQUERADE_ICMP) [Y/n/?]
  - YES: Enable support for masquerading ICMP ping packets (ICMP error codes will be MASQed regardless).  This is an important feature for troubleshooting connections.

* IP: masquerading special modules support (CONFIG_IP_MASQUERADE_MOD) [Y/n/?]
  - YES: Though OPTIONAL, this enables the OPTION to later enable the TCP/IP Port forwarding system to allow external computers to directly connect to specified internal MASQed machines.

* IP: ipautofw masq support (EXPERIMENTAL) (CONFIG_IP_MASQUERADE_IPAUTOFW) [N/y/m/?]
  - NO:  IPautofw is a legacy method of port forwarding.  It is mainly old code and has been found to have some issues.  NOT recommended.

* IP: ipportfw masq support (EXPERIMENTAL) (CONFIG_IP_MASQUERADE_IPPORTFW) [Y/m/n/?]
  - YES: Enables IPPORTFW which allows external computers on the Internet to directly communicate to specified internal MASQed machines.  This feature is typically used to access internal SMTP, TELNET, and WWW servers.  FTP port forwarding will need an additional patch as described in the FAQ section of the MASQ HOWTO.  Additional information on port forwarding is available in the Forwards section of this HOWTO.

* IP: ip fwmark masq-forwarding support (EXPERIMENTAL) (CONFIG_IP_MASQUERADE_MFW) [Y/m/n/?]
  - OPTIONAL:  This is a new method of doing PORTFW.  With this option, IPCHAINS can mark packets that should have additional work on.  Using a UserSpace tool, much like IPMASQADM or IPPORFW, IPCHAINS would then automaticaly re-address the packets. Currently, this code is less tested than PORTFW but it looks promising.  For now, the recommended method is to use IPMASQADM and IPPORTFW.  If you have thoughts on MFW, please email me.

* IP: optimize as router not host (CONFIG_IP_ROUTER) [Y/n/?]
  - YES:  This optimizes the kernel for the network subsystem though it isn't known if it makes a siginificant performance difference.

* IP: tunneling (CONFIG_NET_IPIP) [N/y/m/?]
  - NO: This OPTIONAL section is for IPIP tunnels through IP Masq.  If you
    need tunneling/VPN functionality, it is recommended to use either GRE or
    IPSEC tunnels.

* IP: GRE tunnels over IP (CONFIG_NET_IPGRE) [N/y/m/?]
  - NO:   This OPTIONAL selection is to enable PPTP and GRE tunnels through the IP MASQ box

  -- Non-MASQ options skipped --

* IP: TCP syncookie support (not enabled per default) (CONFIG_SYN_COOKIES) [Y/n/?]
  - YES: HIGHLY recommended for basic TCP/IP network security

  -- Non-MASQ options skipped --

* IP: Allow large windows (not recommended if <16Mb of memory) * (CONFIG_SKB_LARGE) [Y/n/?]
  - YES:  This is recommended to optimize Linux's TCP window

  -- Non-MASQ options skipped --

* Network device support (CONFIG_NETDEVICES) [Y/n/?]
  - YES: Enables the Linux Network device sublayer

  -- Non-MASQ options skipped --

* Dummy net driver support (CONFIG_DUMMY) [M/n/y/?]
  - YES:  Though OPTIONAL, this option can help when debugging problems

== Don't forget to compile in support for your network card !! ==

  -- Non-MASQ options skipped --

== Don't forget to compile in support for PPP/SLIP if you use a modem or
   use a PPPoE DSL modem ==

  -- Non-MASQ options skipped --

* /proc filesystem support (CONFIG_PROC_FS) [Y/n/?]
  - YES:  Required to enable the Linux network forwarding system
```

NOTE: These are just the components you need for IP Masquerade. You will need to select whatever other options needed for your specific setup.

- After compiling the kernel, you should compile and install the IP MASQ modules by doing:
```
make modules; make modules_install
```

- Then you should add a few lines into your `/etc/rc.d/rc.local` file to load the IP Masquerade modules and enable IP MASQ automatically after each reboot:

```
.
.
.
#rc.firewall script - Start IPMASQ and the firewall
/etc/rc.d/rc.firewall
.
.
.
```

## Linux 2.0.x Kernels

**Please see the 2.0.x−Requirements section for any required software, patches, etc.**

- First of all, you need the kernel source (preferably the latest kernel version 2.0.38 or above)
- If this is your first time compiling the kernel, don't be scared. In fact, it's rather easy and it's covered in several URLs found in the 2.0.x−Requirements section.
- Unpack the kernel source to `/usr/src/` with a command: `tar xvzf linux-2.0.x.tar.gz -C /usr/src`, where the "x" in 2.0.x is the current Linux 2.0 kernel. Once finished, make sure there is a directory or symbolic link to `/usr/src/linux/`
- Apply any appropriate or optional patches to the kernel source code. As of 2.0.36, IP Masq does not require any specific patching to get everything working. Features like IPPORTFW, PPTP, and Xwindows forwarders are optional. Please refer to the 2.0.x−Requirements section for URLs and the IP Masquerade Resources for up−to−date information and additional patch URLs.
- Here are the MINIMUM options that are needed to be compiled into the kernel. You will also need to confi gure the kernel to use your installed network interfaces as well. Refer to the Linux Kernel HOWTO and the README file in the kernel source directory for further instructions on compiling a kernel

  Please note the ***YES or NO ANSWERS*** to the following options. Not all options will be available without the proper kernel patches described later in this HOWTO:

```
* Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n/?]
  - YES: this will allow you to later select the IP Masquerade feature code

* Enable loadable module support (CONFIG_MODULES) [Y/n/?]
  - YES: allows you to load kernel IP MASQ modules

* Networking support (CONFIG_NET) [Y/n/?]
  - YES: Enables the network subsystem
```

```
* Network firewalls (CONFIG_FIREWALL) [Y/n/?]
  - YES: Enables the IPFWADM firewall tool

* TCP/IP networking (CONFIG_INET)
  - YES: Enables the TCP/IP protocol

* IP: forwarding/gatewaying (CONFIG_IP_FORWARD)
  - YES: Enables Linux network packet forwarding and routing - Controlled by IPFWADM

* IP: syn cookies (CONFIG_SYN_COOKIES) [Y/n/?]
  - YES: HIGHLY recommended for basic network security

* IP: firewalling (CONFIG_IP_FIREWALL) [Y/n/?]
  - YES: Enable the firewalling feature

* IP: firewall packet logging (CONFIG_IP_FIREWALL_VERBOSE) [Y/n/?]
  - YES: (OPTIONAL but HIGHLY recommended): Allows for the reporting of firewall hits

* IP: masquerading (CONFIG_IP_MASQUERADE [Y/n/?]
  - YES: Enable IP MASQ to re-address specific internal to external TCP/IP packets

* IP: ipautofw masquerade support (EXPERIMENTAL) (CONFIG_IP_MASQUERADE_IPAUTOFW) [Y/n/?]
  - NO:  IPautofw is a legacy method of TCP/IP port forwarding.  Though it works, IPPORTFW
         is a better way so IPAUTOFW is not recommended.

* IP: ipportfw masq support (EXPERIMENTAL) (CONFIG_IP_MASQUERADE_IPPORTFW) [Y/n/?]
  - YES: This option is ONLY AVAILABLE VIA A PATCH for the 2.0.x kernels.

         With this option, external computers on the Internet can directly communicate to specified internal MASQed machines.  This feature is typically used to access internal SMTP, TELNET, and WWW servers.  FTP port forwarding sometimes might require an additional patch as described in the FAQ section.  Additional information on port forwarding is available in the Forwards section of this HOWTO.

* IP: ICMP masquerading (CONFIG_IP_MASQUERADE_ICMP) [Y/n/?]
  - YES: Enable support for masquerading ICMP packets. Though thought of as optional, many programs will NOT function properly with out ICMP support.

* IP: loose UDP port managing (EXPERIMENTAL) (CONFIG_IP_MASQ_LOOSE_UDP) [Y/n/?]
  - YES: This option is ONLY AVAILABLE VIA A PATCH for the 2.0.x kernels.

         With this option, internally masqueraded computers can play NAT-friendly games over the Internet.  Explicit details are given in the FAQ section of this HOWTO.

* IP: always defragment (CONFIG_IP_ALWAYS_DEFRAG) [Y/n/?]
  - YES:  This feature optimizes IP MASQ connections - HIGHLY recommended

* IP: optimize as router not host (CONFIG_IP_ROUTER) [Y/n/?]
  - YES:  This optimizes the kernel for the network subsystem

* IP: Drop source routed frames (CONFIG_IP_NOSR) [Y/n/?]
  - YES: HIGHLY recommended for basic network security

* Dummy net driver support (CONFIG_DUMMY) [M/n/y/?]
  - YES:  Though OPTIONAL, this option can help when debugging problems

* /proc filesystem support (CONFIG_PROC_FS) [Y/n/?]
  - YES:  Required to enable the Linux network forwarding system
```

NOTE: These are just the components you need for IP Masquerade functionality. You will need to also select whatever other options you need for your specific network and hardware setup.

- After compiling the kernel, you need to also compile and install the IP MASQ kernel modules by doing:

```
make modules; make modules_install
```

- Next, add a few lines into your `/etc/rc.d/rc.local` file to load the IP Masquerade script and thus enable IP MASQ automatically after each reboot:

```
.
.
.
#rc.firewall script - Start IPMASQ and the firewall
/etc/rc.d/rc.firewall
.
.
.
```

### Linux 2.3.x / 2.4.x Kernels

**The 2.3.x and 2.4.x kernels are NOT covered in this HOWTO yet. Please see the 2.3.x/2.4.x−Requirements section for URLs, etc until it is covered by this or a NEW howto.**

## 3.2 Assigning Private Network IP Addresses to the Internal LAN

Since all **INTERNAL MASQed** machines should NOT have official Internet assigned addressees, there must be specific and accepted way to allocate address to those machines without conflicting with anyone else's Internet addresses.

From the original IP Masquerade FAQ:

RFC 1918 is the official document on which IP addresses are to be used on a non−connected or "private" network. There are 3 blocks of numbers set aside specifically for this purpose

```
Section 3: Private Address Space

The Internet Assigned Numbers Authority (IANA) has reserved the
following three blocks of the IP address space for private networks:

             10.0.0.0        -   10.255.255.255
             172.16.0.0      -   172.31.255.255
             192.168.0.0     -   192.168.255.255

We will refer to the first block as "24-bit block", the second as
"20-bit block", and to the third as "16-bit" block".  Note that the
first block is nothing but a single class A network number, while the
second block is a set of 16 contiguous class B network numbers, and
third block is a set of 255 contiguous class C network numbers.
```

For the record, my preference is to use the 192.168.0.0 network with a 255.255.255.0 Class−C subnet mask and this HOWTO reflects this. But, any of the above private networks are valid but just be SURE to use the correct subnet−mask.

So, if you're using a Class−C network, you should number your TCP/IP enabled machines as 192.168.0.1, 192.168.0.2, 192.168.0.3, ..., 192.168.0.x

192.168.0.1 is usually the internal gateway or Linux MASQ machine to get out to the external network. Please note that 192.168.0.0 and 192.168.0.255 are the Network and Broadcast address respectively (these addresses are RESERVED). Avoid using these addresses on your machines or your network will not work properly.

## 3.3 Configuring IP Forwarding Policies

At this point, you should have your kernel and other required packages installed. All network IP addresses, gateway, and DNS addresses should be configured on your Linux MASQ server as well. If you don't know how to configure your Linux network cards, please consult the HOWTOs listed in either the 2.0.x−Requirements or 2.2.x−Requirements sections.

Now, the only thing left to do is to configure the IP firewalling tools to both FORWARD and MASQUERADE the appropriate packets to the appropriate machine:

> ** This can be accomplished in many different ways. The following suggestions and examples worked for me, but you may have different ideas or needs.
>
> ** This section ONLY provides you with the bare minimum firewall ruleset to get the IP Masquerade feature working. Once IP MASQ has been successfully tested (as described later in this HOWTO), please refer to the Strong−IPFWADM−Rulesets and Strong−IPCHAINS−Rulesets sections for more secure firewall rulesets. In addition, check out the IPFWADM (2.0.x) and/or IPCHAINS(2.2.x) man pages for more details.

### Linux 2.2.x Kernels

Please note that **IPFWADM is no longer the firewall tool** for manipulating IP Masquerading rules for both the 2.1.x and 2.2.x kernels. These new kernels now use the IPCHAINS tool. For a more detailed reason for this change, please see the FAQ section.

Create the file /etc/rc.d/rc.firewall with the following initial SIMPLE ruleset:

```
#!/bin/sh
#
# rc.firewall - Initial SIMPLE IP Masquerade test for 2.1.x and 2.2.x kernels
#               using IPCHAINS
#
# Load all required IP MASQ modules
#
#   NOTE:  Only load the IP MASQ modules you need.  All current IP MASQ modules
#          are shown below but are commented out from loading.

# Needed to initially load modules
#
/sbin/depmod -a

# Supports the proper masquerading of FTP file transfers using the PORT method
#
/sbin/modprobe ip_masq_ftp

# Supports the masquerading of RealAudio over UDP.  Without this module,
#       RealAudio WILL function but in TCP mode.  This can cause a reduction
#       in sound quality
#
#/sbin/modprobe ip_masq_raudio

# Supports the masquerading of IRC DCC file transfers
#
```

```
#/sbin/modprobe ip_masq_irc


# Supports the masquerading of Quake and QuakeWorld by default.  This modules is
#   for for multiple users behind the Linux MASQ server.  If you are going to
#   play Quake I, II, and III, use the second example.
#
#   NOTE:  If you get ERRORs loading the QUAKE module, you are running an old
#   -----  kernel that has bugs in it.  Please upgrade to the newest kernel.
#
#Quake I / QuakeWorld (ports 26000 and 27000)
#/sbin/modprobe ip_masq_quake
#
#Quake I/II/III / QuakeWorld (ports 26000, 27000, 27910, 27960)
#/sbin/modprobe ip_masq_quake 26000,27000,27910,27960


# Supports the masquerading of the CuSeeme video conferencing software
#
#/sbin/modprobe ip_masq_cuseeme

#Supports the masquerading of the VDO-live video conferencing software
#
#/sbin/modprobe ip_masq_vdolive


#CRITICAL:  Enable IP forwarding since it is disabled by default since
#
#           Redhat Users:  you may try changing the options in
#                          /etc/sysconfig/network from:
#
#                       FORWARD_IPV4=false
#                             to
#                       FORWARD_IPV4=true
#
echo "1" > /proc/sys/net/ipv4/ip_forward


#CRITICAL:  Enable automatic IP defragmenting since it is disabled by default
#           in 2.2.x kernels.  This used to be a compile-time option but the
#           behavior was changed in 2.2.12
#
echo "1" > /proc/sys/net/ipv4/ip_always_defrag


# Dynamic IP users:
#
#   If you get your IP address dynamically from SLIP, PPP, or DHCP, enable this #   following option.  This enables dynamic-ip address hacking in IP MASQ,
#   making the life with Diald and similar programs much easier.
#
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr


# Enable the LooseUDP patch which some Internet-based games require
#
#  If you are trying to get an Internet game to work through your IP MASQ box,
#  and you have set it up to the best of your ability without it working, try
#  enabling this option (delete the "#" character).  This option is disabled
#  by default due to possible internal machine UDP port scanning
#  vunerabilities.
#
#echo "1" > /proc/sys/net/ipv4/ip_masq_udp_dloose
```

```
# MASQ timeouts
#
#   2 hrs timeout for TCP session timeouts
#  10 sec timeout for traffic after the TCP/IP "FIN" packet is received
#  160 sec timeout for UDP traffic (Important for MASQ'ed ICQ users)
#
/sbin/ipchains -M -S 7200 10 160


# DHCP:  For people who receive their external IP address from either DHCP or
#        BOOTP such as ADSL or Cablemodem users, it is necessary to use the
#        following before the deny command.  The "bootp_client_net_if_name"
#        should be replaced the name of the link that the DHCP/BOOTP server
#        will put an address on to?  This will be something like "eth0",
#        "eth1", etc.
#
#        This example is currently commented out.
#
#
#/sbin/ipchains -A input -j ACCEPT -i bootp_clients_net_if_name -s 0/0 67 -d 0/0 68 -p udp

# Enable simple IP forwarding and Masquerading
#
#  NOTE:  The following is an example for an internal LAN address in the
#         192.168.0.x network with a 255.255.255.0 or a "24" bit subnet mask
#         connecting to the Internet on interface eth0.
#
#         ** Please change this network number, subnet mask, and your Internet
#         ** connection interface name to match your internal LAN setup
#
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i eth0 -s 192.168.0.0/24 -j MASQ
```

Once you are finished with editing the /etc/rc.d/rc.firewall ruleset, make it executable by typing in `chmod 700 /etc/rc.d/rc.firewall`

Now that the firewall ruleset is ready to go, you need to let it run after every reboot. You could either do this by running it by hand everytime (a pain) or add it to the boot scripts. We have covered two methods below:

- Redhat and Redhat−derived distros:
- There are two ways to load things in Redhat: /etc/rc.d/rc.local or a init script in /etc/rc.d/init.d/. The first method is the easiest. All you have to do is add the line:
    - echo "Loading the rc.firewall ruleset.." /etc/rc.d/rc.firewall

  to the end of the /etc/rc.d/rc.local file and thats it. The problem with this approach is that if you are running a STRONG firewall ruleset, the firewall isn't executed until the last stages of booting. The preferred approach is to have the firewall loaded just after the networking subsystem is loaded. For now, the HOWTO only covers how to do the /etc/rc.d/rc.local way. If you want the stronger system, I recommend you check out Section 10 of TrinityOS found in the links section at the bottom of this HOWTO.
- Slackware:

- There are two ways to load things in Slackware: /etc/rc.d/rc.local or editing the /etc/rc.d/rc.inet2 file. The first method is the easiest. All you have to do is add the line:

  ♦ echo "Loading the rc.firewall ruleset.."

  /etc/rc.d/rc.firewall

  to the end of the /etc/rc.d/rc.local file and thats it. The problem with this approach is that if you are running a STRONG firewall ruleset, the firewall isn' t executed until the last stages of booting. The preferred approach is to have the firewall loaded just after the networking subsystem is loaded. For now, the HOWTO only covers how to do the /etc/rc.d/rc.local way. If you want the strong er system, I recommend you check out Section 10 of TrinityOS found in the links section at the bottom of this HOWTO.

**Notes on how users might want to change the above firewall ruleset:**

You could have also enabled IP Masquerading on a PER MACHINE basis instead of the above method enabling an ENTIRE TCP/IP network. For example, say if I wanted only the 192.168.0.2 and 192.168.0.8 hosts to have access to the Internet and NOT any of the other internal machines. I would change the in the "Enable simple IP forwarding and Masquerading" section (shown above) of the /etc/rc.d/rc.firewall ruleset.

```
#!/bin/sh
#
# Enable simple IP forwarding and Masquerading
#
#  NOTE:  The following is an example to only allow IP Masquerading for the
#         192.168.0.2 and 192.168.0.8 machines with a 255.255.255.0 or a
#         "24" bit subnet mask connecting to the Internet on interface eth0.
#
#         ** Please change this network number, subnet mask, and your Internet
#         ** connection interface name to match your internal LAN setup
#
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i eth0 -s 192.168.0.2/32 -j MASQ
/sbin/ipchains -A forward -i eth0 -s 192.168.0.8/32 -j MASQ
```

**Common mistakes:**

What appears to be a common mistake with new IP Masq users is to make the first command:

/sbin/ipchains −P forward masquerade

Do **NOT** make your default policy be MASQUERADING. Otherwise someone who can manipulate their routing tables will be able to tunnel straight back through your gateway, using it to masquerade their OWN identity!

Again, you can add these lines to the `/etc/rc.d/rc.firewall` file, one of the other rc files you prefer, or do it manually every time you need IP Masquerade.

Please see the Strong−IPFWADM−Rulesets and Strong−IPCHAINS−Rulesets sections for a detailed guide on IPCHAINS and a strong IPCHAINS ruleset example. For additional details on IPCHAINS usage, please refer to http://netfilter.filewatcher.org/ipchains/ for the primary IPCHAINS site or the Linux IP CHAINS HOWTO Backup site

## Linux 2.0.x Kernels

Create the file /etc/rc.d/rc.firewall with the following initial SIMPLE ruleset:

```
# rc.firewall - Initial SIMPLE IP Masquerade setup for 2.0.x kernels using
#               IPFWADM
#
# Load all required IP MASQ modules
#
#   NOTE:  Only load the IP MASQ modules you need.  All current available IP
#          MASQ modules are shown below but are commented out from loading.

# Needed to initially load modules
#
/sbin/depmod -a

# Supports the proper masquerading of FTP file transfers using the PORT method
#
/sbin/modprobe ip_masq_ftp

# Supports the masquerading of RealAudio over UDP.  Without this module,
#       RealAudio WILL function but in TCP mode.  This can cause a reduction
#       in sound quality
#
#/sbin/modprobe ip_masq_raudio

# Supports the masquerading of IRC DCC file transfers
#
#/sbin/modprobe ip_masq_irc

# Supports the masquerading of Quake and QuakeWorld by default.  This modules is
#   for for multiple users behind the Linux MASQ server.  If you are going to
#   play Quake I, II, and III, use the second example.
#
#   NOTE:  If you get ERRORs loading the QUAKE module, you are running an old
#   -----  kernel that has bugs in it.  Please upgrade to the newest kernel.
#
#Quake I / QuakeWorld (ports 26000 and 27000)
#/sbin/modprobe ip_masq_quake
#
#Quake I/II/III / QuakeWorld (ports 26000, 27000, 27910, 27960)
#/sbin/modprobe ip_masq_quake 26000,27000,27910,27960

# Supports the masquerading of the CuSeeme video conferencing software
#
#/sbin/modprobe ip_masq_cuseeme

#Supports the masquerading of the VDO-live video conferencing software
#
#/sbin/modprobe ip_masq_vdolive


#CRITICAL:  Enable IP forwarding since it is disabled by default
#
#           Redhat Users:  you may try changing the options in
#                          /etc/sysconfig/network from:
#
#                       FORWARD_IPV4=false
#                             to
#                       FORWARD_IPV4=true
```

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

#CRITICAL:  Enable automatic IP defragmenting since it is disabled by default
#
#           This used to be a compile-time option but the behavior was changed
#           in 2.2.12.  This option is required for both 2.0 and 2.2 kernels.
#
echo "1" > /proc/sys/net/ipv4/ip_always_defrag

# Dynamic IP users:
#
#   If you get your Internet IP address dynamically from SLIP, PPP, or DHCP,
#   enable this following option.  This enables dynamic-ip address hacking in
#   IP MASQ, making the life with DialD, PPPd, and similar programs much easier.
#
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr


# MASQ timeouts
#
#   2 hrs timeout for TCP session timeouts
#  10 sec timeout for traffic after the TCP/IP "FIN" packet is received
#  160 sec timeout for UDP traffic (Important for MASQ'ed ICQ users)
#
/sbin/ipfwadm -M -s 7200 10 160


# DHCP:  For people who receive their external IP address from either DHCP or
#        BOOTP such as ADSL or Cablemodem users, it is necessary to use the
#        following before the deny command.  The "bootp_client_net_if_name"
#        should be replaced the name of the link that the DHCP/BOOTP server
#        will put an address on to.  This will be something like "eth0",
#        "eth1", etc.
#
#        This example is currently commented out.
#
#
#/sbin/ipfwadm -I -a accept -S 0/0 67 -D 0/0 68 -W bootp_clients_net_if_name -P udp


# Enable simple IP forwarding and Masquerading
#
#  NOTE:  The following is an example for an internal LAN address in the
#         192.168.0.x network with a 255.255.255.0 or a "24" bit subnet mask
#         connecting to the Internet on interface eth0.
#
#         ** Please change this network number, subnet mask, and your Internet
#         ** connection interface name to match your internal LAN setup
#
/sbin/ipfwadm -F -p deny
/sbin/ipfwadm -F -a m -W eth0 -S 192.168.0.0/24 -D 0.0.0.0/0
```

Once you are finished with editing the /etc/rc.d/rc.firewall ruleset, make it executable by typing in "`chmod 700 /etc/rc.d/rc.firewall`"

Now that the firewall ruleset is ready to go, you need to let it run after every reboot. You could either do this by running it by hand everytime (a pain) or add it to the boot scripts. We have covered two methods below:

- Redhat and Redhat−derived distros:

- ♦ There are two ways to load things in Redhat: /etc/rc.d/rc.local or a init script in /etc/rc.d/init.d/. The first method is the easiest. All you have to do is add the line:

  ◊ echo "Loading the rc.firewall ruleset.."

  /etc/rc.d/rc.firewall

to the end of the /etc/rc.d/rc.local file and thats it. The problem with this approach is that if you are running a STRONG firewall ruleset, the firewall isn't executed until the last stages of booting. The preferred approach is to have the firewall loaded just after the networking subsystem is loaded. For now, the HOWTO only covers how to do the /etc/rc.d/rc.local way. If you want the stronger system, I recommend you check out Section 10 of TrinityOS found in the links section at the bottom of this HOWTO.

- Slackware:

  - ♦ There are two ways to load things in Slackware: /etc/rc.d/rc.local or editing the /etc/rc.d/rc.inet2 file. The first method is the easiest. All you have to do is add the line:

    ◊ echo "Loading the rc.firewall ruleset.."

    /etc/rc.d/rc.firewall

    to the end of the /etc/rc.d/rc.local file and thats it. The problem with this approach is that if you are running a STRONG firewall ruleset, the firewall isn' t executed until the last stages of booting. The preferred approach is to have the firewall loaded just after the networking subsystem is loaded. For now, the HOWTO only covers how to do the /etc/rc.d/rc.local way. If you want the strong er system, I recommend you check out Section 10 of TrinityOS found in the links section at the bottom of this HOWTO.

**Notes on how users might want to change the above firewall ruleset:**

You could have also enabled IP Masquerading on a PER MACHINE basis instead of the above method enabling an ENTIRE TCP/IP network. For example, say if I wanted only the 192.168.0.2 and 192.168.0.8 hosts to have access to the Internet and NOT any of the other internal machines. I would change the in the "Enable simple IP forwarding and Masquerading" section (shown above) of the /etc/rc.d/rc.firewall ruleset.

```
# Enable simple IP forwarding and Masquerading
#
#  NOTE:  The following is an example to only allow IP Masquerading for the
#         192.168.0.2 and 192.168.0.8 machines with a 255.255.255.0 or a "24"
#         bit subnet mask connecting to the Internet on interface eth0.
#
#         ** Please change this network number, subnet mask, and your Internet
#         ** connection interface name to match your internal LAN setup
#
#         Please use the following in ADDITION to the simple ruleset above for
#         specific MASQ networks.
#
/sbin/ipfwadm -F -p deny
/sbin/ipfwadm -F -a m -W eth0 -S 192.168.0.2/32 -D 0.0.0.0/0
/sbin/ipfwadm -F -a m -W eth0 -S 192.168.0.8/32 -D 0.0.0.0/0
```

**Common mistakes:**

What appears to be a common mistake with new IP Masq users is to make the first command:

```
ipfwadm -F -p masquerade
```

Do **NOT** make your default policy be MASQUERADING. Otherwise someone who can manipulate their routing tables will be able to tunnel straight back through your gateway, using it to masquerade their OWN identity!

Again, you can add these lines to the `/etc/rc.d/rc.firewall` file, one of the other rc files you prefer, or do it manually every time you need IP Masquerade.

Please see the Strong−IPCHAINS−Rulesets and Strong−IPFWADM−Rulesets sections for a detailed guide and stronger examples of IPCHAINS and IPFWADM ruleset examples.

---

## 4. Configuring the other internal to−be MASQed machines

Besides setting the appropriate IP address for each internal MASQed machine, you should also set each internal machine with the appropriate gateway IP address of the Linux MASQ server and required DNS servers. In general, this is rather straight forward. You simply enter the address of your Linux host (usually 192.168.0.1) as the machine's gateway address.

For the Domain Name Service, you can add in any DNS servers that are available. The most apparent one should be the one that your Linux server is using. You can optionally add any "domain search" suffix as well.

After you have properly reconfigured the internal MASQed machines, remember to restart their appropriate network services or reboot them.

The following configuration instructions assume that you are using a Class C network with 192.168.0.1 as your Linux MASQ server's address. Please note that 192.168.0.0 and 192.168.0.255 are reserved TCP/IP address.

As it stands, the following Platforms have been tested as internal MASQed machines. This is only an EXAMPLE of all of the compatible OSes out there:

- Apple Macintosh OS (with MacTCP or Open Transport)
- Commodore Amiga (with AmiTCP or AS225−stack)
- Digital VAX Stations 3520 and 3100 with UCX (TCP/IP stack for VMS)
- Digital Alpha/AXP with Linux/Redhat
- IBM AIX running on a RS/6000
- IBM OS/2 (including Warp v3)
- IBM OS400 running on a AS/400
- Linux 1.2.x, 1.3.x, 2.0.x, 2.1.x, 2.2.x
- Microsoft DOS (with NCSA Telnet package, DOS Trumpet works partially)
- Microsoft Windows 3.1 (with the Netmanage Chameleon package)
- Microsoft Windows For Workgroup 3.11 (with TCP/IP package)
- Microsoft Windows 95, OSR2, 98, 98se
- Microsoft Windows NT 3.51, 4.0, 2000 (both workstation and server)
- Novell Netware 4.01 Server with the TCP/IP service

- SCO Openserver (v3.2.4.2 and 5)
- Sun Solaris 2.51, 2.6, 7

## 4.1 Configuring Microsoft Windows 95

1. If you haven't installed your network card and adapter driver, do so now. Description of this is beyond the scope of this document.

2. Go to the *'Control Panel'* —> *'Network'*.

3. Click on *Add* —> *Protocol* —> *Manufacture: Microsoft* —> *Protocol: 'TCP/IP protocol'* if you don't already have it.

4. Highlight the TCP/IP item bound to your Windows95 network card and select *'Properties'*. Now goto the *'IP Address'* tab and set IP Address to 192.168.0.x, (1 < x < 255), and then set the Subnet Mask to 255.255.255.0

5. Now select the *"Gateway"* tab and add 192.168.0.1 as your gateway under *'Gateway'* and hit "Add".

6. Under the *'DNS Configuration'* tab, make sure to put in a name for this machine and enter in your official domain name. If you don't have your own domain, put in the domain of your ISP. Now, add all of the DNS server that your Linux host uses (usually found in `/etc/resolv.conf`). Usually these DNS servers are located at your ISP though you can be running either your own CACHING or Authoritative DNS server on your Linux MASQ server as well. Optionally, you can add any appropriate domain search suffixes as well.

7. Leave all the other settings as they are unless you know what you're doing.

8. Click *'OK'* on all dialog boxes and restart system.

9. `Ping` the linux box to test the network connection: *'Start/Run'*, type: `ping 192.168.0.1` (This is only an INTERNAL LAN connection test, you can't `ping` the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.

10. You can optionally create a `HOSTS` file in the C:\Windows directory so that you can ping the "hostname" of the machines on your LAN without the need for a DNS server. There is an example called `HOSTS.SAM` in the C:\windows directory.

## 4.2 Configuring Windows NT

1. If you haven't installed your network card and adapter driver, do so now. Description of this is beyond the scope of this document.

2. Go to *'Control Panel'* —> *'Network'* —> *Protocols*

3. Add the TCP/IP Protocol and related Components from the *'Add Software'* menu if you don't have TCP/IP service installed already.

4. Under *'Network Software and Adapter Cards'* section, highlight the *'TCP/IP Protocol'* in the *'Installed Network Software'* selection box.

5. In *'TCP/IP Configuration'*, select the appropriate adapter, e.g. `[1]Novell NE2000 Adapter`. Then set the IP Address to 192.168.0.x (1 < x < 255), then set Subnet Mask to 255.255.255.0 and Default Gateway to 192.168.0.1

6. Do not enable any of the following options (unless you know what you are e xactly doing):

   - *'Automatic DHCP Configuration'* : Unless you have a DHCP server running on your network.
   - Put anything in the *'WINS Server'* input areas : Unless you hav e setup one or more WINS servers.
   - *Enable IP Forwardings* : Unless you are routing on your NT machine and really −REALLY− know EXACTLY what you're doing.

7. Click *'DNS'*, fill in the appropriate information that your Linux host uses (usually found in /etc/resolv.conf) and then click *'OK'* when you're done.

8. Click *'Advanced'*, be sure to DISABLE *'DNS for Windows Name Resolution'* and *'Enable LMHOSTS lookup'* unless you known what these options do. If you want to use a LMHOSTS file, it is stored in C:\winnt\system32\drivers\etc.

9. Click *'OK'* on all dialog boxes and restart system.

10. `Ping` the linux box to test the network connection: *'File/Run'*, type: `ping 192.168.0.1` (This is only an INTERNAL LAN connection test, you can't `ping` the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.

## 4.3 Configuring Windows for Workgroup 3.11

1. If you haven't installed your network card and adapter driver, do so now. Description of this is beyond the scope of this document.

2. Install the TCP/IP 32b package if you don't have it already.

3. In *'Main'/'Windows Setup'/'Network Setup'*, click on *'Drivers'*.

4. Highlight *'Microsoft TCP/IP−32 3.11b'* in the *'Network Drivers'* section, click *'Setup'*.

5. Set the IP Address to 192.168.0.x (1 < x < 255), then set the Subnet Mask to 255.255.255.0 and Default Gateway to 192.168.0.1

6. Do not enable any of the following options (unless you know what you are exactly doing):

   - *'Automatic DHCP Configuration'* : Unless you have a DHCP server running on your network.
   - Put anything in the *'WINS Server'* input areas : Unless you have setup one or more WINS servers.

7. Click *'DNS'*, fill in the appropriate information your Linux host uses (usually found in /etc/resolv.conf). Then click *'OK'* when you're done with it.

8. Click *'Advanced'*, check *'Enable DNS for Windows Name Resolution'* and *'Enable LMHOSTS lookup'* found in c:\windows.

9. Click *'OK'* on all dialog boxes and restart system.

10. `Ping` the linux box to test the network connection: *'File/Run'*, type: `ping 192.168.0.1` (This is only an INTERNAL LAN connection test, you can't `ping` the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.

## 4.4 Configuring UNIX Based Systems

1. If you haven't installed your network card and recompile your kernel with the appropriate adapter driver, do so now. Description of this is beyond the scope of this document.
2. Install TCP/IP networking, such as the net−tools package, if you don't have it already.

3. Set *IPADDR* to 192.168.0.x (1 < x < 255), then set *NETMASK* to 255.255.255.0, *GATEWAY* to 192.168.0.1, and *BROADCAST* to 192.168.0.255

   For example with Redhat Linux systems, you can edit the `/etc/sysconfig/network−scripts/ifcfg−eth0` file, or simply do it through the Control Panel. These changes are different for other UNIXes such as SunOS, BSDi, Slackware Linux, Solaris, SuSe, Debian, etc...). Please refer to your UNIX documentation for more information.

4. Add your domain name service (DNS) and domain search suffix in `/etc/resolv.conf` and for the appropreiate UNIX versions, edit the /etc/nsswitch.conf file to enable DNS services.

5. You may want to update your `/etc/networks` file depending on your settings.

6. Restart the appropriate services, or simply restart your system.

7. Issue a `ping` command: `ping 192.168.0.1` to test the connection to your gateway machine. (This is only an INTERNAL LAN connection test, you can't `ping` the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.

## 4.5 Configuring DOS using NCSA Telnet package

1. If you haven't installed your network card, do so now. Description of this is beyond the scope of this document.

2. Load the appropriate packet driver. For example: using a NE2000 Ethernet card set for I/O port 300 and IRQ 10, issue `nwpd 0x60 10 0x300`

3. Make a new directory, and then unpack the NCSA Telnet package: `pkunzip tel2308b.zip`

4. Use a text editor to open the `config.tel` file

5. Set `myip=192.168.0.x` (1 < x < 255), and netmask=255.255.255.0

6. In this example, you should set `hardware=packet, interrupt=10, ioaddr=60`

7. You should have at least one individual machine specification set as the gateway, i.e. the Linux host:

```
name=default
```

```
host=yourlinuxhostname
hostip=192.168.0.1
gateway=1
```

8. Have another specification for a domain name service:

   ```
   name=dns.domain.com ; hostip=123.123.123.123; nameserver=1
   ```

   Note: substitute the appropriate information about the DNS that your Linux host uses
9. Save your `config.tel` file
10. Telnet to the linux box to test the network connection: `telnet 192.168.0.1` If you don't receive a LOGIN prompt, please verify your network configuration.

## 4.6 Configuring MacOS Based System Running MacTCP

1. If you haven't installed the appropriate driver software for your Ethernet adapter, do so now. Description of this is beyond the scope of this document.
2. Open the *MacTCP control panel*. Select the appropriate network driver (Ethernet, NOT EtherTalk) and click on the *'More...'* button.
3. Under *'Obtain Address:'*, click *'Manually'*.
4. Under *'IP Address:'*, select *class C* from the popup menu. Ignore the rest of this section of the dialog box.
5. Fill in the appropriate information under *'Domain Name Server Information:'*.
6. Under *'Gateway Address:'*, enter 192.168.0.1
7. Click *'OK'* to save the settings. In the main window of the *MacTCP control panel*, enter the IP address of your Mac (192.168.0.x, 1 < x < 255) in the *'IP Address:'* box.
8. Close the *MacTCP control panel*. If a dialog box pops up notifying you to do so, restart the system.
9. You may optionally ping the Linux box to test the network connection. If you have the freeware program *MacTCP Watcher*, click on the *'Ping'* button, and enter the address of your Linux box (192.168.0.1) in the dialog box that pops up. (This is only an INTERNAL LAN connection test, you can't ping the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.
10. You can optionally create a `Hosts` file in your System Folder so that you can use the hostnames of the machines on your LAN. The file should already exist in your System Folder, and should contain some (commented−out) sample entries which you can modify according to your needs.

## 4.7 Configuring MacOS Based System Running Open Transport

1. If you haven't installed the appropriate driver software for your Ethernet adapter, do so now. Description of this is beyond the scope of this document.

2. Open the *TCP/IP Control Panel* and choose *'User Mode ...'* from the *Edit* menu. Make sure the user mode is set to at least *'Advanced'* and click the *'OK'* button.

3. Choose *'Configurations...'* from the *File* menu. Select your *'Default'* configuration and click the *'Duplicate...'* button. Enter 'IP Masq' (or something to let you know that this is a special configuration) in the *'Duplicate Configuration'* dialog, it will probably say something like *'Default copy'*. Then click the *'OK'* button, and the *'Make Active'* button

4. Select *'Ethernet'* from the *'Connect via:'* pop−up.

5. Select the appropriate item from the *'Configure:'* pop−up. If you don't know which option to choose, you probably should re−select your *'Default'* configuration and quit. I use *'Manually'*.

6. Enter the IP address of your Mac (192.168.0.x, 1 < x < 255) in the *'IP Address:'* box.

7. Enter 255.255.255.0 in the *'Subnet mask:'* box.

8. Enter 192.168.0.1 in the *'Router address:'* box.

9. Enter the IP addresses of your domain name servers in the *'Name server addr.:'* box.

10. Enter the name of your Internet domain (e.g. 'microsoft.com') in the *'Starting domain name'* box under *'Implicit Search Path:'*.

11. The following procedures are optional. Incorrect values may cause erratic behavior. If you're not sure, it's probably better to leave them blank, unchecked and/or un−selected. Remove any information from those fields, if necessary. As far as I know there is no way through the TCP/IP dialogs, to tell the system not to use a previously select alternate "Hosts" file. If you know, I would be interested.

    Check the *'802.3'* if your network requires 802.3 frame types.

12. Click the *'Options...'* button to make sure that the TCP/IP is active. I use the *'Load only when needed'* option. If you run and quit TCP/IP applications many times without rebooting your machine, you may find that unchecking the *'Load only when needed'* option will prevent/reduce the effects on your machines memory management. With the item unchecked the TCP/IP protocol stacks are always loaded and available for use. If checked, the TCP/IP stacks are automatically loaded when needed and un−loaded when not. It's the loading and unloading process that can cause your machines memory to become fragmented.

13. You may ping the Linux box to test the network connection. If you have the freeware program *MacTCP Watcher*, click on the *'Ping'* button, and enter the address of your Linux box (192.168.0.1) in the dialog box that pops up. (This is only an INTERNAL LAN connection test, you can't ping the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.

14. You can optionally create a `Hosts` file in your System Folder so that you can use the hostnames of the machines on your LAN. The file may or may not already exist in your System Folder. If so, it

should contain some (commented−out) sample entries which you can modify according to your needs. If not, you can get a copy of the file from a system running MacTCP, or just create your own (it follows a subset of the Unix `/etc/hosts` file format, described on RFC952). Once you've created the file, open the *TCP/IP control panel*, click on the *'Select Hosts File...'* button, and open the `Hosts` file.

15. Click the close box or choose *'Close'* or *'Quit'* from the *File* menu, and then click the *'Save'* button to save the changes you have made.

16. The changes take effect immediately, but rebooting the system won't hurt.

## 4.8 Configuring Novell network using DNS

1. If you haven't installed the appropriate driver software for your Ethernet adapter, do so now. Description of this is beyond the scope of this document.

2. Downloaded tcpip16.exe from The Novell LanWorkPlace page

3. `edit c:\nwclient\startnet.bat`

   : (here is a copy of mine)

```
SET NWLANGUAGE=ENGLISH
LH LSL.COM
LH KTC2000.COM
LH IPXODI.COM
LH tcpip
LH VLM.EXE
F:
```

4. `edit c:\nwclient\net.cfg`

   : (change link driver to yours i.e. NE2000)

```
Link Driver KTC2000
        Protocol IPX 0 ETHERNET_802.3
        Frame ETHERNET_802.3
        Frame Ethernet_II
        FRAME Ethernet_802.2

NetWare DOS Requester
           FIRST NETWORK DRIVE = F
           USE DEFAULTS = OFF
           VLM = CONN.VLM
           VLM = IPXNCP.VLM
           VLM = TRAN.VLM
           VLM = SECURITY.VLM
           VLM = NDS.VLM
           VLM = BIND.VLM
           VLM = NWP.VLM
           VLM = FIO.VLM
           VLM = GENERAL.VLM
           VLM = REDIR.VLM
           VLM = PRINT.VLM
           VLM = NETX.VLM
```

```
Link Support
        Buffers 8 1500
        MemPool 4096

Protocol TCPIP
        PATH SCRIPT     C:\NET\SCRIPT
        PATH PROFILE    C:\NET\PROFILE
        PATH LWP_CFG    C:\NET\HSTACC
        PATH TCP_CFG    C:\NET\TCP
        ip_address      192.168.0.xxx
        ip_router       192.168.0.1

Change the IP address in the above "ip_address" field (192.168.0.x, 1 < x < 255)
and finally create c:\bin\resolv.cfg:

SEARCH DNS HOSTS SEQUENTIAL
NAMESERVER xxx.xxx.xxx.xxx
NAMESERVER yyy.yyy.yyy.yyy
```

5. Now edit the above "NAMESERVER" entries and replace them with the correct IP addresses for your local DNS server.

6. Issue a `ping` command: `ping 192.168.0.1` to test the connection to your gateway machine. (This is only an INTERNAL LAN connection test, you can't `ping` the outside world yet.) If you don't see "replies" to your PINGs, please verify your network configuration.

## 4.9 Configuring OS/2 Warp

1. If you haven't installed the appropriate driver software for your Ethernet adapter, do so now. Description of this is beyond the scope of this document.

2. Install the TCP/IP protocol if you don't have it already.

3. Go to *Programs/TCP/IP (LAN) / TCP/IP* Settings

4. In *'Network'* add your TCP/IP Address (192.168.0.x) and set your netmask (255.255.255.0)

5. Under *'Routing'* press *'Add'*. Set the *Type* to *'default'* and type the IP Address of your Linux Box in the Field *'Router Address'*. (192.168.0.1).

6. Set the same DNS (Nameserver) Address that your Linux host uses in *'Hosts'*.

7. Close the TCP/IP control panel. Say yes to the following question(s).

8. Reboot your system

9. You may ping the Linux box to test the network configuration. Type `'ping 192.168.0.1'` in a 'OS/2 Command prompt Window'. When ping packets are received all is ok.

## 4.10 Configuring OS/400 on a IBM AS/400

The description of how to configure TCP/IP on OS/400 version V4R1M0 running on a AS/400 is beyond the scope of this document.

1) To perform any communications configuration tasks on your AS/400, you must have the special authority of *IOSYSCFG (I/O System Configuration) defined in your user profile. You can check the characteristics of your user profile with the DSPUSRPRF command.

2) Type GO CFGTCP command th reach the Configure TCP/IP menu.

3) Select Option 2 − Work with TCP/IP Routes.

4) Enter a 1 on the Opt field to add a route. * In Route Destination type *DFTROUTE * In Subnet Mask type *NONE * In Type of Service type *NORMAL * In Nex Hop type the address of your gataway (the Linux box)

## 4.11 Configuring Other Systems

The same logic should apply to setting up other platforms. Consult the sections above. If you're interested in writing about any of systems that have not been covered yet, please send a detail setup instruction to ambrose@writeme.com and dranch@trinnet.net.

---

# 5. Testing IP Masquerade

Finally, it's time to give IP Masquerading an official try after all this hard work. If you haven't already rebooted your Linux box, do so to make sure the machines boots ok, executes the /etc/rc.d/rc.firewall ruleset, etc. Next, make sure that both the internal LAN connection and connection of your Linux hosts to the Internet is okay.

Follow these −10− tests to make sure all aspects of your MASQ setup is running properly:

## 5.1 Testing local PC connectivity

- 
- **Step One: Testing local PC connectivity**

  From an internal MASQed computer, try pinging its local IP address (i.e. *ping 192.168.0.10* ). This will verify that TCP/IP is correctly working on the local machine. Almost ALL modern operating systems have built−in support for the "ping" command. If this ping doesn't work, make sure that TCP/IP is correctly configured on the MASQed PC as described earlier in Configuring−clients section of this HOWTO. The output should look something like the following (hit Control−C to abort the ping):

---

```
masq-client# ping 192.168.0.10
PING 192.168.0.10 (192.168.0.10): 56 data bytes
64 bytes from 192.168.0.10: icmp_seq=0 ttl=255 time=0.8 ms
64 bytes from 192.168.0.10: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 192.168.0.10: icmp_seq=2 ttl=255 time=0.4 ms
64 bytes from 192.168.0.10: icmp_seq=3 ttl=255 time=0.5 ms

--- 192.168.0.10 ping statistics ---
```

```
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.5/0.8 ms
```

## 5.2 Testing internal Linux connectivity

- 
- **Step Two: Testing internal Linux connectivity**

  On the MASQ server itself, ping then internal IP address of the MASQ server's network interface card (i.e. *ping 192.168.0.1*). The output should look something like the following (hit Control−C to abort the ping):

```
masq-client# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: icmp_seq=0 ttl=255 time=0.8 ms
64 bytes from 192.168.0.1: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=255 time=0.4 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=255 time=0.5 ms

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.5/0.8 ms
```

## 5.3 Testing External Linux connectivity

- 
- **Step Three: Testing External Linux connectivity**

  Next ping the external IP address of the MASQ server's network interface card connected to the Internet. This address might be from a PPP, Ethernet, etc connection to your ISP. If you don't know what this IP address is, run the Linux command *"/sbin/ifconfig"* on the MASQ server itself to get the Internet address. The output should look something like the following (we are looking for the IP address of eth0):

```
eth0      Link encap:Ethernet  HWaddr 00:08:C7:A4:CC:5B
          inet addr:12.13.14.15  Bcast:64.220.150.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6108459 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5422798 errors:8 dropped:0 overruns:0 carrier:8
          collisions:4675 txqueuelen:100
          Interrupt:11 Base address:0xfcf0
```

  As you can see from the above, the external IP address is "12.13.14.15" for this example. So, now that you have your IP address after running the "ipconfig" command, ping your external IP address. This will confirm that the MASQ server has full network connectivity. The output should look something like the following (hit Control−C to abort the ping):

```
masq-server# ping 12.13.14.15
PING 12.13.14.15 (12.13.14.15): 56 data bytes
64 bytes from 12.13.14.15: icmp_seq=0 ttl=255 time=0.8 ms
64 bytes from 12.13.14.15: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 12.13.14.15: icmp_seq=2 ttl=255 time=0.4 ms
64 bytes from 12.13.14.15: icmp_seq=3 ttl=255 time=0.5 ms

--- 12.13.14.15 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.5/0.8 ms
```

If either of these tests doesn't work, you need to go back and double check your network cabling, and verify that the two NICs in the MASQ server are seen in "dmesg". An example of this output would be the following towards the END of the "dmesg" command:

```
.
.
PPP: version 2.3.7 (demand dialling)
TCP compression code copyright 1989 Regents of the University of California
PPP line discipline registered.
3c59x.c:v0.99H 11/17/98 Donald Becker
http://cesdis.gsfc.nasa.gov/linux/drivers/
vortex.html
eth0: 3Com 3c905 Boomerang 100baseTx at 0xfe80,  00:60:08:a7:4e:0e, IRQ 9
  8K word-wide RAM 3:5 Rx:Tx split, autoselect/MII interface.
  MII transceiver found at address 24, status 786f.
  Enabling bus-master transmits and whole-frame receives.
eth1: 3Com 3c905 Boomerang 100baseTx at 0xfd80,  00:60:97:92:69:f8, IRQ 9
  8K word-wide RAM 3:5 Rx:Tx split, autoselect/MII interface.
  MII transceiver found at address 24, status 7849.
  Enabling bus-master transmits and whole-frame receives.
Partition check:
 sda: sda1 sda2 < sda5 sda6 sda7 sda8 >
 sdb:
.
.
```

Also, don't forget to verify the NIC configurations under the Linux distro is correct, etc. per the recommendations in the beginning of this HOWTO.

## 5.4 Testing local PC to Linux connectivity

- 
- **Step Four: Testing local PC to Linux connectivity**

On a internal MASQed computer, try pinging the IP address of the Masquerading Linux box's internal Ethernet card, (i.e. *ping 192.168.0.1*). This will prove that your internal network and routing is ok. The output should look something like the following (hit Control−C to abort the ping):

```
masq-client# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: icmp_seq=0 ttl=255 time=0.8 ms
64 bytes from 192.168.0.1: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=255 time=0.4 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=255 time=0.5 ms

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.5/0.8 ms
```

If this fails, make sure Ethernet cards of the MASQ server and the MASQed computer have "link". This is usually a LED light on either the back of each Ethernet card and also on the Ethernet hub/switch (if you are using one). If this fails, make sure that the internal MASQ machine is correctly configured as shown in the Configuring−clients section. If the MASQ client is ok, double−check your network cabling, make sure you have a LINK light on both the internal MASQed computer's NIC −and− the internal NIC on the Linux box.

## 5.5 Testing internal MASQ ICMP forwarding

- 
- **Step Five: Testing internal MASQ ICMP forwarding**

From an internal MASQed computer, ping the IP address of the MASQ server's EXTERNAL TCP/IP address obtained in Step THREE above. This address might be your PPP, Ethernet, etc. address connected to your ISP. This ping test will prove that masquerading is working (ICMP Masquerading specifically).

If it doesn't work, first make sure that the "Default Gateway" on the MASQed PC is pointing to the IP address on the MASQ −SERVERs− INTERNAL NIC. Also double check that the /etc/rc.d/rc.firewall script was run without any errors. Just as a test, try re−running the /etc/rc.d/rc.firewall script now to see if it runs ok. Also, though most kernels support it by default, make sure that you enabled "ICMP Masquerading" in the kernel comfiguration and "IP Forwarding" in your /etc/rc.d/rc.firewall script.

If you still can't get things to work, take a look at the output from the following commands run on the Linux MASQ server:

- 
- "*ifconfig*" : Make sure the interface for your Internet connection (be it ppp0, eth0, etc.) is UP and you have the correct IP address for the Internet connection. An example of this output is shown in STEP THREE above.
- "*netstat −rn*" : Make sure your default gateway (the column one with the IP address in the Gateway column) is set. An example of this output might look like:

```
masq-server# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.0.1     0.0.0.0         255.255.255.255 UH        0 16384      0 eth1
12.13.14.15     0.0.0.0         255.255.255.255 UH        0 16384      0 eth0
12.13.14.0      0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

```
192.168.0.0     0.0.0.0         255.255.255.0   U         0 0          0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U         0 16384      0 lo
0.0.0.0         12.13.14.1      0.0.0.0         UG        0 16384      0 eth0
```

Notice that very LAST line that starts with 0.0.0.0? Notice that it has an IP address in the "Gateway" field? You should an IP address for your specific setup in that field.

- "*cat /proc/sys/net/ipv4/ip_forward*" : Make sure it says "1" so that Linux forwarding is enabled
- Run the command "*/sbin/ipchains −n −L*" for 2.2.x users or "*/sbin/ipfwadm −F −l*" for 2.0.x users. Specifically, look for the FORWARDing section to make sure you have MASQ enabled. An example of an IPCHAINS output might look like for users using the Simple rc.firewall ruleset:

```
.
.
Chain forward (policy REJECT):
target     prot opt     source                destination           ports
MASQ       all  ------  192.168.0.0/24       0.0.0.0/0             n/a
ACCEPT     all  ----l-  0.0.0.0/0            0.0.0.0/0             n/a
.
.
```

## 5.6 Testing external MASQ ICMP forwarding

- **Step Six: Testing external MASQ ICMP forwarding**

From an internal MASQed computer, now ping a static TCP/IP address out on the Internet (i.e. *ping 152.19.254.81* (this is http://metalab.unc.edu − home of MetaLabs' Linux Archive). If this works, that means that ICMP Masquerading is working over the Internet. If it didn't work, again check your Internet connection. If this still doesn't work, make sure you are using the simple rc.firewall ruleset and that you have ICMP Masqurading compiled into the Linux kernel. Also, make sure that the ruleset that enable IP MASQ is pointing to the correct EXNTERNAL interface.

## 5.7 Testing MASQ functionality without DNS

-
- **Step Seven: Testing MASQ functionality without DNS**

Now try TELNETing to a remote IP address (i.e. *telnet 152.2.254.81* (metalab.unc.edu − Note that this might take a while to get a login prompt since this is a VERY busy server.) Did you get a login prompt after a while? If that worked, that means that TCP Masquerading is running ok. If not, try TELNETing to some other hosts you think will support TELNET like 198.182.196.55 (www.linux.org). If this still doesn't work, make sure you are using the simple rc.firewall ruleset for now. An example of this output might look like (hit Control−D to exit out of the TELNET):

```
masq-client# telnet 152.2.254.81
Trying 152.2.254.81...
Connected to 152.2.254.81.
Escape character is '^]'.


SunOS 5.7


******************** Welcome to MetaLab.unc.edu *******************

 To login to MetaLab as a user, connect to login.metalab.unc.edu.
           This machine allows no public telnet logins.

login: Connection closed by foreign host.
```

## 5.8 Testing MASQ functionality with DNS

- **Step Eight: Testing MASQ functionality with DNS**

  Now try TELNETing to a remote HOSTNAME (i.e. *"telnet metalab.unc.edu"* (152.2.254.81). If this works, this means that DNS is working fine as well. If this didn't work but step SIX did work, make sure that you have valid DNS servers configured on your MASQed computer as shown in the Configuring−clients section.

## 5.9 Testing more MASQ functionality with DNS

- 
- **Step Nine: Testing more MASQ functionality with DNS**

  As a last test, try browsing some *'INTERNET'* WWW sites on one of your **MASQed** machines, and see if you can reach them. For example, access the Linux Documentation Project site. If this works, you can be fairly certain that everything is working FINE! If some sites are having problems where others work just fine, see the next step for more ideas.

  If you see The Linux Documentation Project homepage, then **CONGRATULATIONS! It's working!** If that WWW site comes up correctly, then all other standard network tolls such as PING, TELNET, SSH, and with their related IP MASQ modules loaded: FTP, Real Audio, IRC DCCs, Quake I/II/III, CuSeeme, VDOLive, etc. should work fine! If FTP, IRC, RealAudio, Quake I/II/III, etc. aren't working or are performing poorly, make sure their associated Masquerading modules are loaded by running "lsmod" and also be sure you are loading the module with any non−default server ports. If you don't see your needed module, make sure your /etc/rc.d/rc.firewall script is loading them (i.e. remove the # character for a give IP MASQ module).

## 5.10 Any remaining functional, performance, etc. issues...

- **Step Ten: Any remaining functional, performance, etc. issues...**

  If your system passes all of these tests above but things like WWW browsing, FTP, and other types of traffic aren't reliable, I recommend that you read the MTU−issues FAQ entry in Section 7. There might be other items in the FAQ section that will help you as they have helped many users in the past.

# 6. Other IP Masquerade Issues and Software Support

## 6.1 Problems with IP Masquerade

Some TCP/IP application protocols will not currently work with Linux IP Masquerading because they either assume things about port numbers or encode TCP/IP addresses and/or port numbers in their data stream. These latter protocols need specific proxies or IP MASQ modules built into the masquerading code to make them work.

## 6.2 Incoming services

By default, Linux IP Masquerading cannot handle incoming services at all but there are a few ways of allowing them.

If you do not require high levels of security then you can simply forward or redirect IP ports. There are various ways of doing this though the most stable method is to use IPPORTFW. For more information, please see the Forwarders section.

If you wish to have some level of authorization on incoming connections then you will need to either configure TCP−wrappers or Xinetd to then allow only specific IP addresses through. The TIS Firewall Toolkit is a good place to look for tools and information.

More details on incoming security can be found in the TrinityOS document and at IP Masquerade Resource.

## 6.3 Supported Client Software and Other Setup Notes

> **** The Linux Masquerade Application list has a lot of good information regarding applications that work through Linux IP masquerading. This site was recently taken over by Steve Grevemeyer who implimented it with a full database backend. Its a great resource!**

Generally, any application that uses standard TCP and UDP should work. If you have any suggestion, hints, etc., please see the IP Masquerade Resource for more details.

### Network Clients that −Work− with IP Masquerade

General Clients:

***Archie***

all supported platforms, file searching client (not all archie clients are supported)

***FTP***

all supported platforms, with the *ip_masq_ftp.o* kernel module for active FTP connections.

***Gopher client***

all supported platforms

***HTTP***

all supported platforms, WWW surfing

***IRC***

all IRC clients on various supported platforms, DCC is supported via the *ip_masq_irc.o* module

***NNTP (USENET)***

all supported platforms, USENET news client

***PING***

all platforms, with ICMP Masquerading kernel option

***POP3***

all supported platforms, email clients

***SSH***

all supported platforms, Secure TELNET/FTP clients

***SMTP***

all supported platforms, email servers like Sendmail, Qmail, PostFix, etc.

***TELNET***

all supported platforms, remote session

***TRACEROUTE***

UNIX and Windows based platforms , some variations may not work

***VRML***

Windows(possibly all supported platforms), virtual reality surfing

***WAIS client***

all supported platforms

Multimedia and Communication Clients:

***All H.323 programs***

– MS Netmeeting, Intel Internet Phone Beta , and other H.323 applications – There are now two solutions to get this to work through IPMASQed connections:

There is a stable BETA module available on the MASQ WWW site or at http://www.coritel.it/projects/sofia/nat.html to work with Microsoft Netmeeting v3.x code on 2.2.x kernels. There is also another module version on the MASQ WWW site specifically for Netmeeting 2.x with 2.0.x kernels but it doesn't support Netmeeting v3.x.

Another commercial solution is the Equivalence's PhonePatch H.323 gateway.

***Alpha Worlds***

Windows, Client−Server 3D chat program

***CU−SeeMe***

all supported platforms, with the *ip_masq_cuseeme* module loaded, please see the CuSeeme section for more details.

***ICQ***

all supported clients. Requires the Linux kernel to be compiled with IPPORTFW support and ICQ is configured to be behind a NON−SOCKS proxy. A full description of this configuration is in the ICQ section.

***Internet Phone 3.2***

Windows, Peer−to−peer audio communications, people can reach you only if you initiate the call, but people cannot call you without a specific port forwarding setup. See the Forwarders section for more details.

***Internet Wave Player***

Windows, network streaming audio

***Powwow***

Windows, Peer−to−peer Text audio whiteboard communications, people can reach you only if you initiate the call, but people cannot call you without a specific port forwarding setup. See the Forwarders se ction for more details.

***Real Audio Player***

Windows, network streaming audio, higher quality available with the *ip_masq_raudio* UDP module

***True Speech Player 1.1b***

Windows, network streaming audio

***VDOLive***

Windows, with the *ip_masq_vdolive* patch

***Worlds Chat 0.9a***

Windows, Client−Server 3D chat program

Games − See the LooseUDP section for more details on the LooseUDP patch

***Battle.net***

Works but requires TCP ports 116 and 118 and UDP port 6112 IPPORTFWed to the game machine. See the Forwarders section for more details. Please note that FSGS and Bnetd servers still require IPPORTFW since they haven't been re−written to be NAT−friendly.

***BattleZone 1.4***

Works with LooseUDP patch and new NAT−friendly .DLLs from Activision

***Dark Reign 1.4***

Works with LooseUDP patch or requires TCP ports 116 and 118 and UDP port 6112 IPPORTFWed to the game machine. See the Forwarders section for more details.

***Diablo***

Works with LooseUDP patch or requires TCP ports 116 and 118 and UDP port 6112 IPPORTFWed to the game machine. Newer versions of Diablo use only TCP port 6112 and UDP port 6112. See the Forwarders section for more details.

***Heavy Gear 2***

Works with LooseUDP patch or requires TCP ports 116 and 118 and UDP port 6112 IPPORTFWed to the game machine. See the Forwarders section for more details.

***Quake I/II/III***

Works right out of the box but requires the *ip_masq_quake* module if there are more than one Quake I/II/III player behind a MASQ box. Also, this module only supports Quake I and QuakeWorld by default. If you need to support Quake II or non−default server ports, please see the module install section of the rc.firewall−2.0.x and rc.firewall−2.2.x rulesets.

***StarCraft***

Works with the LooseUDP patch and IPPORTFWing TCP and UDP ports 6112 to the internal MASQed game machine. See the Forwarders section for more details.

***WorldCraft***

Works with LooseUDP patch

Other Clients:

***Linux net−acct package***

Linux, network administration−account package

***NCSA Telnet 2.3.08***

DOS, a suite containing telnet, ftp, ping, etc.

***PC−anywhere for Windows***

MS−Windows, Remotely controls a PC over TCP/IP, only work if it is a client but not a host without a specific port forwarding setup. See the Forwarders section for more details.

***Socket Watch***

uses NTP − network time protocol

### Clients that do not have full support in IP MASQ:

***Intel Streaming Media Viewer Beta 1***

Cannot connect to server

***Netscape CoolTalk***

Cannot connect to opposite side

***WebPhone***

Cannot work at present (it makes invalid assumptions about addresses).

## 6.4 Stronger IP Firewall (IPFWADM) Rulesets

This section provides a more in−depth guide on using the 2.0.x firewall tool, IPFWADM. See below for IPCHAINS rulesets

This example is for a firewall/masquerade system behind a PPP link with a static PPP address (dynamic PPP instructions are included but disabled). The trusted interface is 192.168.0.1 and the PPP interface IP address has been changed to protect the guilty :). I have listed each incoming and outgoing interface individually to catch IP spoofing as well as stuffed routing and/or masquerading. Anything not explicitly allowed is **FORBIDDEN** (well.. rejected actually). If your IP MASQ box breaks after implementing this rc.firewall script, be sure that you edited it for your configuration and check your /var/log/messages or /var/adm/messages SYSLOG file for any firewall errors.

For more comprehensive examples of a strong IP Masqueraded IPFWADM rulesets for PPP, Cablemodem users, etc., please see TrinityOS − Section 10 and GreatCircle's Firewall WWW page

**NOTE:** If you get a dynamically assigned TCP/IP address from your ISP (PPP, ADSL, Cablemodems, etc.), you **CANNOT load** this strong ruleset upon boot. You will either need to reload this firewall ruleset EVERY TIME you get a new IP address or make your /etc/rc.d/rc.firewall ruleset more intelligent. To do this for PPP users, carefully read and un−comment out the properly lines in the "Dynamic PPP IP fetch" section below. You can also find more details in the TrinityOS − Section 10 doc for more details on Strong rulesets and Dynamic IP addresses.

**Please also be aware that there are several GUI Firewall creation tools available as well. Please see the FAQ section for full details.**

Lastly, if you are using a STATIC PPP IP address, change the "ppp_ip="your.static.PPP.address"" line to reflect your address.

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

```
#!/bin/sh
#
# /etc/rc.d/rc.firewall: An example of a semi−STRONG IPFWADM firewall ruleset
#

PATH=/sbin:/bin:/usr/sbin:/usr/bin

# testing, wait a bit then clear all firewall rules.
# uncomment following lines if you want the firewall to automatically
# disable after 10 minutes.
# (sleep 600; \
# ipfwadm −I −f; \
# ipfwadm −I −p accept; \
# ipfwadm −O −f; \
# ipfwadm −O −p accept; \
# ipfwadm −F −f; \
# ipfwadm −F −p accept; \
# ) &

# Load all required IP MASQ modules
#
#   NOTE:  Only load the IP MASQ modules you need.  All current IP MASQ modules
#          are shown below but are commented from loading.

# Needed to initially load modules
#
/sbin/depmod −a

# Supports the proper masquerading of FTP file transfers using the PORT method
#
/sbin/modprobe ip_masq_ftp

# Supports the masquerading of RealAudio over UDP.  Without this module,
#       RealAudio WILL function but in TCP mode.  This can cause a reduction
#       in sound quality
#
#/sbin/modprobe ip_masq_raudio

# Supports the masquerading of IRC DCC file transfers
#
```

```
#/sbin/modprobe ip_masq_irc


# Supports the masquerading of Quake and QuakeWorld by default.  This modules is
#   for for multiple users behind the Linux MASQ server.  If you are going to
#   play Quake I, II, and III, use the second example.
#
#   NOTE:  If you get ERRORs loading the QUAKE module, you are running an old
#   -----  kernel that has bugs in it.  Please upgrade to the newest kernel.
#
#Quake I / QuakeWorld (ports 26000 and 27000)
#/sbin/modprobe ip_masq_quake
#
#Quake I/II/III / QuakeWorld (ports 26000, 27000, 27910, 27960)
#/sbin/modprobe ip_masq_quake 26000,27000,27910,27960


# Supports the masquerading of the CuSeeme video conferencing software
#
#/sbin/modprobe ip_masq_cuseeme

#Supports the masquerading of the VDO-live video conferencing software
#
#/sbin/modprobe ip_masq_vdolive


#CRITICAL:  Enable IP forwarding since it is disabled by default since
#
#           Redhat Users:  you may try changing the options in /etc/sysconfig/network from:
#
#                       FORWARD_IPV4=false
#                             to
#                       FORWARD_IPV4=true
#
echo "1" > /proc/sys/net/ipv4/ip_forward


#CRITICAL:  Enable automatic IP defragmenting since it is disabled by default
#           in 2.2.x kernels
#
#           This used to be a compile-time option but the behavior was changed
#           in 2.2.12
#
echo "1" > /proc/sys/net/ipv4/ip_always_defrag


# Dynamic IP users:
#
#   If you get your IP address dynamically from SLIP, PPP, or DHCP, enable this
#   following option.  This enables dynamic-ip address hacking in IP MASQ,
#   making the life with Diald and similar programs much easier.
#
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr


# Specify your Static IP address here.
#
#   If you have a DYNAMIC IP address, you need to make this ruleset understand
#   your IP address everytime you get a new IP.  To do this, enable the
#   following one-line script.  (Please note that the different single and
#   double quote characters MATTER).
#
```

```
#
#   DHCP users:
#   -----------
#   If you get your TCP/IP address via DHCP, **you will need ** to enable the
#   #ed out command below underneath the PPP section AND replace the word
#   "ppp0" with the name of your EXTERNAL Internet connection (eth0, eth1,
#   etc).  It should be also noted that the DHCP server can change IP
#   addresses on you.  To fix this, users should configure their DHCP client
#   to re-run the firewall ruleset everytime the DHCP lease is renewed.
#
#     NOTE #1:  Some DHCP clients like the older version of "pump" (the newer
#               versions have been fixed) did NOT have the ability to run
#               scripts after a lease-renew.  Because of this, you need to
#               replace it with something like "dhcpcd" or "dhclient".
#
#     NOTE #2:  The syntax for "dhcpcd" has changed in recent versions.
#
#               Older versions used syntax like:
#                         dhcpcd -c /etc/rc.d/rc.firewall eth0
#
#               Newer versions use syntax like:
#                         dhcpcd eth0 /etc/rc.d/rc.firewall
#
#     NOTE #3:  For Pump users, put the following line in /etc/pump.conf:
#
#                   script /etc/rc.d/rc.firewall
#
#   PPP users:
#   ----------
#   If you aren't already aware, the /etc/ppp/ip-up script is always run when
#   a PPP connection comes up.  Because of this, we can make the ruleset go
#   and get the new PPP IP address and update the strong firewall ruleset.
#
#   If the /etc/ppp/ip-up file already exists, you should edit it and add a line
#   containing "/etc/rc.d/rc.firewall" near the end of the file.
#
#   If you don't already have a /etc/ppp/ip-up sccript, you need to create the
#   following link to run the /etc/rc.d/rc.firewall script.
#
#       ln -s /etc/rc.d/rc.firewall /etc/ppp/ip-up
#
#   * You then want to enable the #ed out shell command below *
#
#
# PPP and DHCP Users:
# -------------------
# Remove the # on the line below and place a # in front of the line after that.
#
#ppp_ip="`/sbin/ifconfig ppp0 | grep 'inet addr' | awk '{print $2}' | sed -e 's/.*://'`"
#
ppp_ip="your.static.PPP.address"


# MASQ timeouts
#
#   2 hrs timeout for TCP session timeouts
#  10 sec timeout for traffic after the TCP/IP "FIN" packet is received
#  60 sec timeout for UDP traffic (MASQ'ed ICQ users must enable a 30sec firewall timeout in ICQ itself)
#
/sbin/ipfwadm -M -s 7200 10 60
```

```
#############################################################################
# Incoming, flush and set default policy of reject. Actually the default policy
# is irrelevant because there is a catch all rule with deny and log.
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p reject

# local interface, local machines, going anywhere is valid
#
/sbin/ipfwadm -I -a accept -V 192.168.0.1 -S 192.168.0.0/24 -D 0.0.0.0/0

# remote interface, claiming to be local machines, IP spoofing, get lost
#
/sbin/ipfwadm -I -a reject -V $ppp_ip -S 192.168.0.0/24 -D 0.0.0.0/0 -o

# remote interface, any source, going to permanent PPP address is valid
#
/sbin/ipfwadm -I -a accept -V $ppp_ip -S 0.0.0.0/0 -D $ppp_ip/32

# loopback interface is valid.
#
/sbin/ipfwadm -I -a accept -V 127.0.0.1 -S 0.0.0.0/0 -D 0.0.0.0/0

# catch all rule, all other incoming is denied and logged. pity there is no
# log option on the policy but this does the job instead.
#
/sbin/ipfwadm -I -a reject -S 0.0.0.0/0 -D 0.0.0.0/0 -o


#############################################################################
# Outgoing, flush and set default policy of reject. Actually the default policy
# is irrelevant because there is a catch all rule with deny and log.
#
/sbin/ipfwadm -O -f
/sbin/ipfwadm -O -p reject

# local interface, any source going to local net is valid
#
/sbin/ipfwadm -O -a accept -V 192.168.0.1 -S 0.0.0.0/0 -D 192.168.0.0/24

# outgoing to local net on remote interface, stuffed routing, deny
#
/sbin/ipfwadm -O -a reject -V $ppp_ip -S 0.0.0.0/0 -D 192.168.0.0/24 -o

# outgoing from local net on remote interface, stuffed masquerading, deny
#
/sbin/ipfwadm -O -a reject -V $ppp_ip -S 192.168.0.0/24 -D 0.0.0.0/0 -o

# outgoing from local net on remote interface, stuffed masquerading, deny
#
/sbin/ipfwadm -O -a reject -V $ppp_ip -S 0.0.0.0/0 -D 192.168.0.0/24 -o

# anything else outgoing on remote interface is valid
#
/sbin/ipfwadm -O -a accept -V $ppp_ip -S $ppp_ip/32 -D 0.0.0.0/0

# loopback interface is valid.
#
/sbin/ipfwadm -O -a accept -V 127.0.0.1 -S 0.0.0.0/0 -D 0.0.0.0/0

# catch all rule, all other outgoing is denied and logged. pity there is no
# log option on the policy but this does the job instead.
```

```
#
/sbin/ipfwadm -O -a reject -S 0.0.0.0/0 -D 0.0.0.0/0 -o


##############################################################################
# Forwarding, flush and set default policy of deny. Actually the default policy
# is irrelevant because there is a catch all rule with deny and log.
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p deny

# Masquerade from local net on local interface to anywhere.
#
/sbin/ipfwadm -F -a masquerade -W ppp0 -S 192.168.0.0/24 -D 0.0.0.0/0
#
# catch all rule, all other forwarding is denied and logged. pity there is no
# log option on the policy but this does the job instead.
#
/sbin/ipfwadm -F -a reject -S 0.0.0.0/0 -D 0.0.0.0/0 -o

#End of file.
```

With IPFWADM, you can block traffic to a particular site using the −I, −O or −F rules. Remember that the set of rules are scanned top to bottom and "−a" tells IPFWADM to "append" this new rule to the existing set of rules. So with this in mind, any specific restrictions need to come before global rules. For example:

Using −I (input ) rules:

Probably the fastest and most efficient method to block traffic but it only stops the MASQed machines and NOT the the firewall machine itself. Of course you might want to allow that combination.

Anyway, to block 204.50.10.13:

In the /etc/rc.d/rc.firewall ruleset:

```
... start of -I rules ...

# reject and log local interface, local machines going to 204.50.10.13
#
/sbin/ipfwadm -I -a reject -V 192.168.0.1 -S 192.168.0.0/24 -D 204.50.10.13/32 -o

# local interface, local machines, going anywhere is valid
#
/sbin/ipfwadm -I -a accept -V 192.168.0.1 -S 192.168.0.0/24 -D 0.0.0.0/0

... end of -I rules ...
```

Using −O (output) rules:

This is the slower method to block traffic because the packets go through masquerading first before they are dropped. Yet, this rule even stops the firewall machine from accessing the forbidden site.

```
... start of -O rules ...

# reject and log outgoing to 204.50.10.13
#
/sbin/ipfwadm -O -a reject -V $ppp_ip -S $ppp_ip/32 -D 204.50.10.13/32 -o
```

```
# anything else outgoing on remote interface is valid
#
/sbin/ipfwadm −O −a accept −V $ppp_ip −S $ppp_ip/32 −D 0.0.0.0/0

... end of −O rules ...
```

Using −F (forward) rules:

Probably slower than −I (input) rules for blocking traffic, this still only stops masqueraded machines (e.g. internal machines). The firewall machine can still reach forbidden site(s).

```
... start of −F rules ...

# Reject and log from local net on PPP interface to 204.50.10.13.
#
/sbin/ipfwadm −F −a reject −W ppp0 −S 192.168.0.0/24 −D 204.50.10.13/32 −o

# Masquerade from local net on local interface to anywhere.
#
/sbin/ipfwadm −F −a masquerade −W ppp0 −S 192.168.0.0/24 −D 0.0.0.0/0

... end of −F rules ...
```

There is no need for a special rule to allow machines on the 192.168.0.0/24 network to go to 204.50.11.0. Why? It is already covered by the global MASQ rule.

NOTE: There is more than one way of coding the interfaces in the above rules. For example instead of "−V 192.168.255.1" you can code "−W eth0", instead of "−V $ppp_ip" , you can use "−W ppp0". The "−V" method was phased out with the imgration to IPCHAINS but for IPFWADM users, its personal choice and documentation more than anything.

## 6.5 Stronger IP Firewall (IPCHAINS) rulesets

This section provides a more in−depth guide on using the 2.2.x firewall tool, IPCHAINS. See above for IPFWADM rulesets.

This example is for a firewall/masquerade system behind a PPP link with a static PPP address (dynamic PPP instructions are included but disabled). The trusted interface is 192.168.0.1 and the PPP interface IP address has been changed to protect the guilty :). I have listed each incoming and outgoing interface individually to catch IP spoofing as well as stuffed routing and/or masquerading. A nything not explicitly allowed is **FORBIDDEN** (well.. rejected actually). If your IP MASQ box breaks after implementing this rc.firewall script, be sure that you edited it for your configuration and check your /var/log/messages or /var/adm/messages SYSLOG file for any firewall errors.

For more comprehensive examples of a strong IP Masqueraded IPFWADM rulesets for PPP, Cablemodem users, etc., please see TrinityOS − Section 10 and GreatCircle's Firewall WWW page

**NOTE #1:** Linux 2.2.x kernels less than 2.2.16 have a TCP root exploit vunerability and versions less than 2.2.11 have a IPCHAINS fragmentation bug. Because of this, people running strong IPCHAINS rulesets are open to attack. Please upgrade your kernel to a fixed version.

**NOTE #2:** If you get a dynamically assigned TCP/IP address from your ISP (PPP, ADSL, Cablemodems, etc.), you **CANNOT load** this strong ruleset upon boot. You will either need to reload this firewall ruleset EVERY TIME you get a new IP address or make your /etc/rc.d/rc.firewall ruleset more intelligent. To do this for PPP users, carefully read and un−comment out the properly lines in the "Dynamic PPP IP fetch" section below. You can also find more details in the TrinityOS − Section 10 doc for more details on Strong rulesets and Dynamic IP addresses.

**Please also be aware that there are several GUI Firewall creation tools available as well. Please see the FAQ section for full details.**

Lastly, if you are using a STATIC PPP IP address, change the "ppp_ip="your.static.PPP.address"" line to reflect your address.

```
----------------------------------------------------------------
```

```
#!/bin/sh
#
# /etc/rc.d/rc.firewall: An example of a Semi-Strong IPCHAINS firewall ruleset.
#

PATH=/sbin:/bin:/usr/sbin:/usr/bin

# Load all required IP MASQ modules
#
#   NOTE:  Only load the IP MASQ modules you need.  All current IP MASQ modules
#          are shown below but are commented from loading.

# Needed to initially load modules
#
/sbin/depmod -a

# Supports the proper masquerading of FTP file transfers using the PORT method
#
/sbin/modprobe ip_masq_ftp

# Supports the masquerading of RealAudio over UDP.  Without this module,
#       RealAudio WILL function but in TCP mode.  This can cause a reduction
#       in sound quality
#
/sbin/modprobe ip_masq_raudio

# Supports the masquerading of IRC DCC file transfers
#
#/sbin/modprobe ip_masq_irc


# Supports the masquerading of Quake and QuakeWorld by default.  This modules is
#   for for multiple users behind the Linux MASQ server.  If you are going to
#   play Quake I, II, and III, use the second example.
#
#   NOTE:  If you get ERRORs loading the QUAKE module, you are running an old
#   -----  kernel that has bugs in it.  Please upgrade to the newest kernel.
#
#Quake I / QuakeWorld (ports 26000 and 27000)
#/sbin/modprobe ip_masq_quake
#
#Quake I/II/III / QuakeWorld (ports 26000, 27000, 27910, 27960)
#/sbin/modprobe ip_masq_quake 26000,27000,27910,27960
```

```
# Supports the masquerading of the CuSeeme video conferencing software
#
#/sbin/modprobe ip_masq_cuseeme

#Supports the masquerading of the VDO-live video conferencing software
#
#/sbin/modprobe ip_masq_vdolive


#CRITICAL:  Enable IP forwarding since it is disabled by default since
#
#           Redhat Users:  you may try changing the options in
#                          /etc/sysconfig/network from:
#
#                       FORWARD_IPV4=false
#                             to
#                       FORWARD_IPV4=true
#
echo "1" > /proc/sys/net/ipv4/ip_forward


#CRITICAL:  Enable automatic IP defragmenting since it is disabled by default
#           in 2.2.x kernels
#
#           This used to be a compile-time option but the behavior was changed
#           in 2.2.12.  It should also be noted that some distributions have
#           removed this option from the /proc table.  If this entry isn't
#           present in your /proc, don't worry about it.
#
echo "1" > /proc/sys/net/ipv4/ip_always_defrag


# Dynamic IP users:
#
#   If you get your IP address dynamically from SLIP, PPP, or DHCP, enable this #   following option.  This enables dynamic-ip address hacking in IP MASQ,
#   making the life with Diald and similar programs much easier.
#
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr


# Enable the LooseUDP patch which some Internet-based games require
#
#  If you are trying to get an Internet game to work through your IP MASQ box,
#  and you have set it up to the best of your ability without it working, try
#  enabling this option (delete the "#" character).  This option is disabled
#  by default due to possible internal machine UDP port scanning
#  vunerabilities.
#
#echo "1" > /proc/sys/net/ipv4/ip_masq_udp_dloose


# Specify your Static IP address here.
#
#   If you have a DYNAMIC IP address, you need to make this ruleset understand
#   your IP address everytime you get a new IP.  To do this, enable the
#   following one-line script.  (Please note that the different single and
#   double quote characters MATTER).
#
#
#   DHCP users:
```

```
#   -----------
#   If you get your TCP/IP address via DHCP, **you will need ** to enable the
#   #ed out command below underneath the PPP section AND replace the word
#   "ppp0" with the name of your EXTERNAL Internet connection (eth0, eth1, etc)
#   on the lines for "ppp-ip" and "extip".  It should be also noted that the
#   DHCP server can change IP addresses on you.  To fix this, users should
#   configure their DHCP client to re-run the firewall ruleset everytime the
#   DHCP lease is renewed.
#
#     NOTE #1:  Some DHCP clients like the original "pump" (the newer
#               versions have been fixed) did NOT have the ability to run
#               scripts after a lease-renew.  Because of this, you need to
#               replace it with something like "dhcpcd" or "dhclient".
#
#     NOTE #2:  The syntax for "dhcpcd" has changed in recent versions.
#
#               Older versions used syntax like:
#                         dhcpcd -c /etc/rc.d/rc.firewall eth0
#
#               Newer versions use syntax like:
#                         dhcpcd eth0 /etc/rc.d/rc.firewall
#
#     NOTE #3:  For Pump users, put the following line in /etc/pump.conf:
#
#                   script /etc/rc.d/rc.firewall
#
#   PPP users:
#   ----------
#   If you aren't already aware, the /etc/ppp/ip-up script is always run when
#   a PPP connection comes up.  Because of this, we can make the ruleset go and
#   get the new PPP IP address and update the strong firewall ruleset.
#
#   If the /etc/ppp/ip-up file already exists, you should edit it and add a line
#   containing "/etc/rc.d/rc.firewall" near the end of the file.
#
#   If you don't already have a /etc/ppp/ip-up sccript, you need to create the
#   following link to run the /etc/rc.d/rc.firewall script.
#
#       ln -s /etc/rc.d/rc.firewall /etc/ppp/ip-up
#
#   * You then want to enable the #ed out shell command below *
#
#
# PPP and DHCP Users:
# -------------------
# Remove the # on the line below and place a # in front of the line after that.
#
#extip="`/sbin/ifconfig ppp0 | grep 'inet addr' | awk '{print $2}' | sed -e 's/.*://'`"

# For PPP users with STATIC IP addresses:
#
extip="your.static.PPP.address"

# ALL PPP and DHCP users must set this for the correct EXTERNAL interface name
extint="ppp0"

# Assign the internal IP
intint="eth0"
intnet="192.168.0.0/24"


# MASQ timeouts
```

```
#
#   2 hrs timeout for TCP session timeouts
#  10 sec timeout for traffic after the TCP/IP "FIN" packet is received
#  60 sec timeout for UDP traffic (MASQ'ed ICQ users must enable a 30sec firewall timeout in ICQ itself)
#
ipchains -M -S 7200 10 60

#############################################################################
# Incoming, flush and set default policy of reject. Actually the default policy
# is irrelevant because there is a catch all rule with deny and log.
#
ipchains -F input
ipchains -P input REJECT

# local interface, local machines, going anywhere is valid
#
ipchains -A input -i $intint -s $intnet -d 0.0.0.0/0 -j ACCEPT

# remote interface, claiming to be local machines, IP spoofing, get lost
#
ipchains -A input -i $extint -s $intnet -d 0.0.0.0/0 -l -j REJECT

# remote interface, any source, going to permanent PPP address is valid
#
ipchains -A input -i $extint -s 0.0.0.0/0 -d $extip/32 -j ACCEPT

# loopback interface is valid.
#
ipchains -A input -i lo -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT

# catch all rule, all other incoming is denied and logged. pity there is no
# log option on the policy but this does the job instead.
#
ipchains -A input -s 0.0.0.0/0 -d 0.0.0.0/0 -l -j REJECT

#############################################################################
# Outgoing, flush and set default policy of reject. Actually the default policy
# is irrelevant because there is a catch all rule with deny and log.
#
ipchains -F output
ipchains -P output REJECT

# local interface, any source going to local net is valid
#
ipchains -A output -i $intint -s 0.0.0.0/0 -d $intnet -j ACCEPT

# outgoing to local net on remote interface, stuffed routing, deny
#
ipchains -A output -i $extint -s 0.0.0.0/0 -d $intnet -l -j REJECT

# outgoing from local net on remote interface, stuffed masquerading, deny
#
ipchains -A output -i $extint -s $intnet -d 0.0.0.0/0 -l -j REJECT

# anything else outgoing on remote interface is valid
#
ipchains -A output -i $extint -s $extip/32 -d 0.0.0.0/0 -j ACCEPT

# loopback interface is valid.
#
ipchains -A output -i lo -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
```

```
# catch all rule, all other outgoing is denied and logged. pity there is no
# log option on the policy but this does the job instead.
#
ipchains -A output -s 0.0.0.0/0 -d 0.0.0.0/0 -l -j REJECT

#######################################################################
# Forwarding, flush and set default policy of deny. Actually the default policy
# is irrelevant because there is a catch all rule with deny and log.
#
ipchains -F forward
ipchains -P forward DENY

# Masquerade from local net on local interface to anywhere.
#
ipchains -A forward -i $extint -s $intnet -d 0.0.0.0/0 -j MASQ
#
# catch all rule, all other forwarding is denied and logged. pity there is no
# log option on the policy but this does the job instead.
#
ipchains -A forward -s 0.0.0.0/0 -d 0.0.0.0/0 -l -j REJECT

#End of file.
```

With IPCHAINS, you can block traffic to a particular site using the "input", "output", and/or "forward" rules. Remember that the set of rules are scanned t op to bottom and "−A" tells IPCHIANS to "append" this new rule to the existing set of rules. So with this in mind, any specific restrictions need to come bef ore global rules. For example:

Using "input" rules:

Probably the fastest and most efficient method to block traffic but it only stops the MASQed machines and NOT the firewall machine itself. Of course you might want to allow that combination.

Anyway, to block 204.50.10.13:

In the /etc/rc.d/rc.firewall ruleset:

```
... start of "input" rules ...

# reject and log local interface, local machines going to 204.50.10.13
#
ipchains -A input -s 192.168.0.0/24 -d 204.50.10.13/32 -l -j REJECT


# local interface, local machines, going anywhere is valid
#
ipchains -A input -s 192.168.0.0/24 -d 0.0.0.0/0 -l -j ACCEPT


... end of "input" rules ...
```

Using "output" rules:

This is the slower method to block traffic because the packets must go through masquerading first before they are dropped. Yet, this rule even stops the firewall machine from accessing the forbidden site.

```
... start of "output" rules ...
```

```
# reject and log outgoing to 204.50.10.13
#
ipchains -A output -s $ppp_ip/32 -d 204.50.10.13/32 -l -j REJECT


# anything else outgoing on remote interface is valid
#
ipchains -A output -s $ppp_ip/32 -d 0.0.0.0/0 -l -j ACCEPT


... end of "output" rules ...
```

Using "forward" rules:

Probably slower than "input" rules for blocking traffic, this still only stops masqueraded machines (e.g. internal machines). The firewall machine can still reach forbidden site(s).

```
... start of "forward" rules ...

# Reject and log from local net on PPP interface to 204.50.10.13.
#
ipchains -A forward -i ppp0 -s 192.168.0.0/24 -d 204.50.10.13/32 -l -j REJECT


# Masquerade from local net on local interface to anywhere.
#
ipchains -A forward -i ppp0 -s 192.168.0.0/24 -d 0.0.0.0/0 -j MASQ

... end of "forward" rules ...
```

No need for a special rule to allow machines on the 192.168.0.0/24 network to go to 204.50.11.0. Why? It is already covered by the global MASQ rule.

NOTE: Unlike IPFWADM, IPCHIANS has only one way of coding the interfaces name. IPCHAINS uses the "−i eth0" option where as IPFWADM had both "−W" for the interface name and "−V" for the interface's IP address.

## 6.6 IP Masquerading multiple internal networks

Masquerading more than one internal network is fairly simple. You need to first make sure that all of your networks are running correctly (both internal and external). You then need to enable traffic to pass to both the other internal interfaces and to be MASQed to the Internet.

Next, you need to enable Masquerading on the INTERNAL interfaces. This example uses a total of THREE interfaces: eth0 is the EXTERNAL connection to the Internet, eth1 is the 192.168.0.0 network, and eth2 is the 192.168.1.0 network. Both eth1 and eth2 will be MASQed out of interface eth0. In your rc.firewall ruleset next to the existing MASQ enable line, add the following:

- 2.2.x kernels with IPCHAINS

  ```
  #Enable internal interfaces to communication between each other
  /sbin/ipchains -A forward -i eth1 -d 192.168.0.0/24
  /sbin/ipchains -A forward -i eth2 -d 192.168.1.0/24
  ```

```
#Enable internal interfaces to MASQ out to the Internet
/sbin/ipchains -A forward -j MASQ -i eth0 -s 192.168.0.0/24 -d 0.0.0.0/0
/sbin/ipchains -A forward -j MASQ -i eth0 -s 192.168.1.0/24 -d 0.0.0.0/0
```

• 2.0.x kernels with IPFWADM

```
#Enable internal interfaces to communication between each other
/sbin/ipfwadm -F -a accept -V 192.168.0.1 -D 192.168.1.0/24
/sbin/ipfwadm -F -a accept -V 192.168.1.1 -D 192.168.0.0/24

#Enable internal interfaces to MASQ out to the Internet
/sbin/ipfwadm -F -a masq -W eth0 -S 192.168.0.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -F -a masq -W eth0 -S 192.168.1.0/24 -D 0.0.0.0/0
```

Please note that it is CORRECT to have "eth0" specified multiple times for the exmples shown above. The reason for this is the Linux kernel needs to know which interface is used for OUTGOING traffic. Since eth0 in the above examples is the Internet connection, it is listed for each internal interface.

## 6.7 IP Masquerade and Dial−on−Demand Connections

1. If you would like to setup your network to automatically dial up the Internet, ether the *Diald* demand dial−up or new versions of the *PPPd* packages will be of great utility. Diald is the recommended solution due to its more granular configuration.
2. To setup Diald, please check out the Setting Up Diald for Linux Page or TrinityOS − Section 23
3. Once Diald and IP Masq have been setup properly, any MASQed client machines that initiate a web, telnet or ftp session will make the Linux box dynamically bring up its Internet link.
4. There is a timeout that will occur with the first connection. This is inevitable if you are using analog modems. The time taken to establish the modem link and the PPP connections may cause your client program (WWW browser, etc.). This isn't common though. If this does happen, just retry that Internet traffic request (say a WWW page) again and it should come up fine. You can also try setting *echo "1" > /proc/sys/net/ipv4/ip_dynaddr* kernel option to help with this initial setup.

## 6.8 IPPORTFW, IPMASQADM, IPAUTOFW, REDIR, UDPRED, and other Port Forwarding tools

IPPORTFW, IPAUTOFW, REDIR, UDPRED, and other programs are generic TCP and/or UDP port forwarding tools for Linux IP Masquerade. These tools are typically used with or as a replacement for specific IP MASQ modules like the current ones for FTP, Quake, etc. With port forwarders, you can now re−direct data connections from the Internet to an internal, privately addressed machine behind your IP MASQ server. This forwarding ability includes network protocols such as TELNET, WWW, SMTP, FTP (with a special patch − see below), ICQ, and many others.

NOTE: If you are just looking to do simple port forwarding without IP Masquerading support, you will **STILL NEED** to enable IP Masquerading in both the kernel AND in either your IPFWADM or IPCHAINS ruleset to then be able to use Linux's port forwarding tools.

So why all the different choices? IPAUTOFW, REDIR, and UDPRED (all URLs are in the 2.0.x−Requirements section) were the first tools available to IP MASQ users to allow this functionality. Later, as Linux IP Masquerade matured, these tools were eventually replaced by IPPORTFW which is a more intelligent solution. Because of the availablity of the newer tools, it is *HIGHLY DISCOURAGED* to use the old tools such as IPAUTOFW and REDIR because they don't properly notify the Linux kernel of their presence and can ultimately **CRASH** your Linux server with extreme use. It should also be noted that there is also the newest method, MFW. MFW's major benefit its its tigher integration with the IPCHAINS tool. With this solution, you use a IPCHAINS ruleset to "Mark" a specific packet and then create a different chain to then do the proper forwarding. Currently, this method isn't covered in the HOWTO.

**NOTE #2: With PORTFW in 2.2.x kernels, *internal machines* CANNOT use the same PORTFWed IP address to access an internal machine though it works fine with external computers on the Internet. If this is an issue for you, you can ALSO impliment the REDIR portfw tool to let internal machines get redirected to the internal server. One good think to note is the upcoming NetFilter toolset solves this issue. If you would like a technical explination of why this internal/external forwarding doesn't work, please page down towards the bottom of the 2.2.x PORTFW section for a note from Juan.**

NOTE #3: The forwarding of FTP server traffic to an internal MASQed FTP server, known as **PORTFW FTP**, is now supported for both the 2.2.x and 2.2.x kernels. This is possible either through the patching the Linux kernel sources though the support is not currently built into the main Linux kernel or using an external FTP proxy program. It should be noted that the kernel module code is still experimental and some people get better results with ACTIVE FTP sessions compared to PASSIVE connections. Interestingly enough, other people have seen the exact opposite behavior. Please let us know what your results are like. More about this is covered below in both the 2.2.x and 2.0.x sections as the solutions use different patches.

Before jumping right into installing either the 2.0.x IPPORTFW or 2.2.x version of IPMASQADM with IPPORTFW support, network security can be an issue with any port forwarder. The reason for this is because these tools basically create a hole in the packet firewall for the forwarded TCP/UDP ports. Though this doesn't pose any threat to your Linux machine, it might be an issue to the internal machine that this traffic is being forwarded to. No worries though, this is what Steven Clarke (the author of IPPORTFW) had to say about that:

```
"Port Forwarding is only called within masquerading functions so it
fits inside the same IPFWADM/IPCHAINS rules. Masquerading is an extension to
IP forwarding. Therefore, ipportfw only sees a packet if it fits
both the input and masquerading ipfwadm rule sets."
```

With this said, it's important to have a strong firewall ruleset. Please see the Strong−IPFWADM−Rulesets and Strong−IPCHAINS−Rulesets sections for more details on strong rulesets.

So, to install IPPORTFW forwarding support for either a 2.2.x or 2.0.x kernel, you need to re−compile the Linux kernel to support IPPORTFW.

- 2.2.x kernel users will already have the IPPORTFW kernel option available via IPMASQADM
- 2.0.x users will need to apply a simple kernel option patch

## IPMASQADM with IPPORTFW support on 2.2.x kernels

First, make sure you have the newest 2.2.x kernel uncompressed into /usr/src/linux. If you haven't already done this, please see the Kernel−Compile section for full details. Next, download the "ipmasqadm.c" program from the 2.2.x−Requirements section into the /usr/src/ directory.

Next, you'll need to compile the 2.2.x kernel as shown in the Kernel−Compile section. Be sure to say YES to the IPPORTFW option when you configure the kernel. Once the kernel compile is complete and you have rebooted, return to this section.

Now, compile and install the IPMASQADM tool:

```
cd /usr/src
tar xzvf ipmasqadm-x.tgz
cd ipmasqadm-x
make
make install
```

Now, for this example, we are going to allow ALL WWW Internet traffic (port 80) hitting your Internet TCP/IP address to then be forwarded to the internal Masqueraded machine at IP address 192.168.0.10.

PORTFW FTP: As mentioned above, there are two solutions for forwarding FTP server traffic to an internal MASQed PC. The first solution *IS* a BETA level *IP_MASQ_FTP* module for 2.2.x kernels to PORT Forward FTP connections to an internal MASQed FTP server. The other method is using a FTP proxy program (the URL is in the 2.2.x−Requirements section. It should also be noted that the FTP kernel module also supports the adding of additional PORTFW FTP ports on the fly without the requirement of unloading and reloaded the IP_MASQ_FTP module and thus breaking any existing FTP transfers. You can find more about this new code at the IPMASQ WWW site at http://ipmasq.cjb.net. There is also examples and some more information about PORTFWed FTP connection below in the 2.0.x. kernel section.

**NOTE:** Once you enable a port forwarder on port 80, that port can no longer be used by the Linux IP Masquerade server. To be more specific, if you have a WWW server already running on the MASQ server, a port forward will now give all Internet users the WWW pages from the −INTERNAL− WWW server and not the pages on your IP MASQ server.

Anyway, to enable port forwarding, edit the /etc/rc.d/rc.firewall ruleset. Add the follow lines but be sure to replace the word "$extip" with your Internet IP address.

**NOTE:** If you use get a DYNAMIC TCP/IP address from your ISP (PPP, ADSL, Cablemodems, etc.), you will NEED to make your /etc/rc.d/rc.firewall ruleset more intelligent. To do this, please see the Strong−IPCHAINS−Rulesets section from above or TrinityOS − Section 10 for more details on strong rulesets and Dynamic IP addresses. I'll give you a hint though: /etc/ppp/ip−up for PPP users.

```
/etc/rc.d/rc.firewall
--

#echo "Enabling IPPORTFW Redirection on the external LAN.."
#
/usr/sbin/ipmasqadm portfw -f
/usr/sbin/ipmasqadm portfw -a -P tcp -L $extip 80 -R 192.168.0.10 80

--
```

That's it! Just re−run your /etc/rc.d/rc.firewall ruleset and test it out!

If you get the error message "ipchains: setsockopt failed: Protocol not available", you AREN'T running your new kernel. Make sure that you moved the new kernel over, re−run LILO, and then reboot again. If you are sure you are running your new kernel, run the command "ls /proc/net/ip_masq" and make sure the "portfw" file exists. If it doesn't, you must have made an error when configuring your kernel. Try again.

For those who want to understand why PORTFW cannot redirect traffic for both external and internal interfaces, here is an email from Juanjo that better explains it:

```
From Juanjo Ciarlante
--

>If I use:
>
> ipmasqadm portfw -a -P tcp  -L 1.2.3.4 80 -R 192.168.2.3 80
>
>Everything works great from the outside but internal requests for the same
>1.2.3.4 address fail. Are there chains that will allow a machine on localnet
>192.168.2.0 to accesss www.periapt.com without using a proxy?

Actually not.

I usually setup a ipmasqadm rule for outside, *AND* a port
redirector for inside. This works because ipmasqadm hooks before
redir will get the eventual outside connection, _but_ leaves things
ok if not (stated by APPROPIATE rules).

The actual "conceptual" problem comes from the TRUE client (peer) IP
goal (thanks to masq) being in same net as target server.

The failing scenario for "local masq" is :
   client: 192.168.2.100
   masq:   192.168.2.1
   serv:   192.168.2.10

1)client->server packet
 a) client:  192.168.2.100:1025  -> 192.168.2.1:80   [SYN]
 b) (masq):  192.168.2.100:1025  -> 192.168.2.10:80  [SYN]
            (and keep  192.168.2.1:61000 192.168.2.100:1025 related)
 c) serv:    gets masqed packet (1b)

2)server->client packet
 a) serv:    192.168.2.10:80     -> 192.168.2.100:1025  [SYN,ACK]
 b) client:  192.168.2.100:1025  -> 192.168.2.10:80     [RST]

Now take a moment to compare (1a) with (2a).
You see, the server replied DIRECTLY to client bypassing masq (not
letting masq to UNDO the packet hacking) because it is in SAME net, so
the client resets the connection.

hope I helped.

Warm regards
Juanjo
```

## IPPORTFW on 2.0.x kernels

First, make sure you have the newest 2.0.x kernel uncompressed into /usr/src/linux. If you haven't already done this, please see the Kernel−Compile section for full details. Next, download the "ipportfw.c" program and the "subs−patch−x.gz" kernel patch from the 2.0.x−Requirements section into the /usr/src/ directory.

NOTE: Please replace the "x" in the "subs−patch−x.gz" file name with the most current version available on the site.

Next, if you plan on port forwarding FTP traffic to an internal server, you will have to apply an additional **NEW** *IP_MASQ_FTP* module patch found in the 2.0.x−Requirements section. More details regarding this are later in this section. Please note that this is NOT the same patch as for the 2.2.x kernels so some functionality such as the dynamic FTP PORT functionality is not present.

Now, copy the IPPORTFW patch (subs−patch−x.gz) into the Linux directory

```
cp /usr/src/subs−patch−1.37.gz /usr/src/linux
```

Next, apply the kernel patch to create the IPPORTFW kernel option:

```
cd /usr/src/linux
zcat subs−patch−1.3x.gz | patch −p1
```

Ok, time to compile the kernel as shown in the Kernel−Compile section. Be sure to say YES to the IPPORTFW option now available when you configure the kernel. Once the compile is complete and you have rebooted, return to this section.

Now with a newly compiled kernel, please compile and install the actual "IPPORTFW" program

```
cd /usr/src
gcc ipportfw.c −o ipportfw
mv ipportfw /usr/local/sbin
```

Now, for this example, we are going to allow ALL WWW Internet traffic (port 80) hitting your Internet TCP/IP address to then be forwarded to the internal Masqueraded machine at IP address 192.168.0.10.

**NOTE:** Once you enable a port forwarder on port 80, that port can no longer be used by the Linux IP Masquerade server. To be more specific, if you have a WWW server already running on the MASQ server and then you port forward port 80 to an internal MASQed computer, ALL internet users will see the WWW pages pages from the −INTERNAL− WWW server and not the pages on your IP MASQ server. The only work around for this is to port forward some other port, say 8080, to your internal MASQ machine. Though this will work, all Internet users will have to append *:8080* to the URL to then contact the internal MASQed WWW server.

Anyway, to enable port forwarding, edit the */etc/rc.d/rc.firewall* ruleset. Add the follow lines but be sure to replace the word "$extip" with your Internet IP address.

**NOTE:** If you use get a DYNAMIC TCP/IP address from your ISP (PPP, ADSL, Cablemodems, etc.), you will NEED to make your /etc/rc.d/rc.firewall ruleset more intelligent. To do this, please see the Strong−IPCHAINS−Rulesets section from above or TrinityOS − Section 10 for more details on strong rulesets and Dynamic IP addresses. I'll give you a hint though: /etc/ppp/ip−up for PPP users.

```
/etc/rc.d/rc.firewall
−−

#echo "Enabling IPPORTFW Redirection on the external LAN.."
#
/usr/local/sbin/ipportfw −C
/usr/local/sbin/ipportfw −A −t$extip/80 −R 192.168.0.10/80
```

```
#Please note that PORTFWing port 20 is NOT required for ACTIVE
#  connections as the internal FTP server will initiate this
#  port 20 connection and it will be properly handled by the
#  classic MASQ mechanisms.
   --
```

That's it! Just re−run your /etc/rc.d/rc.firewall ruleset and test it out!

If you get the error message "ipfwadm: setsockopt failed: Protocol not available", you AREN'T running your new kernel. Make sure that you moved the new kernel over, re−run LILO, and then reboot again.

Port Forwarding FTP servers:

If you plan on port forwarding FTP to an internal machine, things get more complicated. The reason for this is because the standard *IP_MASQ_FTP* kernel module wasn't written for this though some users report that it works without any problems. Personally, without the patch, I've heard that extended file transfers in excess of 30 minutes will fail without the patch while other people swear that it works flawlessly. Anyway, I recommend that you try the following PORTFW instruction with the STOCK ip_masq_ftp module and see if it works for you. If it doesn't, try using the modified ip_masq_ftp module.

For those who need the module, Fred Viles wrote a modified IP_MASQ_FTP module to make things work. If you are curious what EXACTLY is the issues, download the following archive since Fred documents it quite well. Also understand that this patch is somewhat experimental and should be treated as such. It should be also noted that this patch is ONLY available for the 2.0.x kernels though there is a different patch available for 2.2.x kernels.

So, to get the 2.0.x patch working, you need to:

- Apply the IPPORTFW kernel patch as shown earlier in this section FIRST.
- Download the "msqsrv−patch−36" from Fred Viles's FTP server in the 2.0.x−Requirements section and put it into /usr/src/linux.
- Patch the kernel with this new code by running "cat msqsrv−patch−36 | patch −p1"
- Next, replace the original *"ip_masq_ftp.c"* kernel module with the new one
    - mv /usr/src/linux/net/ipv4/ip_masq_ftp.c /usr/src/linux/net/ipv4/ip_masq_ftp.c.orig
    - mv /usr/src/linux/ip_masq_ftp.c /usr/src/linux/net/ipv4/ip_masq_ftp.c
- Lastly build and install the kernel with this new code in place.

Once this is complete, edit the /etc/rc.d/rc.firewall ruleset and add the follow lines but be sure to replace the word "$extip" with your Internet IP address.

This example, like above, will allow ALL FTP Internet traffic (port 21) hitting your Internet TCP/IP address to then be forwarded to the internal Masqueraded machine at IP address 192.168.0.10.

NOTE: Once you enable a port forwarder on port 21, that port can no longer be used by the Linux IP Masquerade server. To be more specific, if you have a FTP server already running on the MASQ server, a port forward will now give all Internet users the FTP files from the −INTERNAL− FTP server and not the files on your IP MASQ server.

```
/etc/rc.d/rc.firewall
--

#echo "Enabling IPPORTFW Redirection on the external LAN.."
#
/usr/local/sbin/ipportfw -C
/usr/local/sbin/ipportfw -A -t$extip/21 -R 192.168.0.10/21

#NOTE: If you are using multiple local port numbers to PORTFW
#      to multuple internal FTP servers (say, 21, 2121, 2112,
#      etc, you need to configure the ip_masq_ftp nodule to
#      listen to these ports.  To do this, edit the
#      /etc/rc.d/rc.firewall script as shown in this HOWTO
#      to look like:
#
# /sbin/modprobe ip_masq_ftp ports=21,2121,2112
#
# Re-run the /etc/rc.d/rc.firewall script for these changes to
# take effect.


#Please note that PORTFWing port 20 is probably NOT required
#  for ACTIVE connections as the internal FTP server will
#  initiate this port 20 connection and it will be properly
        #  handled by the classic MASQ mechanisms.
--
```

That's it! Just re−run your /etc/rc.d/rc.firewall ruleset and test it out!

If you get the error message "ipchains: setsockopt failed: Protocol not available", you AREN'T running your new kernel. Make sure that you moved the new kernel over, re−run LILO, and then reboot again. If you are sure you are running your new kernel, run the command "ls /proc/net" and make sure the "ip_portfw" file exists. If it doesn't, you must have made an error when configuring your kernel. Try again.

## 6.9 CU−SeeMe and Linux IP−Masquerade

Linux IP Masquerade supports CuSeeme via the *"ip_masq_cuseeme"* kernel module. This kernel modules should be loaded in the /etc/rc.d/rc.firewall script. Once the "ip_masq_cuseeme" module is installed, you should be able to both initiate and receive CuSeeme connections to remote reflectors and/or users.

NOTE: It is recommended to use the IPPORTFW tool instead of the old IPAUTOFW tool for running CuSeeme.

If you need more explicit information on configuring CuSeeme, see Michael Owings's CuSeeMe page for a Mini−HOWTO or The IP Masquerade Resources for a mirror of the Mini−HOWTO.

## 6.10 Mirabilis ICQ

There are two methods of getting ICQ to work behind a Linux MASQ server. One solution is to use a new ICQ Masq module and the other solution is to use IPPORTFW.

The ICQ module has some benefits. It allows for simple setup of multiple ICQ users behind a MASQ server. It also doesn't require any special changes to the ICQ client(s). Recently, the 2.2.x version of the module was updated to support file transfer and read−time chat. Yet, for the 2.0.x kernel module, file transfers and real−time chat still isn't fully supported. Anyway, I now feel this is the PREFERRED method to get ICQ working with IP Masq running on 2.2.x+ kernels.

With the IPPORTFW setup, you will have to make some changes on both Linux and ICQ clients but all ICQ messaging, URLs, chat, file transfer, etc. work.

If you are interested in Andrew Deryabin's djsf@usa.net ICQ IP Masq module for the 2.2.x kernels. Please see the 2.2.x−Requirements section for details.

If you rather use the classic method of getting ICQ to run behind a MASQ server, follow these steps:

- First, you need to be running a Linux kernel with IPPPORTFW enabled. Please see the Forwarders section for more details.

    - Next, you need to add the following lines to your /etc/rc.d/rc.firewall file. This example assumes that 10.1.2.3 is your external Internet IP address and your internal MASQed ICQ machine is 192.168.0.10:

      The following example is for a 2.0.x kernel with IPFWADM:

```
I have included two examples here for the user:  Either once works
fine:

Example #1
--
/usr/local/sbin/ipportfw -A -t10.1.2.3/2000 -R 192.168.0.10/2000
/usr/local/sbin/ipportfw -A -t10.1.2.3/2001 -R 192.168.0.10/2001
/usr/local/sbin/ipportfw -A -t10.1.2.3/2002 -R 192.168.0.10/2002
/usr/local/sbin/ipportfw -A -t10.1.2.3/2003 -R 192.168.0.10/2003
/usr/local/sbin/ipportfw -A -t10.1.2.3/2004 -R 192.168.0.10/2004
/usr/local/sbin/ipportfw -A -t10.1.2.3/2005 -R 192.168.0.10/2005
/usr/local/sbin/ipportfw -A -t10.1.2.3/2006 -R 192.168.0.10/2006
/usr/local/sbin/ipportfw -A -t10.1.2.3/2007 -R 192.168.0.10/2007
/usr/local/sbin/ipportfw -A -t10.1.2.3/2008 -R 192.168.0.10/2008
/usr/local/sbin/ipportfw -A -t10.1.2.3/2009 -R 192.168.0.10/2009
/usr/local/sbin/ipportfw -A -t10.1.2.3/2010 -R 192.168.0.10/2010
/usr/local/sbin/ipportfw -A -t10.1.2.3/2011 -R 192.168.0.10/2011
/usr/local/sbin/ipportfw -A -t10.1.2.3/2012 -R 192.168.0.10/2012
/usr/local/sbin/ipportfw -A -t10.1.2.3/2013 -R 192.168.0.10/2013
/usr/local/sbin/ipportfw -A -t10.1.2.3/2014 -R 192.168.0.10/2014
/usr/local/sbin/ipportfw -A -t10.1.2.3/2015 -R 192.168.0.10/2015
/usr/local/sbin/ipportfw -A -t10.1.2.3/2016 -R 192.168.0.10/2016
/usr/local/sbin/ipportfw -A -t10.1.2.3/2017 -R 192.168.0.10/2017
/usr/local/sbin/ipportfw -A -t10.1.2.3/2018 -R 192.168.0.10/2018
/usr/local/sbin/ipportfw -A -t10.1.2.3/2019 -R 192.168.0.10/2019
/usr/local/sbin/ipportfw -A -t10.1.2.3/2020 -R 192.168.0.10/2020
--

Example #2
--
port=2000
while [ $port -le 2020 ]
  do
      /usr/local/sbin/ipportfw -A t10.1.2.3/$port -R 192.168.0.10/$port
      port=$((port+1))
  done
```

```
--
```

The following example is for a 2.2.x kernel with IPCHAINS:

```
I have included two examples here for the user:  Either once works
fine:

Example #1
--
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2000 -R 192.168.0.10 2000
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2001 -R 192.168.0.10 2001
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2002 -R 192.168.0.10 2002
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2003 -R 192.168.0.10 2003
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2004 -R 192.168.0.10 2004
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2005 -R 192.168.0.10 2005
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2006 -R 192.168.0.10 2006
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2007 -R 192.168.0.10 2007
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2008 -R 192.168.0.10 2008
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2009 -R 192.168.0.10 2009
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2010 -R 192.168.0.10 2010
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2011 -R 192.168.0.10 2011
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2012 -R 192.168.0.10 2012
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2013 -R 192.168.0.10 2013
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2014 -R 192.168.0.10 2014
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2015 -R 192.168.0.10 2015
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2016 -R 192.168.0.10 2016
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2017 -R 192.168.0.10 2017
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2018 -R 192.168.0.10 2018
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2019 -R 192.168.0.10 2019
/usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 2020 -R 192.168.0.10 2020
--

Example #2
--
port=2000
while [ $port -le 2020 ]
  do
      /usr/local/sbin/ipmasqadm portfw -a -P tcp -L 10.1.2.3 $port -R 192.168.0.10 $port
      port=$((port+1))
  done
--
```

- Once your new rc.firewall is ready, reload the ruleset to make sure things are ok by simple typing in "/etc/rc.d/rc.firewall". If you get any errors, you either don't have IPPORTFW support in the kernel or you made a typo in the rc.firewall file.

- Now, in ICQ's Preferences−−>Connection, configure it to be "Behind a LAN" and "Behind a firewall or Proxy". Now, click on "Firewall Settings" and configure it to be "I don't use a SOCK5 proxy". Also note that it was repviously recommended to change ICQ's "Firewall session timeouts" to "30" seconds BUT many users have found that ICQ becomes unreliable. It has been found that ICQ is more reliable with its stock timeout setting (don't enable that ICQ option) and simply change MASQ's timeout to 160 seconds. You can see how to change this timeout in the rc.firewall−2.0.x and rc.firewall−2.2.x rulesets. Finally, click on Next and configure ICQ to "Use the following TCP listen ports.." from "2000" to "2020". Now click done.

Now ICQ will tell you that you have to restart ICQ for the changes to take effect. To be honest, I had to REBOOT the Windows9x machine to get things to work right but other people say otherwise. So.. try it both ways.

- It should also be noted that one user told me that simply portforwarding port 4000 to his ICQ machine worked best. He reported that everything worked fine (chat, file transfers, etc) WITHOUT re−configuring ICQ from its default settings. Your mileage might vary on this topic but I though you might like to hear about this alternative configuration.

## 6.11 Gamers: The LooseUDP patch

The LooseUDP patch allows NAT−friendly games that usually use UDP connections to both WORK and perform quite well behind a Linux IP Masquerade server. Currently, LooseUDP is available as a patch for 2.0.36+ kernels and it is already built into 2.2.3+ kernels though it is now DISABLED by DEFAULT in 2.2.16+.

To get LooseUDP running on a 2.0.x kernel, follow the following steps:

- Have the newest 2.0.x kernel sources uncompressed in the /usr/src/linux directory
- ABSOLUTELY REQUIRED for v2.0.x: Download and install the IPPORTFW patch from the 2.0.x−Requirements section and as described in the Forwarders Section of the HOWTO.
- Download the LooseUDP patch from the 2.0.x−Requirements section

  Now, put the LooseUDP patch in the /usr/src/linux directory. Once this is done, type in:

  ```
  For a compressed patch file: zcat
  loose-udp-2.0.36.patch.gz | patch -p1

  For a NON-compressed patch file: cat
  loose-udp-2.0.36.patch | patch -p1
  ```

  Now, depending on your version of "patch", You will then see the following text:

  ```
  patching file `CREDITS'
  patching file `Documentation/Configure.help'
  patching file `include/net/ip_masq.h'
  patching file `net/ipv4/Config.in'
  patching file `net/ipv4/ip_masq.c'
  ```

  If you see the text "Hunk FAILED" only ONCE and ONLY ONCE at the very beginning of the patching, don't be alarmed. You probably have an old patch file (this as been fixed) but it still works. If it fails completely, make sure you have applied the IPPORTFW kernel patch FIRST.

  Once the patch is installed, re−configure the kernel as shown in the Kernel−Compile section and be sure to say "Y" to the "IP: loose UDP port managing (EXPERIMENTAL) (CONFIG_IP_MASQ_LOOSE_UDP) [Y/n/?]" option.

To get LooseUDP running on a 2.2.x kernel, follow the following steps:

- In the /etc/rc.d/rc.firewall script, goto the BOTTOM of the file and find the LooseUDP section. Change the "0" in the line: `echo "0" > /proc/sys/net/ipv4/ip_masq_udp_dloose` to a "1" and re−run the rc.firewall ruleset. An example of this is given in both the rc.firewall−2.2.x example and the stronger−rc.firewall−2.2.x example.

Once you are running the new LooseUDP enabled kernel, you should be good to go for most NAT−friendly games. Some URLs have been given for patches to make games like BattleZone and others NAT friendly. Please see the Game−Clients section for more details.

---

## 7. Frequently Asked Questions

If you can think of any useful FAQ suggestions, please send it to dranch@trinnet.net. Please clearly state the question and an appropriate answer (if you have it). Thank you!

### 7.1 What Linux Distributions support IP Masquerading out of the box?

If your Linux distribution doesn't support IP MASQ out of the box, don't worry. All you have to do is re−compile a kernel as shown above in this HOWTO.

NOTE: If you can help us fill out this table, please email ambrose@writeme.com or dranch@trinnet.net.

- Caldera < v1.2 : NO − ?
- Caldera v1.3 : YES − 2.0.35 based
- Caldera v2.2 : YES − 2.2.5 based
- Caldera eServer v2.3 : YES − ? based
- Debian v1.3 : NO − ?
- Debian v2.0 : NO − ?
- Debian v2.1 : YES − 2.2.1 based
- Debian v2.2 : YES − 2.2.15 based
- DLX Linux v? : ? − ?
- DOS Linux v? : ? − ?
- FloppyFW v1.0.2 : ? − ?
- Hal91 Linux v? : ? − ?
- Linux Mandrake v5.3 : YES − ?
- Linux Mandrake v6.0 : YES − 2.2.5 based
- Linux PPC vR4 : NO − ?
- Linux Pro v? : ? − ?
- LinuxWare v? : ? − ?
- Mandrake v6.0 : YES − ?
- Mandrake v6.1 : YES − ?
- Mandrake v7.0 : YES − 2.2.14
- Mandrake v7.1 : YES − 2.2.15
- Mandrake v7.2 : YES − 2.2.17
- MkLinux v? : ? − ?
- MuLinux v3rl : YES − ?

- Redhat < v4.x : NO − ?
- Redhat v5.0 : YES − ?
- Redhat v5.1 : YES − 2.0.34 based
- Redhat v5.2 : YES − 2.0.36 based
- Redhat v6.0 : YES − 2.2.5 based
- Redhat v6.1 : YES − 2.2.12 based
- Redhat v6.2 : YES − 2.2.14 based
- Redhat v7.0 : YES − 2.2.16 based
- Slackware v3.0 : ? − ?
- Slackware v3.1 : ? − ?
- Slackware v3.2 : ? − ?
- Slackware v3.3 : ? − 2.0.34 based
- Slackware v3.4 : ? − ?
- Slackware v3.5 : ? − ?
- Slackware v3.6 : ? − ?
- Slackware v3.9 : ? − 2.0.37pre10 based
- Slackware v4.0 : ? − ?
- Slackware v7.0 : YES − 2.2.13 based
- Slackware v7.1 : YES − 2.2.16 based
- Stampede Linux v? : ? − ?
- SuSE v5.2 : YES − 2.0.32 base
- SuSE v5.3 : YES − ?
- SuSE v6.0 : YES − 2.0.36 based
- SuSE v6.1 : YES − 2.2.5 based
- SuSE v6.3 : YES − 2.2.13 based
- Tomsrbt Linux v? : ? − ?
- TurboLinux Lite v4.0 : YES − ?
- TurboLinux v6.0 : YES − 2.2.12 based
- TriLinux v? : ? − ?
- Yggdrasil Linux v? : ? − ?

## 7.2 What are the minimum hardware requirements and any limitations for IP Masquerade? How well does it perform?

A 486/66 box with 16MB of RAM was more than sufficient to fill a 1.54Mb/s T1 100%! MASQ has also be known run quite well on 386SX−16s with 8MB of RAM. Yet, it should be noted that Linux IP Masquerade starts thrashing with more than 500 MASQ entries.

The only application that I known that can temporarily break Linux IP Masquerade is GameSpy. Why? When it refreshes its lists, it creates 10,000s of quick connections in a VERY short time. Until these sessions timeout, the MASQ tables become "FULL". See the No−Free−Ports section of the FAQ for more details.

While we are at it:

There is a hard limit of 4096 concurrent connections each for TCP & UDP. This limit can be changed by fiddling the values in */usr/src/linux/net/ipv4/ip_masq.h* − a upwards limit of 32000 should by OK. If you want to change the limit − you need to change the PORT_MASQ_BEGIN & PORT_MASQ_END values to get an appropriately sized range above 32K and below 64K.

## 7.3 When I run the rc.firewall command, I get "command not found" errors. Why?

How did you put the rc.firewall onto your machine? Did you cut&paste it into a TELNET window, FTP it from a Windows/DOS machine, etc? Try this.. log into your Linux box and run "vim −b /etc/rc.d/rc.firewall" and see if all your lines end in a ^M. If they do, delete all the ^M and try again.

## 7.4 I've checked all my configurations, I still can't get IP Masquerade to work. What should I do?

- Stay calm. Get yourself a cup of tea, coffee, soda, etc., and have a rest. Once your mind is clear, try the suggestions mentioned below. Setting up Linux IP Masquerading is NOT hard but there are several concepts that will be new to you.
- Again, go through all the steps in the Testing section. 99% of all first−time Masquerade users who have problems haven't looked here.
- Check the IP Masquerade Mailing List Archives, most likely your question or problem is a common one and can be found in a simple Archive search.
- Check out the TrinityOS document. It covers IP Masquerading for both the 2.0.x and 2.2.x kernels and MANY other topics including PPPd, DialD, DHCP, DNS, Sendmail, etc.
- Make sure that you aren't running ROUTED or GATED. To verify, run "ps aux | grep −e routed −e gated"
- Post your question to the IP Masquerade Mailing List (see next the FAQ section for details). Please only use this if you cannot find the answer from the IP Masquerading Archive. Be sure to include all the information requested in the Testing section in your email!!
- Post your question to a related Linux NNTP newsgroup.
- Send email to ambrose@writeme.com and dranch@trinnet.net. You have a better chance of getting a reply from the IP Masquerading Email list than either of us.
- Check your configurations again :−)

## 7.5 How do I join or view the IP Masquerade and/or IP Masqurade Developers mailing lists and archives?

There are two ways to join the two Linux IP Masquerading mailing lists. The first way is to send an email to masq−request@indyramp.com. To join the Linux IP Masquerading Developers mailing list, send an email to masq−dev−request@indyramp.com. Please see the bullet below for more details.

- Subscribe via email: Now put the word "subscribe" in either the subject or body of the e−mail message. If you want to only subscribe to the Digest version of either the main MASQ or MASQ−DEV list (all e−mails on the given list during the week are sent to you in one big email), put the words "subscribe digest" instead in either the subject or body of the e−mail message.

Once the server receives your request, it will subscribe you to your requested list and give you a PASSWORD. Save this password as you will needed to to later unsubscribe from the list or change your options.

The second method is to use a WWW browser and subscribe via a form at
http://www.indyramp.com/masq−list/ for the main MASQ list or
http://www.indyramp.com/masq−dev−list/ for the MASQ−DEV list.

Once subscribed, you will get emails from your subscribed list. It should be also noted that both subscribed and NON−subscribed users can access the two list's archives. To do this, please see the above two WWW URLs for more details.

Lastly, please note that you can only post to the MASQ list from an account/address you originally subscribed from.

If you have any problem regarding the mailing lists or the mailing list archive, please contact Robert Novak.

## 7.6 How does IP Masquerade differ from Proxy or NAT services?

```
Proxy:  Proxy servers are available for: Win95, NT, Linux, Solaris, etc.

                Pro:    + (1) IP address ; cheap
                        + Optional caching for better performance (WWW, etc.)

                Con:    - All applications behind the proxy server must both SUPPORT
                          proxy services (SOCKS) and be CONFIGURED to use the Proxy
                          server
                        - Screws up WWW counters and WWW statistics

         A proxy server uses only (1) public IP address, like IP MASQ, and acts
         as a translator to clients on the private LAN (WWW browser, etc.).
         This proxy server receives requests like TELNET, FTP, WWW,
         etc. from the private network on one interface.  It would then in turn,
         initiate these requests as if someone on the local box was making the
         requests.   Once the remote Internet server sends back the requested
         information, it would re-translate the TCP/IP addresses back to the
         internal MASQ client and send traffic to the internal requesting host.
         This is why it is called a PROXY server.

                Note:  ANY applications that you might want to use on the
                        internal machines *MUST* have proxy server support
                        like Netscape and some of the better TELNET and FTP
                        clients.  Any clients that don't support proxy servers
                        won't work.

         Another nice thing about proxy servers is that some of them
         can also do caching (Squid for WWW).  So, imagine that you have 50
         proxied hosts all loading Netscape at once.  If they were installed
         with the default homepage URL, you would have 50 copies of the same
         Netscape WWW page coming over the WAN link for each respective computer.
         With a caching proxy server, only one copy would be downloaded by the proxy
```

```
         server and then the proxied machines would get the WWW page from the
         cache.  Not only does this save bandwidth on the Internet connection,
         it will be MUCH MUCH faster for the internal proxied machines.



MASQ:    IP Masq is available on Linux and a few ISDN routers such
 or      as the Zytel Prestige128, Cisco 770, NetGear ISDN routers, etc.
1:Many
 NAT
                Pro:    + Only (1) IP address needed (cheap)
                        + Doesn't require special application support
                        + Uses firewall software so your network can become
                          more secure

                Con:    - Requires a Linux box or special ISDN router
                          (though other products might have this..  )
                        - Incoming traffic cannot access your internal LAN
                          unless the internal LAN initiates the traffic or
                          specific port forwarding software is installed.
                          Many NAT servers CANNOT provide this functionality.
                        - Special protocols need to be uniquely handled by
                          firewall redirectors, etc.  Linux has full support
                          for this (FTP, IRC, etc.) capabilty but many routers
                          do NOT (NetGear DOES).

         Masq or 1:Many NAT is similar to a proxy server in the sense that the
         server will do IP address translating and fake out the remote server
         (WWW for example) as if the MASQ server made the request instead of an
         internal machine.

         The major difference between a MASQ and PROXY server is that MASQ servers
         don't need any configuration changes to all the client machines.  Just
         configure them to use the linux box as their default gateway and everything
         will work fine.  You WILL need to install special Linux modules for things
         like RealAudio, FTP, etc. to work)!

         Also, many people use IP MASQ for TELNET, FTP, etc. *AND* also setup a
         caching proxy on the same Linux box for WWW traffic for the additional
         performance.


NAT:     NAT servers are available on Windows 95/NT, Linux, Solaris, and some
         of the better ISDN routers (not Ascend)

                Pro:    + Very configurable
                        + No special application software needed

                Con:    - Requires a subnet from your ISP (expensive)

         Network Address Translation is a name for a box that would have a pool of
         valid IP addresses on the Internet interface that it can use.  When on the
         Internal network wanted to goto the Internet, it associates an available
         VALID IP address from the Internet interface to the original requesting
         PRIVATE IP address.  After that, all traffic is re-written from the NAT
         public IP address to the NAT private address.  Once the associated PUBLIC
         NAT address becomes idle for some pre-determined amount of time, the
         PUBLIC IP address is returned back into the public NAT pool.

         The major problem with NAT is, once all of the free public IP addresses are
         used, any additional private users requesting Internet service are out of
         luck until a public NAT address becomes free.
```

For an excellent and very comprehensive description of the various forms of NAT, please see:

- http://www.suse.de/~mha/linux−ip−nat/diplom/nat.html

Here is another good site to learn about NAT though many of the URLs are old but still valid:

- http://www.linas.org/linux/load.html

This is a great URL for learning about other NAT solutions for Linux as well as other platforms:

- http://www.uq.net.au/~zzdmacka/the−nat−page/

## 7.7 Are there any GUI firewall creation/management tools?

Yes! They vary in user interface, complexity, etc. but they are quite good though most are only for the IPFWADM tool so far. Here is a short list of available tools in alphabetical order. If you know of any others or have any thoughts on which ones are good/bad/ugly, please email David.

- John Hardin's IPFWADM Dot file generator − a IPCHAINS version is in the works
- Sonny Parlin's fBuilder: From the author of FWCONFIG, this new solution is fully WWW based and offers redundancy options, etc for both IPCHAINS and NetFilter.
- William Stearns's Mason − A Build−a−ruleset on−the−fly type system

## 7.8 Does IP Masquerade work with dynamically assigned IP addresses?

**Yes**, it works with either dynamic IP addressed assigned by your ISP via either PPP or a DHCP/BOOTp server. As long as you have an valid Internet IP address, it should work. Of course, static IP works too. Yet, if you plan on implementing a strong IPFWADM/IPCHAINS ruleset and/or plan on using a Port forwarder, your ruleset will have to be re−executed everytime your IP address changes. Please see the top of TrinityOS − Section 10 for additional help with strong firewall rulesets and Dynamic IP addresses.

## 7.9 Can I use a cable modem (both bi−directional and with modem returns), DSL, satellite link, etc. to connect to the Internet and use IP Masquerade?

**Yes**, as long as Linux supports that network interface, it should work. If you receive a dynamic IP address, please see the URL under the "Does IP Masquerade work with dynamically assigned IP" FAQ item above.

## 7.10 Can I use Diald or the Dial−on−Demand feature of PPPd with IP MASQ?

Definitely! IP Masquerading is totally transparent to Diald or PPP. The only thing that might become an issue is if you use STRONG firewall rulesets with dynamic IP addresses. See the FAQ item, "Does IP Masquerade work with dynamically assigned IP addresses?" above for more details.

## 7.11 What applications are supported with IP Masquerade?

It is very difficult to keep track of a list of "working applications". However, most of the normal Internet applications are supported, such as WWW browsing (Netscape, MSIE, etc.), FTP (such as WS_FTP), TELNET, SSH, RealAudio, POP3 (incoming email -- Pine, Eudora, Outlook), SMTP (outgoing email), etc. A somewhat more complete list of MASQ--compatible clients can be found in the Clients section of this HOWTO.

Applications involving more complicated protocols or special connection methods such as video conferencing software need special helper tools.

For more detail, please see the Linux IP masquerading Applications page.

## 7.12 How can I get IP Masquerade running on Redhat, Debian, Slackware, etc.?

No matter what Linux distribution you have, the procedures for setting up IP Masquerade mentioned in this HOWTO should apply. Some distributions may have GUI or special configuration files that make the setup easier. We try our best to write the HOWTO as general as possible.

## 7.13 TELNET connections seem to break if I don't use them often. Why is that?

IP Masq, by default, sets its timers for TCP session, TCP FIN, and UDP traffic to 15 minutes. It is recommend to use the following settings (as already shown in this HOWTO's /etc/rc.d/rc.firewall ruleset) for most users:

Linux 2.0.x with IPFWADM:

```
# MASQ timeouts
#
#   2 hrs timeout for TCP session timeouts
#  10 sec timeout for traffic after the TCP/IP "FIN" packet is received
#  60 sec timeout for UDP traffic (MASQ'ed ICQ users must enable a 30sec firewall timeout in ICQ itself)
#
/sbin/ipfwadm -M -s 7200 10 60
```

Linux 2.2.x with IPCHAINS:

```
# MASQ timeouts
#
#   2 hrs timeout for TCP session timeouts
#  10 sec timeout for traffic after the TCP/IP "FIN" packet is received
#  60 sec timeout for UDP traffic (MASQ'ed ICQ users must enable a 30sec firewall timeout in ICQ itself)
#
/ipchains -M -S 7200 10 60
```

## 7.14 When my Internet connection first comes up, nothing works. If I try again, everything then works fine. Why is

## this?

The reason is because you have a dynamic IP address and when your Internet connection first comes up, IP Masquerade doesn't know its IP address. There is a solution to this. In your /etc/rc.d/rc.firewall ruleset, add the following:

```
# Dynamic IP users:
#
#   If you get your IP address dynamically from SLIP, PPP, or DHCP, enable this following
#       option.  This enables dynamic-ip address hacking in IP MASQ, making the life
#       with Diald and similar programs much easier.
#
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
```

## 7.15 ( MTU ) − IP MASQ seems to be working fine but some sites don't work. This usually happens with WWW and FTP.

There is two possible reasons for this. The first one is VERY common and the second is very UNCOMMON.

- As of the 2.0.38 and 2.2.9+ Linux kernels, there is a debatable BUG in the Masquerade code.

  Some users point their finger to the fact that IPMASQ might have problems with packets that have the DF or "Don't Fragment" bit set. Basically, when a MASQ box connects to the Internet with an MTU of anything less than 1500, some packets will have the DF field set. Though changing the MTU 1500 on the Linux box will seemingly fix the problem, the possible bug is still there. What is believed to be happening is that the MASQ code is not properly re−writing the returning ICMP packets with the ICMP 3 Sub 4 code back to the originating MASQed computer. Because of this, the packets get dropped.

  Other users point their finger at the adminstrators of the problem remote sites (typically SSL connected sites, etc) and say that because they are filtering ALL FORMS of ICMP (including Type4 − Fragmentation Needed) messages in a fray of security paranoia, they are breaking the fundamental aspects of the TCP/IP protocol.

  Both arguments have valid points and people from each camp continue to debate this to this day. If you are a network programmer and you think you can either fix or surmise this.. PLEASE TRY! For more details, check out this following MTU Thread from the Linux−Kernel list.

  No worries though. A perfectly good workaround is to change your Internet link's MTU to 1500. Now some users will balk at this because it can hurt some latency specific programs like TELNET and games but the impact is only slight. On the flip site, most HTTP and FTP traffic will SPEED UP!

  [ −− If you have a PPPoE connection for your DSL/Cablemodem or choose not to change the MTU to 1500, see below for a different solution. −− ]

  To fix this, first see what your MTU for your Internet link is now. To do this, run "/bin/ifconfig". Now look at the lines that corresponds to your Internet connection and look for the MTU. This NEEDs to be set to 1500. Usually, Ethernet links will default to this but serial PPP links will default to 576.

### Changing the MTU of a PPP link:

- To fix the MTU issue on your PPP link, edit your /etc/ppp/options file and towards the top, add the following text on two seperate lines: "mtu 1500" and "mru 1500". Save these new changes and then restart PPP. Like above, verify that your PPP link now has the correct MTU and MTU.

- To fix the MTU issue on a standard Ethernet link to your bridged or routed DSL, Cablemodem, etc. connection, you need to edit the correct network startup scripts for your Linux distribution. Please see the TrinityOS − Section 16 document for network optimizations.

### Old UNIX Serial interfaces:

- Lastly, though this isn't a common problem, some people have found the following to be their solution. With PPP users, verify what port is your PPPd code connecting to. Is it a /dev/cua* port or a /dev/ttyS* port? It NEEDS to be a /dev/ttyS* port. The cua style is OLD and it breaks some things in very odd ways.

### PPPoE Users:

For those users who use PPPoE (that has a maximum MTU of 1490) or for those users who choose NOT to use an MTU of 1500, not is all lost. If you reconfigure ALL of your MASQed PCs to use the SAME MTU as your external Internet link's MTU, everything should work fine. It should be noted that some PPPoE ISPs might require a MTU of 1460 for proper connectivity.

How do you do this? Follow these simple steps for your respective operating system.

The follow examples show an example of an MTU of 1490 for typical PPPoE connections for some DSL and Cablemodem users. It is recommended to use the HIGHEST values possible for all connections that are 128Kb/s and faster.

The only real reason to use smaller MTUs is to lower latency but at the cost of throughput. Please see:

http://www.ecst.csuchico.edu/~dranch/PPP/ppp−performance.html#mtu

for more details on this topic.

*** If you have had SUCCESS, FAILURE, or have procedures for OTHER operating *** systems, please email David Ranch. Thanks!

### Linux:

---

```
1. The setting of MTU can vary from Linux distribution to distribution.

   For Redhat: You need to edit the various "ifconfig" statements in
               the /sbin/ifup script

   For Slackware: You need to edit the various "ifconfig" statements in
                  the /etc/rc.d/rc1.inet
```

```
2. Here is one good, any-distribution-will-work example, edit the
   /etc/rc.d/rc.local file and put the following at the END of the file:

        echo "Changing the MTU of ETH0"
        /sbin/ifconfig eth0 mtu 1490

     Replace "eth0" with the interface name that is the machine's upstream
     connection that is connected to the Internet.

3. For advanced options like "TCP Receive Windows" and such, detailed examples
   on how to edit the respective networking scripts for your specific Linux
   distro, etc., please see Chapter 16 of
   http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html#trinityos
```

## MS Windows 95:

```
1. Making ANY changes to the Registry is inheritantly risky but
   with a backup copy, you should be safe.  Proceed at your OWN RISK.

2. Goto Start-->Run-->RegEdit

3. You should make a backup copy fo your Registry before doing anything.  To
   do this, copy the "user.dat" and "system.dat" files from the \WINDOWS
   directory and put them into a safe place.  It should be noted that the
   previously mentioned method of using "Regedit: Registry-->Export Registry
   File-->Save a copy of your registry" would only do Registry MERGES and NOT
   do a replacement.

4. Search though each of the Registry trees that end in "n" (e.g. 0007)
   that have a Registry entry called "IPAddress" that has the IP address
   of your NIC.  Under that key, add the following:

   From http://support.microsoft.com/support/kb/articles/q158/4/74.asp

     [Hkey_Local_Machine\System\CurrentControlset\Services\Class\NetTrans\000n]

         type=DWORD
         name="MaxMTU"            (Do NOT include the quotes)
         value=1490 (Decimal)     (Do NOT include the text "(Decimal)")

         type=DWORD
         name="MaxMSS"            (Do NOT include the quotes)
         value=1450 (Decimal)     (Do NOT include the text "(Decimal>")


5. You can also change the "TCP Receive Window" which sometimes
   increases network performance SUBSTANTIALLY.  If you notice your
   throughput has DECREASED, put these items BACK to their original
   settings and reboot.

     [HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP]
        type=DWORD
        name="DefaultRcvWindow"   (Do NOT include the quotes)
        value=32768 (Decimal)     (Do NOT include the text "(Decimal>")

        type=DWORD
        name="DefaultTTL"         (Do NOT include the quotes)
        value=128 (Decimal)       (Do NOT include the text "(Decimal>")
```

```
6. Reboot to make the changes take effect.
```

## MS Windows 98:

```
1. Making ANY changes to the Registry is inheritantly risky but
   with a backup copy, you should be safe.  Proceed at your OWN RISK.

2. Goto Start-->Run-->RegEdit

3. You should make a backup copy fo your Registry before doing anything.  To
   do this, copy the "user.dat" and "system.dat" files from the \WINDOWS
   directory and put them into a safe place.  It should be noted that the
   previously mentioned method of using "Regedit: Registry-->Export Registry
   File-->Save a copy of your registry" would only do Registry MERGES and NOT
   do a replacement.

4. Search though each of the Registry trees that end in "n" (e.g. 0007)
   that have a Registry entry called "IPAddress" that has the IP address
   of your NIC.  Under that key, add the following:

   From http://support.microsoft.com/support/kb/articles/q158/4/74.asp

     [Hkey_Local_Machine\System\CurrentControlset\Services\Class\NetTrans\000n]
         type=STRING
         name="MaxMTU"             (Do NOT include the quotes)
         value=1490 (Decimal)      (Do NOT include the text "(Decimal)")


5. You can also change the "TCP Receive Window" which sometimes
   increases network performance SUBSTANTIALLY.  If you notice your
   throughput has DECREASED, put these items BACK to their original
   settings and reboot.

     [HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP]

        type=STRING
        name="DefaultRcvWindow"    (Do NOT include the quotes)
        value=32768 (Decimal)      (Do NOT include the text "(Decimal>")

        type=STRING
        name="DefaultTTL"          (Do NOT include the quotes)
        value=128 (Decimal)        (Do NOT include the text "(Decimal>")

6. Reboot to make the changes take effect.
```

## MS Windows NT 4.x

```
1. Making ANY changes to the Registry is inheritantly risky but
   with a backup copy, you should be safe.  Proceed at your
```

```
   OWN RISK.

2. Goto Start-->Run-->RegEdit

3. Registry-->Export Registry File-->Save a copy of your registry
   to a reliable place

4. Create the following keys in the the Registry trees two
   possible Registry trees.  Multiple entries are for various
   network devices like DialUp Networking (ppp), Ethernet NICs,
   PPTP VPNs, etc.

   http://support.microsoft.com/support/kb/articles/Q102/9/73.asp?LN=EN-US&SD=gn&FR=0


   [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Parameters\Tcpip]
                     and
   [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<Adapter-name>\Parameters\Tcpip]

      Replace "<Adapter-Name>" with the respective name of your uplink LAN NIC
      interface

         type=DWORD
         name="MTU"              (Do NOT include the quotes)
         value=1490 (Decimal)    (Do NOT include the text "(Decimal>")

       (Do NOT include the quotes)


 *** If you know how to also change the MSS, TCP Window Size, and the
 *** TTL parameters in NT 4.x, please email dranch@trinnet.net as I
 *** would love to add it to the HOWTO.

5. Reboot to make the changes take effect.
```

## MS Windows 2000

```
1. Making ANY changes to the Registry is inheritantly risky but
   with a backup copy, you should be safe.  Proceed at your
   OWN RISK.

2. Goto Start-->Run-->RegEdit

3. Registry-->Export Registry File-->Save a copy of your registry
   to a reliable place

4. Navigate down to the key:

   [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Inter
faces\<ID for Adapter>

   Each ID Adapter has default keys for DNS, TCP/IP address, Default Gateway,
   subnet mask, etc. Find the key one that is for your network card.

5. Create the following Entry:

      type=DWORD
      name="MTU"                              (Do NOT include the quotes)
```

```
      value=1490 (Decimal)      (Do NOT include the text "(Decimal)")

http://support.microsoft.com/support/kb/articles/Q120/6/42.asp?LN=EN-US&SD=gn&FR=0


 *** If you know how to also change the MSS, TCP Window Size, and the
 *** TTL parameters in NT 2000, please email dranch@trinnet.net as I
 *** would love to add it to the HOWTO.

5. Reboot to make the changes take effect.
```

As stated above, if you know how to make similar changes like these to other OSes like OS/2, MacOS, etc. please email David Ranch so it can be included in the HOWTO.

## 7.16 MASQed FTP clients don't work.

Check to see that the "ip_masq_ftp" module is loaded. To do this, log into the MASQ server and run the command "/sbin/lsmod". If you don't see the "ip_masq_ftp" module loaded, make sure that you followed the BASIC /etc/rc.d/rc.firewall recommendations found in firewall−examples section. If you are implimenting your own ruleset, make sure you at include most of the examples from the HOWTO or you will have lots of continuing problems.

## 7.17 IP Masquerading seems slow

There might be a few reasons for this:

- You might be expecting more out of your modem line than is realistic. Lets do the math for a typical 56k modem connection:
  1. 56k modems = 56,000 bits per second.
  2. You really DON'T have a 56k modem but a 52k modem per US FCC limitations.
  3. You'll almost NEVER get 52k, the best connection I used to get was 48k
  4. 48,000 bits per second is 4,800 BYTES per second (8 bits to a byte + 2 bits for the START and STOP RS−232 serial bits)
  5. With an MTU of 1500, you will get (3.2) packets in one second. Since this will involve fragmentation, you need to round DOWN to (3) packets per second.
  6. Again with MTU of 1500, thats 3.2 x 40 bytes of TCP/IP overhead (8%)
  7. So the BEST throughput you could hope for is 4.68KB/s w/o compression. Compression, be it v.42bis hardware compression, MNP5, or MS/Stac compression can yeild impressive numbers on highly compressable stuff like TEXT files but acutally slow things down when transfering pre−compressed files like ZIPs, MP3s, etc.

- Ethernet attached setups (DSL, Cablemodem, LANs, etc)

  - Make sure you don't have both your INTERNAL and EXTERNAL networks running on the same network card with the "IP Alias" feature. If you *ARE* doing this, it can be made to work but it will be excessively slow due to high levels of collisions, IRQ usage, etc. It is highly recommended to get another network card so that the internal and external networks have their own interface.

    Make sure you have the right Ethernet settings for both SPEED and DUPLEX.

◊ Some 10Mb/s Ethernet cards and most 100Mb/s cards support FULL Duplex connections. Direct connections from Ethernet card to, say, DSL modem (without any hubs in between) *CAN* be set to FULL DUPLEX but only if the DSL modem supports it. You should also be sure that you have Ethernet cables with all eight wires used and they are good quality.
◊ Internal networks that use HUBs −cannot− use Full Duplex. You need either a 10 or 100Mb.s Ethernet *SWITCH* to be able to do this.
◊ Both auto 10/100Mb/s SPEED negotiation and Full/Half DUPLEX negotiation on Ethernet cards can wreck havoc on networks. I recommend to hard code both the NIC speed and duplex into the NIC(s) if possible. This is directly possible via Linux NIC kernel modules but isn't directly possible in monolithic kernels. You will need to either use MII utilities from Donald Becker's NIC drivers and utils FAQ−Hardware or hardcode the kernel source.

• Optimize your MTU and set the TCP Sliding window to at least 8192

♦ Though this is COMPLETELY out of the scope of this document, this helps QUITE A BIT on ANY network link you have be it an internal or external PPP, Ethernet, TokenRing, etc. link. For more details, this topic is briefly touched on above in the MTU−issues section. For even more details, check out the Network Optimization section of TrinityOS − Section 16.

• Serial based modem users with PPP

♦ If you have an external modem, make sure you have a good serial cable. Also, many PCs have cheesy ribbon cables connecting the serial port from the motherboard or I/O card to the serial port connection. If you have one of these, make sure it is in good condition. Personally, I have ferrite coils (those grey−black metal like rings) around ALL of my ribbon cables.

♦ Make sure your MTU is set to 1500 as described in the FAQ section of this HOWTO above

♦ Make sure that your serial port is a 16550A or better UART. Run "dmesg | more" to verify

♦ Setup IRQ−Tune for your serial ports

◊ On most PC hardware, the use of Craig Estey's IRQTUNE tool and significantly increase serial port performance including SLIP and PPP connections.

♦ Make sure that your serial port for your PPP connection is running at 115200 (or faster if both your modem and serial port can handle it.. a.k.a ISDN terminal adapters)

◊ 2.0.x kernels: The 2.0.x kernels are kind of an odd ball because you can't directly tell the kernel to clock the serial ports at 115200. So, in one of your startup scripts like the /etc/rc.d/rc.local or /etc/rc.d/rc.serial file, execute the following commands for a modem on COM2:

· setserial /dev/ttyS1 spd_vhi

· In your PPPd script, edit the actual pppd execution line to include the speed "38400" per the pppd man page.

◊ 2.2.x kernels: Unlike the 2.0.x kernels, both the 2.1.x and 2.2.x kernels don't have this "spd_vhi" issue.

- So, in your PPPd script, edit the actual pppd execution line to include the speed "115200" per the pppd man page.

- All interface types:

## 7.18 IP Masquerading with PORTFWing seems to break when my line is idle for long periods

If you have a DSL or Cablemode, this behavior is unfortunately quite common. Basically what is happening is your ISP is putting your connection into a very low priority queue to better service non−idle connections. The problem is that some enduser's connections will actually be taken OFFLINE until some traffic from the user's DSL/Cablemodem connection wakens the ISP's hardware

- 
- Some DSL installations can take an idle connection OFFLINE and only be checked for activity once every 30 seconds or so.
- Some Cablemodem setups can set an idle connection into a low priority queue and only be checked for activity every minute or so.

What do I recommend to do? Ping your default gateway every 30 seconds. To do this, edit the /etc/rc.d/rc.local file and add the following to the bottom of the file:

```
        ping -i 30 100.200.212.121 > /dev/null &
```

Replace the 100.200.212.121 with your default router (upstream router).

## 7.19 Now that I have IP Masquerading up, I'm getting all sorts of weird notices and errors in the SYSLOG log files. How do I read the IPFWADM/IPCHAINS firewall errors?

There is probably two common things that you are going to see:

- **MASQ: Failed TCP Checksum error:** You will see this error when a packet coming from the Internet gets corrupt in the data section of the packet but the rest of it "seems" ok. When the Linux box receives this packet, it will calculate the CRC of the packet and determine that its corrupt. On most machines running OSes like Microsoft Windows, they just silently drop the packets but Linux IP MASQ reports it. If you get a LOT of them over your PPP link, first follow the FAQ entry above for "Masq is slow".

- If all of those tips don't help, try adding the line "−vj" to your /etc/ppp/options file and restart PPPd.

- **Firewall hits**: Being on the Internet with a decent firewall, you are going to be surprised how many people are going to try to get into your Linux box! So what do all these firewall logs mean?

From the TrinityOS − Section 10 doc:

```
        In the below rulesets, any lines that either DENY or REJECT any
        traffic also have a "−o" to LOG this firewall hit to the SYSLOG
        messages file found either in:

                Redhat:         /var/log
                Slackware:      /var/adm

        If you look at one of these firewall logs, do would see something like:

        −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
        IPFWADM:
        Feb 23 07:37:01 Roadrunner kernel: IP fw−in rej eth0 TCP 12.75.147.174:1633
           100.200.0.212:23 L=44 S=0x00 I=54054 F=0x0040 T=254

        IPCHAINS:
        Packet log: input DENY eth0 PROTO=17 12.75.147.174:1633 100.200.0.212:23
          L=44 S=0x00 I=54054 F=0x0040 T=254
        −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

  There is a LOT of information in this just one line.  Lets break out this example
  so refer back to the original firewall hit as you read this.  Please note that this
  example is for IPFWADM though it is DIRECTLY readable for IPCHAINS users.

        −−−−−−−−−−−−−−

        − This firewall "hit" occurred on "Feb 23 07:37:01"

        − This hit was on the "RoadRunner" computer.

        − This hit occurred on the "IP" or TCP/IP protocol

        − This hit came IN to ("fw−in") the firewall
                * Other logs can say "fw−out" for OUT or "fw−fwd" for FORWARD

        − This hit was then "rejECTED".
                * Other logs can say "deny" or "accept"

        − This firewall hit was on the "eth0" interface (Internet link)

        − This hit was a "TCP" packet

        − This hit came from IP address "12.75.147.174" on return port "1633".

        − This hit was addressed to "100.200.0.212" on port "23" or TELNET.
                * If you don't know that port 23 is for TELNET, look at your
                         /etc/services file to see what other ports are used for.

        − This packet was "44" bytes long

        − This packet did NOT have any "Type of Service" (TOS) set
                −−Don't worry if you don't understand this.. not required to know
                * divide this by 4 to get the Type of Service for ipchains users

        − This packet had the "IP ID" number of "18"
                −−Don't worry if you don't understand this.. not required to know

        − This packet had a 16bit fragment offset including any TCP/IP packet
          flags of "0x0000"
                −−Don't worry if you don't understand this.. not required to know
                * A value that started with "0x2..." or "0x3..." means the "More
```

```
                Fragments" bit was set so more fragmented packet will be coming in
                to complete this one BIG packet.
              * A value which started with "0x4..." or "0x5..." means that the
                "Don't Fragment" bit is set.
              * Any other values is the Fragment offset (divided by 8) to be later
                used to recombine into the original LARGE packet

      - This packet had a TimeToLive (TTL) of 20.
              * Every hop over the Internet will subtract (1) from this number.  Usually,
                packets will start with a number of (255) and if that number ever reaches
                (0), it means that realistically the packet was lost and will be deleted.
```

## 7.20 Can I configure IP MASQ to allow Internet users to directly contact internal MASQed servers?

Yes! With IPPORTFW, you can allow ALL or only a select few Internet hosts to contact ANY of your internal MASQed computers. **This topic is completely covered in the Forwarders section of this HOWTO.**

## 7.21 I'm getting "kernel: ip_masq_new(proto=UDP): no free ports." in my SYSLOG files. Whats up?

One of your internal MASQed machine is creating an abnormally high number of packets destined for the Internet. As the IP Masq server builds the MASQ table and forwards these packets out over the Internet, the table is quickly filling. Once the table is full, it will give you this error.

The only application that I known that temporarily creates this situation is a gaming program called "GameSpy". Why? Gamespy builds a server list and then pings all of the servers in the list (1000s of game servers). By creating all these pings, it creates 10,000s of quick connections in a VERY short time. Until these sessions timeout via the IP MASQ timeouts, the MASQ tables become "FULL".

So what can you do about it? Realistically, don't use programs that do things like this. If you do get this error in your logs, find it and stop using it. If you really like GameSpy, just don't do a lot of server refreshes. Regardless, once you stop running this MASQ'ed program, this MASQ error will go away as these connections timeout in the MASQ tables.

## 7.22 I'm getting "ipfwadm: setsockopt failed: Protocol not available" when I try to use IPPORTFW!

If you get the error message "ipfwadm: setsockopt failed: Protocol not available", you AREN'T running your new kernel. Make sure that you moved the new kernel over, re−run LILO, and then reboot again.

Please see the end of the Forwarders section for full details.

this Microsoft KnowledgeBase article.

The first work−around is to configure IPPORTFW from the Forwarders section and portfw TCP ports 137, 138, and 139 to the internal Windows machine's IP address. Though this solution works, it will only works for ONE internal machine.

The second solution is to install and configure Samba on the Linux MASQ server. With Samba running, you can then map your internal Windows File and Print shares onto the Samba server. Then, you can mount these newly mounted SMB shares to all of your external clients. Configuring Samba is fully covered in a HOWTO found in a Linux Documentation Project and in the TrinityOS document as well.

The third solution is to configure a VPN (virtual private network) between the two Windows machines or between the two networks. This can either be done via the PPTP or IPSEC VPN solutions. There is a PPTP patch for Linux and also a full IPSEC implimentation available for both 2.0.x and 2.2.x kernels. This solution will probably be the most reliable and secure method of all three solutions.

All of these solutions are NOT covered by this HOWTO. I recommend that you look at the TrinityOS documentation for IPSEC help and JJohn Hardin's PPTP page for more information.

*Also PLEASE understand that Microsoft's SMB protocol is VERY insecure. Because of this, running either Microsoft File and Print sharing or Windows Domain login traffic over the Internet without any encryption is a VERY BAD idea.*

## 7.24 ( IDENT ) − IRC won't work properly for MASQed IRC users. Why?

The main possible reason is because most common Linux distribution's IDENT or "Identity" servers can't deal with IP Masqueraded links. No worries though, there are IDENTs out there that will work.

Installing this software is beyond the scope of this HOWTO but each tool has its own documentation. Here are some of the URLs:

- Oident is a favorite IDENT server for MASQ users.
- Mident is another popular IDENT server.
- Sident
- Other Idents

Please note that some Internet IRCs servers still won't allow multiple connections from the same host even if they get Ident info and the users are different though. Complain to the remote sys admin. :)

## 7.25 ( DCC ) − mIRC doesn't work with DCC Sends

This is a configuration problem on your copy of mIRC. To fix this, first disconnect mIRC from the IRC server. Now in mIRC, go to File −−> Setup and click on the "IRC servers tab". Make sure that it is set to port 6667. If you require other ports, see below. Next, goto File −−> Setup −−> Local Info and clear the fields for Local Host and IP Address. Now select the checkboxes for "LOCAL HOST" and "IP address" (IP address may be checked but disabled). Next under "Lookup Method", configure it for "normal". It will NOT work if "server" is selected. That's it. Try to the IRC server again.

If you require IRC server ports other than 6667, (for example, 6969) you need to edit the /etc/rc.d/rc.firewall startup file where you load the IRC MASQ modules. Edit this file and the line for "modprobe ip_masq_irc" and add to this line "ports=6667,6969". You can add additional ports as long as they are separated with commas.

Finally, close down any IRC clients on any MASQed machines and re−load the IRC MASQ module:

/sbin/rmmod ip_masq_irc /etc/rc.d/rc.firewall

## 7.26 ( IP Aliasing ) − Can IP Masquerade work with only ONE Ethernet network card?

**Yes and no**. With the "IP Alias" kernel feature, users can setup multiple aliased interfaces such as eth0:1, eth0:2, etc but its is NOT recommended to use aliased interfaces for IP Masquerading. Why? Providing a secure firewall becomes very difficult with a single NIC card. In addition to this, you will experience an abnormal amount of errors on this link since incoming packets will almost simultaneously be sent out at the same time. Because of all this and NIC cards now cost less than $10, I highly recommend to just get a NIC card for each MASQed network segment.

Users should also understand that IP Masquerading will only work out a physical interface such as eth0, eth1, etc. MASQing out an aliased interface such as "eth0:1, eth1:1, etc" will NOT work. In other words, the following WILL NOT WORK:

- /sbin/ipfwadm −F −a m −W eth0:1 −S 192.168.0.0/24 −D 0.0.0.0/0
- /sbin/ipchains −A forward −i eth0:1 −s 192.168.0.0/24 −j MASQ"

If you are still interested in using aliased interfaces, you need to enable the "IP Alias" feature in the kernel. You will then need to re−compile and reboot. Once running the new kernel, you need to configure Linux to use the new interface (i.e. /dev/eth0:1, etc.). After that, you can treat it as a normal Ethernet interface with some restrictions like the one above.

## 7.27 ( MULTI−LAN ) − I have two MASQed LANs but they cannot communicate with eachother!

Please see the multiple−masqed−lans section for full details.

## 7.28 ( SHAPING ) − I want to be able to limit the speed of specific types of traffic

This topic really doesn't have anything to do with IPMASQ and everthing to do with Linux's built−in traffic shaping and rate−limiting. Please see the /usr/src/linux/Documentation/networking/shaper.txt file from your local kernel sources for more details.

You will also find more information about this including several URLs under the 2.2.x−Requirements section for IPROUTE2.

## 7.29 ( ACCOUNTING ) − I need to do accounting on who is using the network

Though this doesn't have much to do with IPMASQ, here are a few ideas. If you kow of any better solutions, please email the author of this HOWTO so they can be added to the HOWTO.

- Idea #1: Say you want to log ALL WWW traffic going out to the Internet. You can setup a firewall rule to ACCEPT PORT 80 traffic with with the SYN bit set and LOG it. Now mind you, this can create VERY large log files.
- Idea #2: You could run the command "ipchains −L −M" once a second and log all of those entries. You then could write a program to merge this information into one large file.

## 7.30 ( MULTIPLE IPs ) − I have several EXTERNAL IP addresses that I want to PORTFW to several internal machines. How do I do this?

You DON'T. MASQ is a 1:Many NAT setup which not the correct tool to do what you are looking for. You are looking for a Many:Many NAT solution which is traditional NAT setup. Give a look at the shaping FAQ entry below for more details on the IPROUTE2 tool that will do what you need.

For people out there who are considering enabling multiple IP addresses on one internal NIC using "IP Alias" and then PORTFWed ALL of those ports (0−65535) and used IPROUTE2 to maintain the proper source/destination IP pairs, this has been done SUCCESSFULLY on 2.0.x kernels and less successfully on 2.2.x kernels. Regardless of success, it isn't the proper way to do it and is not a supported MASQ configuration. Please, give IPROUTE2 a look.. its the right way to do true NAT.

One thing to also note:

If you have a bridged DSL or Cablemodem connection (not PPPoE), things are a little more difficult because your setup isn't routed. No worries though, check out the "Bridge+Firewall, Linux Bridge+Firewall Mini−HOWTO" on the LDP. It will teach you how to get your Linux box to support multiple IP addresses on a single interface!

## 7.31 I'm trying to use the NETSTAT command to show my Masqueraded connections but its not working

There might be a problem with the "netstat" program in 2.0.x−based Linux distros. After a Linux reboot, running "netstat −M" works fine but after a MASQed computer runs some successful ICMP traffic like ping, traceroute, etc., you might see something like:

```
masq_info.c: Internal Error `ip_masquerade unknown type'.
```

The workaround for this is to use the "/sbin/ipfwadm −M −l" command. You will also notice that once the listed ICMP masquerade entries timeout, "netstat" works again.

## 7.32 ( VPNs ) − I would like to get Microsoft PPTP (GRE tunnels) and/or IPSEC (Linux SWAN) tunnels running through IP MASQ

This IS possible. Though it is somewhat out of the scope of this document, check out John Hardin's PPTP Masq page for all the details.

## 7.33 I want to get the XYZ network game to work through IP MASQ but it won't work. Help!

First, check Steve Grevemeyer's MASQ Applications page. If your solution isn't listed there, try patching your Linux kernel with Glenn Lamb's LooseUDP patch which is covered in the LooseUDP section above. Also check out Dan Kegel's NAT Page for more information.

If you are technically inclined, use the program "tcpdump" and sniff your network. Try to find out what protocols and port numbers your XYZ game is using. With this information in hand, subscribe to the IP Masq email list and email your results for help.

## 7.34 IP MASQ works fine for a while but then it stops working. A reboot seems to fix this for a while. Why?

I bet you are using IPAUTOFW and/or you have it compiled into the kernel huh?? This is a known problem with IPAUTOFW. It is recommend to NOT even configure IPAUTOFW into the Linux kernel and use IPPORTFW option instead. This is all covered in more detail in the Forwarders section.

## 7.35 Internal MASQed computers cannot send SMTP or POP−3 mail!

Though this isn't a Masquerading issue per se but many people do this so it should be mentioned.

SMTP: The issue is that you are probably using your Linux box as a SMTP relay server and get the following error:

```
"error from mail server: we do not relay"
```

Newer versions of Sendmail and other Mail Transfer Agents (MTAs) disable relaying by default (this is a good thing). So do the following to fix this:

- Sendmail: Enable specific relaying for your internal MASQed machines by editing the /etc/sendmail.cw file and add the hostname and domain name of your internal MASQed machine. You should also check to see that the /etc/hosts file has the IP address and Fully Qualified Domain Name (FQDN) configured in it. Once this is done, you need to restart Sendmail for it to re−read its configuration files. This is covered in TrinityOS − Section 25

POP−3: Some users configure their internal MASQ'ed computer's POP−3 clients to connect to some external SMTP server. While this is fine, many SMTP servers out there will try to IDENT your connection on port

113. Most likely your problem stems around your default Masquerade policy being set to DENY. This is BAD. Set it to REJECT and re−run your rc.firewall ruleset.

## 7.36 ( IPROUTE2 ) – I need different internal MASQed networks to exit on different external IP addresses

Say you have the following setup: You have multiple internal networks and also multiple external IP addresses and/or networks. What you want to do is have LAN #1 to only use External IP #1 but you wan LAN #2 to use External IP #2.

Internal LAN −−−−−−−−−−> official IP

LAN #1 External IP #1 192.168.1.x −−> 123.123.123.11

LAN #2 External IP #2 192.168.2.x −−> 123.123.123.12

Basically, what we have described here is routing NOT only on the destination address (typical IP routing) but also routing based upon the SOURCE address as well. This is typically called "policy−based routing" or "source routing". This functionality is NOT available in 2.0.x kernels, it *IS* available for 2.2.x kernels via the IPROUTE2 package, and it is not built into the new 2.4.x kernels using IPTABLES.

First, you have to understand that both IPFWADM and IPCHAINS get involved *AFTER* the routing system has decided where to send a given packet. This statement really ought to be stamped in big red letters on all IPFWADM/IPCHAINS/IPMASQ documentation. The reason for this is that users MUST get their routing setup right first and then start adding IPFWADM/IPCHAINS and/or Masq features.

Anyway, for the example case shown above, you need to persuade the routing system to direct packets from 192.168.1.x via 123.123.1233.11 and packets from 192.168.2.x via 123.123.123.12. That is the hard part and adding Masq on top of correct routing is easy.

To do this fancy routing, you will use IPROUTE2. Because this functionality has NOTHING to do with IPMASQ, this HOWTO does not cover this topic in great detail. Please see 2.2.x−Requirements for complete URLs and documentation for this topic.

The "iprule" and "iproute" commands are the same as "ip rule" and "ip route" commands (I prefer the former since it is easier to search for.) All the commands below are completely untested, if they do not work, please contact the author of IPROUTE2.. not David Ranch or anyone on the Masq email list as it has NOTHING to do with IP Masquerading.

The first few commands only need to be done once at boot, say in /etc/rc.d/rc.local file.

```
# Allow internal LANs to route to each other, no masq.
  /sbin/iprule add from 192.168.0.0/16 to 192.168.0.0/16 table main pref 100
# All other traffic from 192.168.1.x is external, handle by table 101
  /sbin/iprule add from 192.168.1.0/24 to 0/0 table 101 pref 102
# All other traffic from 192.168.2.x is external, handle by table 102
  /sbin/iprule add from 192.168.2.0/24 to 0/0 table 102 pref 102

These commands need to be issued when eth0 is configured, perhaps in
```

```
/etc/sysconfig/network-scripts/ifup-post (for Redhat systems).  Be sure to
do them by hand first to make sure they work.

# Table 101 forces all assigned packets out via 123.123.123.11
  /sbin/iproute add table 101 via 62123.123.123.11
# Table 102 forces all assigned packets out via 123.123.123.12
  /sbin/iproute add table 102 via 62123.123.123.12

At this stage, you should find that packets from 192.168.1.x to the
outside world are being routed via 123.123.123.11, packets from
192.168.2.x are routed via 123.123.123.12.

Once routing is correct, now you can add any IPFWADM or IPCHAINS rules.
The following examples are for IPCHAINS:


/sbin/ipchains -A forward -i ppp+ -j MASQ

If everything hangs together, the masq code will see packets being
routed out on 123.123.123.11 and 123.123.123.12 and will use those addresses
as the masq source address.
```

## 7.37 Why do the new 2.1.x and 2.2.x kernels use IPCHAINS instead of IPFWADM?

IPCHAINS supports the following features that IPFWADM doesn't:

- "Quality of Service" (QoS support)
- A TREE style chains system vs. LINEAR system like IPFWADM (Eg. this allows something like "if it is ppp0, jump to this chain (which contains its own difference set of rules)"
- IPCHAINS is more flexible with configuration. For example, it has the "replace" command (in addition to "insert" and "add"). You can also negate rules (e.g. "discard any outbound packets that don't come from my registered IP" so that you aren't the source of spoofed attacks).
- IPCHAINS can filter any IP protocol explicitly, not just TCP, UDP, ICMP

## 7.38 I've just upgraded to the 2.2.x kernels, why isn't IP Masquerade working?

There are several things you should check assuming your Linux IP Masq box already have proper connection to the Internet and your LAN:

- Make sure you have the necessary features and modules are compiled and loaded. See earlier sections for detail.
- Check `/usr/src/linux/Documentation/Changes` and make sure you have the minimal requirement for the network tools installed.
- Make sure you followed all the tests in the Testing section of the HOWTO.

- You should use ipchains to manipulate IP Masq and firewalling rules.
- The standard IPAUTOFW and IPPORTFW port forwarders have been replaced by IPMASQADM. You'll need to apply these patches to the kernel, re−compile the kernel, compile the new IPMASQADM tool and then convert your old IPAUTOFW/IPPORTFW firewall rulesets to the new syntax. This is completely covered in the Forwarders section.
- Go through all setup and configuration again! A lot of time it's just a typo or a simple mistake you are overlooking.

## 7.39 I've just upgraded to a 2.0.38+ kernels later, why isn't IP Masquerade working?

There are several things you should check assuming your Linux IP Masq box already have proper connection to the Internet and your LAN:

- Make sure you have the necessary features and modules are compiled and loaded. See earlier sections for detail.
- Check `/usr/src/linux/Documentation/Changes` and make sure you have the minimal requirement for the network tools installed.
- Make sure you followed all the tests in the Testing section of the HOWTO.
- You should use ipfwadm to manipulate IP Masq and firewalling rules. If you want to use IPCHAINS, you'll need to apply a patch the 2.0.x kernels.
- Go through all setup and configuration again! A lot of time it's just a typo or a simple mistake you overlooked.

## 7.40 I need help with EQL connections and IP Masq

EQL has nothing to do with IP Masq though they are commonly teamed up on Linux boxes. Because of this, I recommend to check out the NEW version of Robert Novak's EQL HOWTO for all your EQL needs.

## 7.41 I can't get IP Masquerade to work! What options do I have for Windows Platforms?

Giving up a free, reliable, high performance solution that works on minimal hardware and pay a fortune for something that needs more hardware, lower performance and less reliable? (IMHO. And yes, I have real life experience with these ;−)

Okay, it's your call. If you want a Windows NAT and/or proxy solution, here is a decent listing. I have no preference of these tools since I haven't used them before.

- Firesock (from the makers of Trumpet Winsock)
  - Does Proxy
  - http://www.trumpet.com.au

- Iproute
    - DOS program designed to run on 286+ class computers
    - requires another box like Linux MASQ
    - http://www.mischler.com/iproute/
- Microsoft Proxy
    - Requires Windows NT Server
    - Quite expensive
    - http://www.microsoft.com
- NAT32
    - Windows 95/98/NT compatible
    - http://www.nat32.com
    - Roughly $25 for Win9x and $47 for Win9x and WinNT
- SyGate
    - http://www.sygate.com
- Wingate
    - Does proxy
    - Costs roughly $30 for 2−3 IPs
    - http://www.wingate.com
- Winroute
    - Does NAT
    - http://www.winroute.cz/en/

Lastly, do a web search on "MS Proxy Server", "Wingate", "WinProxy", or goto www.winfiles.com. And definitely DON'T tell anyone that we sent you.

## 7.42 I want to help on IP Masquerade development. What can I do?

Join the Linux IP Masquerading DEVELOPERS list and ask the developers there what you can help with. For more details on joining the lists, check out the Masq−List FAQ section.

Please DON'T ask NON−IP−Masquerade development related questions there!!!!

## 7.43 Where can I find more information on IP Masquerade?

You can find more information on IP Masquerade at the Linux IP Masquerade Resource that David Ranch maintains.

You can also find more information at Dranch's Linux page where the TrinityOS and other Linux documents are kept.

You may also find more information at The Semi−Original Linux IP Masquerading Web Site maintained by Indyramp Consulting, who also provides the IP Masq mailing lists.

Lastly, you can look for specific questions in the IP MASQ and IP MASQ DEV email archives or ask a specific question on these lists. Check out the Masq−List FAQ item for more details.

## 7.44 I want to translate this HOWTO to another language, what should I do?

Make sure the language you want to translate to is not already covered by someone else. But, most of the translated HOWTOs are now OLD and need to be updated. A list of available HOWTO translations are available at the Linux IP Masquerade Resource.

If a copy of a **current** IP MASQ HOWTO isn't in your proposed language, please download the newest copy of the IP−MASQ HOWTO SGML code from the Linux IP Masquerade Resource. From there, begin your work while maintaining good SGML coding. For more help on SGML, check out www.sgmltools.org

## 7.45 This HOWTO seems out of date, are you still maintaining it? Can you include more information on ...? Are there any plans for making this better?

Yes, this HOWTO is still being maintained. In the past, we've been guilty of being too busy working on two jobs and don't have much time to work on this, my apology. As of v1.50, David Ranch has begun to revamp the document and get it current again.

If you think of a topic that could be included in the HOWTO, please send email to ambrose@writeme.com and dranch@trinnet.net. It will be even better if you can provide that information. We will then include the information into the HOWTO once it is both found appropriate and tested. Many thanks for your contributions!

We have a lot of new ideas and plans for improving the HOWTO, such as case studies that will cover different network setup involving IP Masquerade, more on security via strong IPFWADM/IPCHAINS firewall rulesets, IPCHAINS usage, more FAQ entries, etc. If you think you can help, please do! Thanks.

## 7.46 I got IP Masquerade working, it's great! I want to thank you guys, what can I do?

- Can you translate the newer version of the HOWTO to another language?
- Thank the developers and appreciate the time and effort they spent on this.
- Join the IP Masquerade email list and support new MASQ users
- Send an email to us and let us know how happy you are
- Introduce other people to Linux and help them when they have problems.

---

# 8. Miscellaneous

## 8.1 Useful Resources

- IP Masquerade Resource page Will have all the current information for setting up IP Masquerade on 2.0.x, 2.2.x, and even old 1.2 kernels!
- Juan Jose Ciarlante's WWW site who is the current Linux IP Masquerade maintainer.
- IP Masquerade mailing list Archives contains the recent messages sent to the mailing lists.
- David Ranch's Linux page including the TrinityOS Linux document and current versions of the IP−MASQ−HOWTO.. Topics such as IP MASQ, strong IPFWADM/IPCHAINS rulesets, PPP, Diald, Cablemodems, DNS, Sendmail, Samba, NFS, Security, etc. are covered.
- The IP Masquerading Applications page: A comprehensive list of applications that work or can be tuned to work through a Linux IP masquerading server.
- For people setting up IP Masq on MkLinux, email Taro Fukunaga at tarozax@earthlink.net for a copy of his short MkLinux version of this HOWTO.
- IP masquerade FAQ has some general information
- Paul Russel's http://netfilter.filewatcher.org/ipchains/ doc and its possibly older backup at Linux IPCHAINS HOWTO. This HOWTO has lots of information for IPCHAINS usage, as well as source and binaries for the ipchains tool.
- X/OS Ipfwadm page contains sources, binaries, documentation, and other information about the `ipfwadm` package
- Check out the GreatCircle's Firewall mailing list for a great resource for strong firewall rulesets.
- The LDP Network Administrator's Guide is a MUST for the beginner Linux administrator trying to set up a network.
- The Linux NET−3−4 HOWTO is also another comprehensive document on how to setup and configure Linux networking.
- Linux ISP Hookup HOWTO and Linux PPP HOWTO gives you information on how to connect your Linux host to the Internet
- Linux Ethernet−Howto is a good source of information about setting up a LAN running over Ethernet.
- Donald Becker's NIC drivers and Support Utils
- You may also be interested in Linux Firewalling and Proxy Server HOWTO
- Linux Kernel HOWTO will guide you through the kernel compilation process
- Other Linux HOWTOs such as Kernel HOWTO

- Posting to the USENET newsgroup: comp.os.linux.networking

## 8.2 Linux IP Masquerade Resource

The Linux IP Masquerade Resource is a website dedicated to Linux IP Masquerade information also maintained by Ambrose Au. It has the latest information related to IP Masquerade and may have information that is not being included in the HOWTO.

You may find the Linux IP Masquerade Resource at the following locations:

- http://ipmasq.cjb.net/, Primary Site, redirected to http://ipmasq.cjb.net/

- http://ipmasq2.cjb.net/, Secondary Site, redirected to
  http://www.geocities.com/SiliconValley/Heights/2288/

## 8.3 Thanks to the following people..

In Alphabetical order:

- Gabriel Beitler, gabrielb@voicenet.com
  on providing section 3.3.8 (setting up Novell)

- Juan Jose Ciarlante, irriga@impsat1.com.ar
  on contributing his work on his IPMASQADM port forward tool, his work on the 2.1.x and 2.2.x kernel code, the original LooseUDP patch, etc.

- Steven Clarke, steven@monmouth.demon.co.uk
  on contributing his IPPORTFW IP port forwarder tool

- Andrew Deryabin, djsf@usa.net
  on contributing his ICQ MASQ module

- Ed Doolittle, dolittle@math.toronto.edu
  on suggestion to `-V` option in `ipfwadm` command for improved security

- Matthew Driver, mdriver@cfmeu.asn.au
  on helping extensively on this HOWTO, and providing section 3.3.1 (setting up Windows 95)

- Ken Eves, ken@eves.com
  on the FAQ that provides invaluable information for this HOWTO

- John Hardin, jhardin@wolfenet.com
  for his PPTP and IPSEC forwarding tools

- Glenn Lamb, mumford@netcom.com
  for the LooseUDP patch

- Ed. Lott, edlott@neosoft.com
  for a long list of tested system and software

- Nigel Metheringham, Nigel.Metheringham@theplanet.net
  on contributing his version of IP Packet Filtering and IP Masquerading HOWTO, which make this HOWTO a better and technical in−depth document
  section 4.1, 4.2, and others
- Keith Owens, kaos@ocs.com.au
  on providing an excellent guide on ipfwadm section 4.2
  on correction to `ipfwadm -deny` option which avoids a security hole, and clarified the status of `ping` over IP Masquerade
- Michael Owings, mikey@swampgas.com
  on providing section for CU−SeeMe and Linux IP−Masquerade Teeny How−To
- Rob Pelkey, rpelkey@abacus.bates.edu
  on providing section 3.3.6 and 3.3.7 (setting up MacTCP and Open Transport)
- Harish Pillay, h.pillay@ieee.org
  on providing section 4.5 (dial−on−demand using Diald)
- Mark Purcell, purcell@rmcs.cranfield.ac.uk
  on providing section 4.6 (IPautofw)
- David Ranch, dranch@trinnet.net
  help updating and maintaining this HOWTO and the Linux IP Masquerade Resource Page, the TrinityOS document , ..., too many to list here :−)
- Paul Russell, rusty@linuxcare.com.au
  for all his work on IP CHAINS, IP Masquerade kernel patches, etc
- Ueli Rutishauser, rutish@ibm.net
  on providing section 3.3.9 (setting up OS/2 Warp)
- Steve Grevemeyer, grevemes@tsmservices.com
  for taking over the IP Masq Applications page from Lee Nevo and updating it to a full DB backend.
- Fred Viles, fv@episupport.com
  for his patches for proper port forarding of FTP.
- John B. (Brent) Williams, forerunner@mercury.net
  on providing section 3.3.7 (setting up Open Transport)
- Enrique Pessoa Xavier, enrique@labma.ufrj.br
  on the BOOTp setup suggestion
- All the people on the IP−MASQ email list, masq@tiffany.indyramp.com
  for their help and support for all the new Linux MASQ users.
- Other code and documentation developers of IP Masquerade for this great feature
  - ◊ Delian Delchev, delian@wfpa.acad.bg
  - ◊ David DeSimone (FuzzyFox), fox@dallas.net

- ◊ Jeanette Pauline Middelink, middelin@polyware.iaf.nl
- ◊ Miquel van Smoorenburg, miquels@q.cistron.nl
- ◊ Jos Vos, jos@xos.nl
- ◊ And more who I may have failed to mention here (please let me know)

- All users sending feedback and suggestion to the mailing list, especially the ones who reported errors in the document and the clients that are supported and not supported

- We apologize if we have omitted any important names, not included information that some fellow users have sent us yet, etc. There are many suggestions and ideas sent but there isn't have enough time to verify and integrate these changes. David Ranch is constantly trying his best to incorporate all the information sent to me into the HOWTO. I thank you for the effort, and I hope you understand our situation.

## 8.4 Reference

- Original IP masquerade FAQ by Ken Eves
- IP masquerade mailing list archive by Indyramp Consulting
- IP Masquerade WWW site by Ambrose Au
- Ipfwadm page by X/OS
- Various networking related Linux HOWTOs
- Some topics covered in TrinityOS by David Ranch

## 8.5 Changes

- TO do − HOWTO:
    - Add the scripted IPMASQADM example to the Forwarders section. Also confirm the syntax.
    - Add a little section on having multiple subnets behind a MASQ server
    - Confirm the IPCHAINS ruleset and make sure it is consistant with the IPFWADM ruleset

  TO DO − WWW page:

    - Update all PPTP urls from lowrent to ftp://ftp.rubyriver.com/pub/jhardin/masquerade/ip_masq_vpn.html
    - Update the PPTP patch on the masq site
    - Update the portfw FTP patch

  Changes from 1.90 to 1.95 − 11/11/00

    - Added a quick upfront notice in the intro that running a SINGLE NIC in MASQ mutliple ethernet segments is NOT recommended and linked to the relivant FAQ entry. Thanks to Daniel Chudnov for helping the HOWTO be more clear.
    - Added a pointer in the Intro section to the FAQ section for users looking for how MASQ is different from NAT and Proxy services.

    - Reordered the Kernel requirements sections to be 2.2.x, 2.4.x, 2.0.x

    - Expanded the kernel testing in Section 3 to see if a given kernel already supports MASQ or not.

- Reversed the order of the displayed simple MASQ ruleset examples (2.2.x and 2.0.x)
- Cleaned up some formatting issues in the 2.0.x and 2.2.x rc.firewall files
- Noted in the 2.2.x rc.firewall that the defrag option is gone in some distro's proc (Debian, TurboLinux, etc)
- Added a NOTE #3 to the rc.firewall scripts to include instructions for Pump. Thanks to Ross Johnson for this one.
- Cleaned up the simple MASQ ruleset examples for both the 2.2.x and 2.2.x kernels
- Updated the simple and stronger IPCHAINS and IPFWADM rulesets to include the external interface names (IPCHAINS is −i; IPFWADM is −W) to avoid some internal traffic MASQing issues.

- Vastly expanded the Section 5 (testing) with even more testing steps with added complete examples of what the output of the testing commands should look like.

- Moved the H.323 application documentation from NOT supported to Supported. :)

- Reordered the Multiple LAN section examples (2.2.x then 2.0.x)
- Made some additional clarifications to the Multiple LAN examples

  Fixed a critical typo with multiple NIC MASQing where the network examples had the specified networks reversed. Thanks to Matt Goheen for catching this.

- Added a little intro to MFW in the PORTFW section.
- Reveresed the 2.0.x and 2.2.x sections for PORTFW
- Updated the news regarding PORTFWing FTP traffic for 2.2.x kernels

```
NOTE:  At this time, there *IS* a BETA level IP_MASQ_FTP module
       for PORT Forwarding FTP connections 2.2.x kernels which also supports
       adding additional PORTFW FTP ports on the fly without the requirement
       of unloading and reloaded the IP_MASQ_FTP module and thus breaking any
       existing FTP transfers.
```

- Added a top level note about PORTFWed FTP support
- Added a noted to the 2.0.x PORTFW'ed FTP example why users DON'T need to PORTFW port 20.
- Updated the PORTFW section to also mention that users can use FTP proxy applications like the one from SuSe to support PORTFWed FTP−like functionality. Thanks to Stephen Graham for this one.
- Updated the example for how to enable PORTFWed FTP to also include required configurations to how the ip_masq_ftp module is loaded for users who use multiple PORTs to contact multiple internal FTP servers. Thanks to Bob Britton for reminding me about this one.

- Added a FAQ entry for users who have embedded ^Ms in their rc.firewall file
- Expanded the FAQ entry talking about how MASQ is different from NAT and Proxy to include some informative URLs.
- Updated the explination of the MASQ MTU issue and describe the two main explinations of the issue.
- Clarified that per the RFC, PPPoE should only require an MTU of 1490 though some ISPs require a setting of 1460. Because of this, I have updated the example to show an MTU of

1490.

- Broke out the Windows 9x sections into Win95 and Win98 as they use different settings (DWORD vs. STRING). I also updated the sections to be more clear and the Registry backup methods have been updated.
- Fixed a typo where the NT 4.0 Registry entries were backwards (Tcpip/Parameters vs. Parameters/Tcpip).
- Fixed an issue where the WinNT entry should have been a DWORD and not a STRING.
- A serious thanks goes out to Geoff Mottram for his various PPPoE and various Windows Registry entry fixes.

- Added an explict URL for Oident in the IRC FAQ entry
- Updated the FAQ section regarding some broken "netstat" versions
- Added new FAQ sections for MASQ accounting ideas and traffic shaping
- Expanded the IPROUTE2 FAQ entry on what Policy−routing is.
- Moved the IPROUTE2 URLs to the 2.2.x Kernel requirements section and also added a few more URLs as well.

- Corrected the "intnet" varible in the stronger IPCHAINS ruleset to reflect the 192.168.0.0 network to be consistent with the rest of the example. Thanks to Ross Johnson for this one.

- Added a new FAQ section for people asking about forwarding problems between multiple internal MASQed LANs.
- Added a new FAQ section about users wanting to PORTFW all ports from multiple external IP addresses to internal ones. I also touched on peopel trying to PORTFW all ports on multiple IP ALIASed interfaces and also noted the Bridge+Firewall HOWTO for DSL and Cablemodem users who have multiple IPs in a non−routed environment.
- Added Mandrake 7.1, Mandrake 7.2, and Slackware 7.1 to the supported list
- Added Redhat 7.0 to the MASQ supported distros. Thanks to Eugene Goldstein for this one.
- Fixed a mathmatical error in the "Maximum Throughput" calculation in the FAQ section. Thanks to Joe White @ ip255@msn.com for this one.
- Fixed the fact that the Windows9x MTU changes are a STRING change and not a DWORD change to the registry. Thanks to jmoore@sober.com for this one.
- Updated the comments in the 2.0.x rc.firewall script to note that the ip_defrag option is for both 2.0 and 2.2 kernels. Thanks to pumilia@est.it for this clarification.

Changes from 1.85 to 1.90 − 07/03/00

- Updated the URL for TrinityOS to reflect its new layout
- Caught a typo in the IPCHAINS rulesets where I was setting "ip_ip_always_defrag" instead of "ip_always_defrag"
- The URL to Taro Fukunaga was invaild since it was using "mail:" instead of "mailto:"
- Added some clarification to the "Masqing multiple internal interfaces" where some people didn't understand why eth0 was referenced multiple times.
- Fixed another "space after the EXTIP variable" bug in the stronger IPCHAINS section. I guess I missed one.
- In Test #7 of Section 5, I referred users to go back to step #4. Thats should have been step #6.
- Updated the kernel versions that came with SuSe 5.2 and 6.0
- Fixed a typo (or vs. of) in Section 7.2
- Added Item #9 to the Testing MASQ section to refer users who are still haing MASQ problems to read the MTU entry in the FAQ
- Improved the itemization in Section 5

- Updated the IPCHAINS syntax to show the MASQ/FORWARD table. Before, it was valid to run "ipchains −F −L" but now only "ipchains −M −L" works.
- Updated the LooseUDP documentation to reflect the new LooseUDP behavior in 2.2.16+ kernels. Before, it was always enabled, now, it defaults to OFF due to a possible MASQed UDP port scanning vunerability. I have updated the BASIC and SEMI−STRONG IPCHAINS rulesets to reflect this option.
- Updated the recommended 2.2.x kernel to be 2.2.16+ since there is a TCP root exploit vunerability in all lesser versions.
- Added Redhat 6.2 to the MASQ supported list
- Updated the link for Sonny Parlin's FWCONFIG to now point to fBuilder.
- Updated the various example IP addresses from 111.222.333.444 to be 111.222.121.212 to be within a valid IP address range
- Updated the URL for the BETA H.323 MASQ module
- Finally updated the MTU FAQ section to help out PPPoE DSL and Cablemodem users. Basically, the MTU−issues section now reflects that users can also change the MTU settings of all of their INTERNAL machines to solve the dreaded MASQ MTU issue.
- Added a clarification to the PORTFW section that PORTFWed connections that work for EXTERNAL clients will not work for INTERNAL clients. If you also need INTERNAL portfw, you will need to also impliment the REDIR tool as well. I also noted that this issue is fixed in the 2.4.x kernels with Netfilter.
- I also added a technical explination from Juanjo to the end of the PORTFW section to why this senario doesn't work properly.
- Updated all of the IPCHAINS URLs to point to Paul Rusty's new site at http://netfilter.filewatcher.org/ipchains/
- Updated Paul Rustys email address
- Added a new FAQ section for users whose connections remain idle for a long time and their PORTFWed connection no longer work.
- Updated all the URLs to the LDP that pointed to metalab.unc.edu to the new site of linuxdoc.org
- Updated the Netfilter URLs to point to renamed HOWTOs, etc.
- I also updated the status of the 2.4.x support to note that I *will* add full Netfilter support to this HOWTO and if the time comes, then split that support off into a different HOWTO.
- Updated the 2.4.x Requirements section to reflect how NetFilter has changed compared to IPFWADM and IPCHAINS and gave a PROs/CONs list of new features and changes to old behaviors.
- Added a TCP/IP math example to the "My MASQ connection is slow" FAQ entry to better explain what a user should expect performance wise.
- Updated the HOWTO to reflect that newer versions of the "pump" DHCP client now can run scripts upon bringup, lease renew, etc.
- Updated the PORTFWing of FTP to reflect that several users say they can successfully forward FTP traffic to internal machines without the need of a special ip_masq_ftp module. I have made the HOWTO reflect that users should try it without the modified module first and then move to the patch if required.

Changes from 1.82 to 1.85 − 05/29/00

- Ambrose Au's name has been taken off the title page as David Ranch has been the primary maintainer for the HOWTO for over a year. Ambrose will still be involved with the WWW site though.
- Deleted a stray SPACE in section 6.4
- Re−ordered the compatible MASQ'ed OS section and added instructions for setting up a

AS/400 system running on OS/400. Thanks to jaco@libero.it for the notes.
- Added an additional PORFW−FTP patch URL for FTP access if HTTP access fails.
- Updated the kernel versions for Redhat 5.1 & 6.1 in the FAQ
- Added FloppyFW to the list of MASQ−enabled Linux distros
- Fixed an issue in the Stronger IPFWADM rule set where there were spaces between "ppp_ip" and the "=".
- In the kernel compiling section for 2.2.x kernels, I removed the reference to enable "CONFIG_IP_ALWAYS_DEFRAG". This option was removed from the compiling section and enabled by default with MASQ enabled in 2.2.12.
- Because of the above change in the kernel behavior, I have added the enabling of ip_always_defrag to all the rc.firewall examples.
- Updated the status of support for H.323. There is now ALPHA versions of modules to support H.323 on both 2.0.x and 2.2.x kernels.
- Added Debian v2.2 to the supported MASQ distributions list
- Fixed a long standing issue where the section that covered explict filtering of IP addresses for IPCHAINS had old IPFWADM syntax. I've also cleaned this section up a little and made it a little more understandable.
- Doh! Added Juan Ciarlante's URL to the important MASQ resources section. Man.. you guys need to make me more honest than this!!
- Updated the HOWTO to reflect kernels 2.0.38 and 2.2.15
- Rerversed the order shown to compile kernels to show 2.2.x kernels first as 2.0.x is getting pretty old.
- Updated the 2.2.x kernel compiling section to reflect the changed options for the latter 2.2.x kernels.
- Added a a possible solution for people that fail to get past MASQ test #5.

Changes from 1.81 to 1.82 − 01/22/00

- Added a missing subsection for /proc/sys/net/ipv4/ip_dynaddr in the stronger IPCHAINS ruleset. Section 6.5
- Changed the IP Masq support for Debian 2.1 to YES
- Reorganized and updated the "Masq is slow" FAQ section to include fixing Ethernet speed and duplex issues.
- Added a link to Donald Becker's MII utilities for Ethernet NIC cards
- Added a missing ")" for the 2.2.x section (previously fixed it only for the 2.0.x version) to the ICQ portfw script and changed the evaluation from −lt to −le
- Added Caldera eServer v2.3 to the MASQ supported list
- Added Mandrake 6.0, 6.1, 7.0 to the MASQ supported list
- Added Slackware v7.0 to the MASQ supported list
- Added Redhat 6.1 to the MASQ supported list
- Added TurboLinux 4.0 Lite to the MASQ supported list
- Added SuSe 6.3 to the MASQ supported list
- Updated the recommended stable 2.2.x kernel to be anything newer than 2.2.11
- In section 3.3, the HOWTO forgot how to tell the user how to load the /etc/rc.d/rc.firewall upon each reboot. This has now been covered for Redhat (and Redhat−based distros) and Slackware.
- Added clarification in the Windows WFWG v3.x and NT setup sections why users should NOT configure the DHCP, WINS, and Forwarding options.
- Added a FAQ section on how to fix FTP problems with MASQed machines.
- Fixed a typo in the Stronger firewall rulesets. The "extip" variabl cannot have the SPACE between the variable name and the "=" sign. Thanks to johnh@mdscomp.com for the sharp

eye.
- Updated the compatibly section: Mandrake 7.0 is based on 2.2.14 and TurboLinux v6.0 runs 2.2.12

Changes from 1.80 to 1.81 − 01/09/00

- Updated the ICQ section to reflect that the new ICQ Masq module supports file transfer and real−time chat. The 2.0.x module still has those limitations.
- Updated Steven E. Grevemeyer's email address. He is the maintainer of the IP Masq Applications page.
- Fixed a few lines that were missing the work AREN'T for the "setsockopt" errors.
- Updated a error the strong IPCHAINS ruleset where it was using the variable name "ppp_ip" instead of "extip".
- Fixed a "." vs a "?" typo in section 3.3.1 in the DHCP comment section.
- Added a missing ")" to the ICQ portfw script and changed the evaluation from −lt to −le
- Updated the Quake Module syntax to NOT use the "ports=" verbage

Changes from 1.79 to 1.80 − 12/26/99

- Fixed a space typo when setting the "ppp_ip" address.
- Fixed a typo in the simple IPCHAINS ruleset. "deny" to "DENY"
- Updated the URLs for Bjorn's "modutils" for Linux
- Added verbage about NetFilter and IPTables and gave URLs until it is added to this HOWTO or a different HOWTO.
- Updated the simple /etc/rc.d/rc.firewall examples to notify users about the old Quake module bug.
- Updated the STRONG IPFWADM /etc/rc.d/rc.firewall to clarify users about dynamic IP addresses (PPP & DHCP), newer DHCPCD syntax, and the old Quake module bug.
- Updated the STRONG IPCHAINS /etc/rc.d/rc.firewall to ADD a missing section on dynamic IP addresses (PPP & DHCP) and the old Quake module bug.
- Added a note in the "Applications that DO NOT work" section that there IS a beta module for Microsoft NetMeeting (H.323 based) v2.x on 2.0.x kernels. There is NO versions available for Netmeeting 3.x and/or 2.2.x kernels as of yet.

Changes from 1.78 to 1.79 − 10/21/99

- Updated the HOWTO name to reflect that it isn't a MINI anymore!

Changes from 1.77 to 1.78 − 8/24/99

- Fixed a typeo in "Section 6.6 − Multiple Internal Networks" where the −a policy was ommited.
- Deleted the 2.2.x kernel configure option "Drop source routed frames" since it is now enabled by default and the kernel compile option was removed.
- Updated the 2.2.x and all other IPCHAINS sections to notify users of the IPCHAINS fragmentation bug.
- Updated all the URLs point at Lee Nevo's old IP Masq Applications page to Seg's new page.

Changes from 1.76 to 1.77 − 7/26/99

- Fixed a typo in the Port fowarding section that used "ipmasqadm ipportfw −C" instead of "ipmasqadm portfw −f"

Changes from 1.75 to 1.76 − 7/19/99

- Updated the "ipfwadm: setsockopt failed: Protocol not available" message in the FAQ to be more clear instead of making the user hunt for the answer in the Forwarders section.
- Fixed incorrect syntax in section 6.7 for IPMASQADM and "portfw"

Changes from 1.72 to 1.75 − 6/19/99

- Fixed the quake module port setup order for the weak IPFWADM & IPCHAINS ruleset and the strong IPFWADM ruleset as well.
- Added a user report about port forwarding ICQ 4000 directly in and using ICQ's default settings WITHOUT enabling the "Non−Sock" proxy setup.
- Updated the URLs for the IPMASQADM tool
- Added references to Taro Fukunaga, tarozax@earthlink.net for his MkLinux port of the HOWTO
- Updated the blurb about Sonny Parlin's FWCONFIG tool to note new IPCHAINS support
- Noted that Fred Vile's patch for portfw'ed FTP access is ONLY available for the 2.0.x kernels
- Updated the 2.2.x kernel step with a few clarifications on the Experiemental tag
- Added Glen Lamb's name to the credits for the LooseUDP patch
- Added a clarification on installing the LooseUDP patch that it should use "cat" for non−compressed patches.
- Fixed a typo in the IPAUTO FAQ section
- I had the DHCP client port numbers reversed for the IPFWADM and IPCHAINS rulesets. The order I had was if your Linux server was a DHCP SERVER.
- Added explict /sbin path to all weak and strong ruleset examples.
- Made some clarifications in the strong IPFWADM section regarding Dynamic IP addresses for PPP and DHCP users. I also noted that the strong rulesets should be re−run when PPP comes up or when a DHCP lease is renewed.
- Added reference in the 2.2.x requirements, updated the ICQ FAQ section, and added Andrew Deryabin to credits section for his ICQ MASQ module.
- Added some clarifcation in the FAQ section why the 2.1.x and 2.2.x kernels went to IPCHAINS.
- Added a little FAQ section on Microsoft File/Print/Domain services (Samba) through a MASQ server. I also added a URL to a Microsoft Knowledge base document for more details.
- Added clarification in the FAQ section that NO Debian distribution supports IP masq out of the box.
- Updated the supported MASQ distributions in the FAQ section.
- Added to the Aliased NIC section of the FAQ that you CANNOT masq out of an aliased interface.
- Wow.. never caught this before but the "ppp−ip" variable in the strong ruleset section is an invalid variable name! It has been renamed to "ppp_ip"
- In both the IPFWADM and IPCHAINS simple ruleset setup areas, I had a commented out section on enabling DHCP traffic. Problem is, it was below the final reject line! Doh! I moved both up a section.
- In the simple IPCHAINS setup, the #ed out line for DHCP users, I was using the IPFWADM "−W" command instead of IPCHAINS's "−i" parameter.
- Added a little blurb to the Forwarders section the resolution to the famous "ipfwadm: setsockopt failed: Protocol not available" error. This also includes a little /proc test to let people confirm if IPPORTFW is enabled in the kernel. I also added this error to a FAQ

section for simple searching.
- ♦ Added a Strong IPCHAINS ruleset to the HOWTO
- ♦ Added a FAQ section explaining the "kernel: ip_masq_new(proto=UDP): no free ports." error.
- ♦ Added an example of scripting IPMASQADM PORTFW rules
- ♦ Updated a few of the Linux Documentation Project (LDP) URLs
- ♦ Added Quake III support in the module loading sections of all the rc.firewall rulesets.
- ♦ Fixed the IPMASQADM forwards for ICQ

- 1.72 − 4/14/99 − Dranch: Added a large list of Windows NAT/Proxy alternatives with rough pricing and URLs to the FAQ.

- 1.71 − 4/13/99 − Dranch: Added IPCHAINS setups for multiple internal MASQed networks. Changed the ICQ setup to use ICQ's default 60 second timeout and change IPFWADM/IPCHAINS timeout to 160 seconds. Updated the MASQ and MASQ−DEV email list and archive subscription instructions.

- 1.70 − 3/30/99 − Dranch: Added two new FAQ sections that cover SMTP/POP−3 timeout problems and how to masquerade multiple internal networks out different external IP addresses with IPROUTE2.

- 1.65 − 3/29/99 − Dranch: Typo fixes, clarifications of required 2.2.x kernel options, added dynamic PPP IP address support to the strong firewall section, additional quake II module ports, noted that the LooseUDP patch is built into later 2.2.x kernels and its from Glenn Lamb and not Dan Kegel, added more game info in the compatibility section.

- 1.62 − Dranch: Make the final first−draft changes to the doc and now announce it the the MASQ email list.

- 1.61 − Dranch: Make editorial changes, cleaned things up and fixed some errors in the Windows95 and NT setups.

- 1.58 − Dranch: Addition of the port forwarding sections; LooseUDP setup; Ident servers for IRC users, how to read firewall logs, deleted the CuSeeme Mini−HOWTO since it is rarely used.

- 1.55 − Dranch: Complete overhaul, feature and FAQ addition, and editing sweep of the v1.50 HOWTO. Completed the 2.2.x kernel and IPCHAINS configurations. Did a conversion from IPAUTOFW to IPPORTFW for the examples that applied. Added many URLs to various other documentation and utility sites. There are so many changes.. I hope everyone likes it. Final publishing of this new rev of the HOWTO to the LDP project won't happen until the doc is looked over and approved by the IP MASQ email list (then v2.00).

- 1.50 − Ambrose: A serious update to the HOWTO and the initial addition of the 2.2.0 and IPCHAINS configurations.

- 1.20 − Ambrose: One of the more recent HOWTO versions that solely dealt with < 2.0.x kernels and IPFWADM.