Using Linux iptables or ipchains to set up an internet gateway / firewall / router for home or office

Methods of connecting your network to the internet:

- Use Linux ipchains / iptables and IP forwarding to configure Linux as a firewall and router. This is the method covered in this tutorial.
- Use SOCKS gateway proxy software running on Linux. (See SOCKS below)
- Use a CISCO router - Configuration tutorial. (Note: PIX series are preferred for firewall use.)

This tutorial will cover using a linux computer as a gateway between a private network and the internet. Any internet connection whether it be a dial-up PPP, DSL, cable modem or a T1 line can be used. In the case of most dial-up PPP connections and cable modem connections, only a single IP address is issued allowing only one computer to connect to the internet at a time. Using Linux and **iptables / ipchains** one can configure a gateway which will allow all computers on a private network to connect to the internet via the gateway and one external IP address, using a technology called "Network Address Translation" (NAT) or masquerading and private subnets. Iptables/ipchains can also be configured so that the Linux computer acts as a firewall, providing protection to the internal network.

search Search | Home Page | Linux Tutorials | Terms | Privacy Policy | Advertising | Contact |

Related YoLinux Tutorials:

°Linux Networking

°Linux Sys Admin

°Web site configuration

°Internet Security

°YoLinux Tutorials Index

Firewall versions vs Linux kernel versions:

Note: References to ipfwadm and ipchains refer to older deprecated software.

Firewall Command	Linux Kernel Version	Red Hat Version
iptables	2.4.x, 2.6.x	7.1 - 9.0, Fedora 1,2,3
ipchains	2.2.x	6.x, 7.0
ipfwadm	2.0.x	5.x

Note: Red Hat 7.1-9.0 and the default Linux 2.4 kernel may use ipchains or iptables but not both. Iptables is the preferred firewall as it supports "state" and can recognize if a network connection has already been "ESTABLISHED" or if the connection is related to the previous connection (required for ftp which makes multiple connections on different ports). Ipchains can not. Ipchain rules take precedence over iptables rules. During system boot, the kernel attempts to activate ipchains, then attempts to activate iptables. If ipchain rules have been activated, the kernel will not start iptables.

Red Hat 7.1 will not support ipchains unless that option is configured (during install or later). If during install you select "Disable Firewall - no protection" then ipchains will not be available and you must rely upon iptables for a manual firewall configuration. (iptables only. ipchains will be unavailable)

GUI configuration:

- **iptables:** The GUI configuration tool `/usr/bin/redhat-config-securitylevel` can be used to choose a preconfigured firewall (High, Medium or no firewall) or it can be used to manually configure rules based on the network services your server will offer. The init script `/etc/rc.d/init.d/iptables` will use rules stored in `/etc/sysconfig/iptables`.
- **ipchains:** The tool that does this is lokkit (or `/usr/bin/gnome-lokkit`), which uses ipchains to configure firewall options for High and Low security options. To support ipchains after install, run `/usr/bin/gnome-lokkit` and configure a firewall. It will configure ipchains to activate the firewall. Lokkit will generate the file `/etc/sysconfig/ipchains`. (Used by init script `/etc/rc.d/init.d/ipchains` which calls `/sbin/ipchains-restore`)

  To see if ipchains and the Lokkit configuration is invoked during system boot, use the command:

  ```
  chkconfig --list | grep ipchains
  ```

The default Red Hat 7.1+ Linux 2.4 kernel is compiled to support both iptables and ipchains. Kernel support for ipchains is available during a kernel configuration and compilation. During `make xconfig` or `make menuconfig` turn on the feature: "IP: Netfilter Configuration" + "ipchains (2.2-style) support".

Check your installation by using the command: `rpm -q iptables ipchains`
These packages must be installed. The commands iptables and ipchains are the command interfaces to configure kernel firewall rules. The default Red Hat 7.1 kernel supports iptables and ipchains. (But not both at the same time.)

[Potential Pitfall]: When performing an upgrade instead of a new install, the upgrade software

will not install iptables as did not exist on the system previously. It will perform an upgrade to a newer version of ipchains. If you wish to use iptables, you must manually install the iptables RPM.
i.e.: `rpm -ivh iptables-XXX.i386.rpm`

[Potential Pitfall]: The Linux operating system kernel may load or not load what you had expected. Use the command `lsmod` to see if ip_tables or ip_chains were loaded.

Switching a running system from ipchains to iptables: (Red Hat 7.1-9.0 - Linux kernel 2.4 specific)

Sequence	Command	Description
1	chkconfig --del ipchains	Remove ipchains from system boot/initialization process
2	chkconfig --add iptables	Add iptables to system boot/initialization process
3	ipchains -F	Flush ipchains rules
4	service ipchains stop	Stop ipchains. Also: `/etc/init.d/ipchains stop`
5	rmmod ipchains	Unload ipchains kernel module. Iptables kernel module can not be loaded if the ipchains module is loaded
6	service iptables start	Load iptables kernel module. Also: `/etc/init.d/iptables stop`

Network Address Translation (NAT):

An individual on a computer on the private network may point their web browser to a site on the internet. This request is recognized to be beyond the local network so it is routed to the Linux gateway using the private network address. The request for the web page is sent to the web site using the external internet IP address of the gateway. The request is returned to the gateway which then translates the IP address to computer on the private network which made the request. This is often called IP masquerading. The software interface which enables one to configure the kernel for masquerading is **iptables (Linux kernel 2.4)** or **ipchains (Linux kernel 2.2)**

The gateway computer will need two IP addresses and network connections, one to the private internal network and another to the external public internet.

A note on private network IP addresses: A set of IP addresses has been reserved by IANA for private networks. They range from 192.168.0.1 to 192.168.254.254 for a typical small business or home network and are often referred to as CIDR private network addresses. Most private networks conform to this scheme.

Block	Range		CIDR Notation	Default Subnet Mask	Number of hosts
24 bit block in class A	10.0.0.0	10.255.255.255	10.0.0.0/8	255.0.0.0	16,777,216
20 bit block in class B	172.16.0.0	172.31.255.255	172.16.0.0/12	255.240.0.0	1,048,576
16 bit block in class C	192.168.0.0	192.168.255.255	192.168.0.0/16	255.255.0.0	65,536

The actual number of hosts will be fewer that listed because addresses on each subnet will be reserved as a broadcast address, etc.

This is detailed in RFC 1918 - Address Allocation for Private Internets. For a description of class A, B, and C networks see the YoLinux Networking Tutorial class description.

The private networks may be subdivided into various subnets as desired. Examples:

Range		CIDR Notation	Default Subnet Mask	Number of hosts
10.2.3.0	10.2.4.255	10.2.3.0/23	255.255.254.0	512
172.16.0.0	172.17.255.255	172.16.0.0/15	255.254.0.0	132608
192.168.5.128	192.168.5.255	192.168.5.128/25	255.255.255.128	128

Example 1: Linux connected via PPP

This example uses a Linux computer connected to the internet using a dial-up line and modem (PPP). The Linux gateway is connected to the internal network using an ethernet card. The internal network consists of Windows PC's.

The Linux box must be configured for the private internal network and PPP for the dial-up connection. See the PPP tutorial to configure the dial-up connection. Use the **ifconfig** command to configure the private network. i.e. (as root)

```
/sbin/ifconfig eth1 192.168.10.101 netmask 255.255.255.0 broadcast 192.168.10.255
```

This is often configured during install or can be configured using the Gnome tool `neat` (or the admin tool `Linuxconf` or `netcfg` for older Red Hat systems). System changes made with the `ifconfig` or `route` commands are **NOT** permanent and are lost upon system reboot. Permanent settings are held in configuration scripts executed during system boot. (i.e. `/etc/sysconfig/...`) See the YoLinux Networking tutorial for more information on assigning network addresses.

Run one of the following scripts on the Linux gateway computer:

iptables:

```
iptables --flush                             # Flush all the rules in filter and nat tables
iptables --table nat --flush
iptables --delete-chain                      # Delete all chains that are not in default filter and nat table
iptables --table nat --delete-chain

# Set up IP FORWARDing and Masquerading
iptables --table nat --append POSTROUTING --out-interface ppp0 -j MASQUERADE
iptables --append FORWARD --in-interface eth0 -j ACCEPT         # Assuming one NIC to local LAN

echo 1 > /proc/sys/net/ipv4/ip_forward    # Enables packet forwarding by kernel
```

ipchains:

```
#!/bin/sh
ipchains -F forward                                    # Flush all previous rules and settings
ipchains -P forward DENY                               # Default set to deny packet forwarding
ipchains -A forward -s 192.168.10.0/24 -j MASQ         # Use IP address of gateway for private network
ipchains -A forward -i ppp0 -j MASQ                    # Sets up external internet connection
echo 1 > /proc/sys/net/ipv4/ip_forward                # Enables packet forwarding by kernel
```

A PPP connection as described by the YoLinux PPP tutorial will create the PPP network connection as the default route.

Example 2: Linux connected via DSL, Cable, T1

High speed connections to the internet result in an ethernet connection to the gateway. Thus the gateway is required to possess two ethernet Network Interface Cards (NICs), one for the connection to the private internal network and another to the public internet. The ethernet cards are named **eth** and are numbered uniquely from 0 upward.

Use the **ifconfig** command to configure both network interfaces.

```
/sbin/ifconfig eth0 XXX.XXX.XXX.XXX netmask 255.255.255.0 broadcast XXX.XXX.XXX.255  # Internet
/sbin/ifconfig eth1 192.168.10.101 netmask 255.255.255.0 broadcast 192.168.10.255    # Private LAN
```

Also see notes on adding a second NIC.

This is often configured during install or can be configured using the Gnome tool `neat` (or the admin tool `Linuxconf` or `netcfg` for older Red Hat systems). System changes made with the `ifconfig` or `route` commands are **NOT** permanent and are lost upon system reboot. Permanent settings are held in configuration scripts executed during system boot. (i.e. `/etc/sysconfig/...`) See the YoLinux Networking tutorial for more information on assigning network addresses.

Run the appropriate script on the linux computer where eth0 is connected to the internet and eth1 is connected to a private LAN:

iptables:

```
# Delete and flush. Default table is "filter". Others like "nat" must be
```

```
explicitly stated.
iptables --flush            # Flush all the rules in filter and nat tables
iptables --table nat --flush
iptables --delete-chain     # Delete all chains that are not in default filter and nat table
iptables --table nat --delete-chain

# Set up IP FORWARDing and Masquerading
iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
iptables --append FORWARD --in-interface eth1 -j ACCEPT

echo 1 > /proc/sys/net/ipv4/ip_forward             # Enables packet forwarding by kernel
```

ipchains:

```
#!/bin/sh
ipchains -F forward                                # Flush rules
ipchains -P forward DENY                           # Default set to deny packet forwarding
ipchains -A forward -s 192.168.10.0/24 -j MASQ     # Use IP address of gateway for private network
ipchains -A forward -i eth1 -j MASQ                # Sets up external internet connection
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Create a route for internal packets:

```
route add  -net 192.168.10.0  netmask 255.255.255.0 gw XXX.XXX.XXX.XXX dev eth1
```

Where *xxx.xxx.xxx.xxx* is the internet gateway defined by your ISP. For more information on routing see the YoLinux networking tutorial

Note: While this configuration requires that the Linux gateway computer have two network cards, if you only have one PCI slot available you may use a card such as the Intel Pro 100 or Pro 1000 Dual Port which has two ethernet connections which reside on a single card. (This is what I use) Yolinux Hardware tutorial: More on Network interface cards

Intel PCI Dual Pro 100 or Pro 1000 NIC card supports two physical ethernet connections (eth0, eth1) on one card.
Compliant Standards: IEEE 802.3-LAN, IEEE 802.3U-LAN , Plug and Play
Connectivity Technology: Cable - 10Base-T, 100Base-TX
Data Link Protocol: Ethernet, Fast Ethernet
Processor: 82550 - Intel

Iptables options: (Linux kernel 2.4/2.6 firewall)

General /sbin/iptables format to add rules:
`iptables [-t|--table table] -command [chain] [-i interface] [-p protocol] [-s address [port[:port]]] [-d address [port[:port]]] -j policy`

Six pre-defined "chain" rules are available:

- INPUT
- OUTPUT
- INPUT
- FORWARD
- PREROUTING
- POSTROUTING
- User defined chains (just give it a new name instead of one of the pre-defined names)

iptables options:

--table -t	Description
filter	Default table. This is used if not specified
nat	Network address translation
mangle	Used for Quality Of Service (QOS) and preferential

Command (Use one)	Description
-A --append	Append rule to chain
-D --delete	Delete rule from chain
-I --insert	Insert rule at beginning or at specified sequence

Command Option	Description
-s --source	Source address of packet
-d --destination	Destination address of packet
-i --in-interface	Interface packet is arriving from

Defined Policies	Description
ACCEPT	Let packet through
DROP	Deny packet with no reply
REJECT	Deny packet and notify sender

	treatment
raw	Enables optimization. i.e. Ignore firewall state matching for port 80 for enhanced speed due to less processing. Requires kernel patch

	number in chain.
-R --replace	Replace rule
-F --flush	Flush all rules
-Z --zero	Zero byte counters in all chains
-L --list	List all rules. Add option `--line-numbers` for rule number.
-N --new-chain	Create new chain
-X --delete-chain	Delete user defined chain
-P --policy	Set default policy for a chain
-E --rename-chain	Rename a chain

-o --out-interface	Interface packet is going to
-p --protocol	Protocol: °tcp --sport port[:port] --dport port[:port] --syn °udp °icmp °mac ...
-j --jump	Target to send packet to
-f --fragment	Fragment matching
-c --set-counters	Set packet/byte counter
-m tcp --match tcp	°--source-port *port[:port]* (port # or range #:#) °--destination-port *port[:port]* °--tcp-flags
-m state --match state	--state °ESTABLISHED °RELATED °NEW °INVALID (Push content, not expected to recieve this packet.)

RETURN	Handled by default targets
MARK	Used for error response. Use with option --reject-with *type*
MASQUERADE	Used with nat table and DHCP.
LOG	Log to file and specify message: °--log-level # °--log-prefix "*prefix*" °--log-tcp-sequence °--log-tcp-options °--log-ip-options
ULOG	Log to file and specify userpace logging messages
SNAT	Valid in PREROUTING chain. Used by nat.
REDIRECT	Used with nat table. Output.
DNAT	Valid in POSTROUTING chain. Output.
QUEUE	Pass packet to userspace.

For the full info see the man page for iptables.

Ipchains options: (Linux kernel 2.2 firewall)

General /sbin/ipchains format to add rules:

```
ipchains -A|I [chain] [-i interface] [-p protocol] [-y] [-s address [port[:port]]] [-d address [port[:port]]] -j policy
[-l]
```

ipchains options:

Command	Description
-A	Add rule to chain
-D	Delete rule from chain
-I	Insert rule
-R	Replace rule
-F	Flush all rules
-L	List all rules
-N	Create new chain
-X	Delete user defined chain
-P	Set default targe

Command Option	Description
-s	Source address of packet
-d	Destination address of packet
-i	Interface packet is arriving from
-p	Protocol
-j	Target to send packet to
-y	For -p tcp. Packet is SYN packet.
--icmp-type	For -p icmp.
-l	Log the packet to syslog. `/var/log/messages` Available in default Red Hat 6.0+ kernel

System targets (policy)	Description
ACCEPT	Let packet through
DENY	Deny packet
REJECT	Deny packet and notify sender
MASQ	Forward chain masquerade
REDIRECT	Send to different port
RETURN	Handled by default targets

Four chain rule types are available:

- IP input chain
- IP output chain
- IP forwarding chain
- User defined chains (just give it a new name instead of the built-in names: input, output or forward)

For the full info see the man page for ipchains. To add firewall rules read the links provided below.

Configuring PCs on the office network:

- All PC's on the private office network should set their "gateway" to be the local private network IP address of the Linux gateway computer.
- The DNS should be set to that of the ISP on the internet.

Windows '95 Configuration:

- Select "Start" + Settings" + "Control Panel"
- Select the "Network" icon
- Select the tab "Configuration" and double click the component "TCP/IP" for the ethernet card. (NOT the TCP/IP -> Dial-Up Adapter)
- Select the tabs:
    - "Gateway": Use the internal network IP address of the Linux box. (192.168.*XXX.XXX*)
    - "DNS Configuration": Use the IP addresses of the ISP Domain Name Servers. (Actual internet IP address)
    - "IP Address": The IP address (192.168.*XXX.XXX* - static) and netmask (typically 255.255.255.0 for a small local office network) of the PC can also be set here.

Linux computers:

- **IP Address**: Use ifconfig or netcfg commands to set the IP address and netmask.
  See Assigning an IP address portion of the Networking tutorial.
- **Gateway**: The gateway is set with the route command. This can also be set by the GUI tool /usr/bin/netcfg or console tool /usr/sbin/netconfig. It is also stored by the system in the /etc/sysconfig/network file.
- **DNS**: Configure file `/etc/resolv.conf` to set the DNS and default domain.
  See the Network configuration files portion of the Networking tutorial.
- Simple firewall for the desktop Linux system:

  ```
  iptables -P INPUT   DROP
  iptables -P FORWARD DROP
  iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
  iptables -A INPUT -i lo -j ACCEPT
  iptables -A OUTPUT -o lo -j ACCEPT
  ```

  Allow network connections which have already been established (started by host) and related to your connection. FTP requires this as it may use various ports in support of the file transfer.)
  Allow network input/output from self (lo).

Adding more security rules to your gateway:

iptables:

Deny a specific host: `iptables -I INPUT -s` ***`XXX.XXX.XXX.XXX`*** `-j DROP`

Block ports by adding the following firewall rules:

```
# Allow loopback access. This rule must come before the rules denying port access!!
iptables -A INPUT  -i lo -p all -j ACCEPT  # Rule for your computer to be able to access itself via the loopback
iptables -A OUTPUT -o lo -p all -j ACCEPT

iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 2049 -j DROP       # Block NFS
iptables -A INPUT -p udp -s 0/0 -d 0/0 --dport 2049 -j DROP       # Block NFS
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 6000:6009 -j DROP  # Block X-Windows
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 7100 -j DROP       # Block X-Windows font server
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 515 -j DROP        # Block printer port
iptables -A INPUT -p udp -s 0/0 -d 0/0 --dport 515 -j DROP        # Block printer port
iptables -A INPUT -p tcp -s 0/0 -d 0/0 --dport 111 -j DROP        # Block Sun rpc/NFS
iptables -A INPUT -p udp -s 0/0 -d 0/0 --dport 111 -j DROP        # Block Sun rpc/NFS
iptables -A INPUT -p all -s localhost  -i eth0 -j DROP            # Deny packets which claim to be from your loopback interface.
```

These rules may be executed on their own to protect your system while attached to the internet or they may be appended to the end of the iptables gateway NAT scripts above.

Debugging and logging:

```
iptables -A INPUT -j LOG --log-prefix "INPUT_DROP: "
iptables -A OUTPUT -j LOG --log-prefix "OUTPUT_DROP: "
```

Add this to the end of your rules and you should be able to monitor dropped connections in `/var/log/messages`. I do **NOT** log in this method due to the outrageous volume of messages it generates. Use this for debugging or short term monitoring of the network.

Another approach to firewalls is to drop everything and then grant access to each port you may need.

```
iptables -F
iptables -A INPUT -i lo -p all -j ACCEPT                       # Allow self access by loopback interface
iptables -A OUTPUT -o lo -p all -j ACCEPT
iptables -A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT # Accept established connections
iptables -A INPUT -p tcp --tcp-option ! 2 -j REJECT --reject-with tcp-reset
iptables -A INPUT -p tcp -i eth0 --dport 21 -j ACCEPT          # Open ftp port
```

```
iptables -A INPUT -p udp -i eth0 --dport 21 -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 22 -j ACCEPT          # Open secure shell port
iptables -A INPUT -p udp -i eth0 --dport 22 -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 80 -j ACCEPT          # Open HTTP port
iptables -A INPUT -p udp -i eth0 --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --syn -s 192.168.10.0/24 --destination-port 139 -j ACCEPT   # Accept local Samba connection
iptables -A INPUT -p tcp --syn -s trancas --destination-port 139 -j ACCEPT
iptables -P INPUT DROP                # Drop all other connection attempts. Only connections defined above are allowed.
```

---

ipchains:

This script configures firewall rules for a Linux computer with two ethernet ports. One port connects the computer to the internet with an external address of ***XXX.XXX.XXX.XXX***. The other ethernet port connects the computer to an internal network of 192.168.10.0 to 192.168.10.255. This script is more complex but preferred to the previous scripts because of the extra security that the extra firewall rules offer. The script does work with a system running portsentry. For more on portsentry see the YoLinux Internet Security: portsentry Tutorial.

Internet external network interface: eth0
Internal private network interface: eth1
Local loopback virtual interface: lo

Gateway script for ipchains firewall and NAT:

```
#!/bin/sh

# Flush Rules
ipchains -F forward
ipchains -F output
ipchains -F input

# Set default to deny all
ipchains -P input   DENY
ipchains -P output  DENY
ipchains -P forward DENY

# Add Rules

# Accept packets from itself (localhost) (s)ource to itself (d)estination
# Keeps system logging, X-Windows or any socket based service working.
ipchains -A input  -j ACCEPT -p all -s localhost -d localhost -i lo
ipchains -A output -j ACCEPT -p all -s localhost -d localhost -i lo

# Deny and log (option -l) spoofed packets from external network (eth0) which mimic internal IP addresses
ipchains -A input -j REJECT -p all -s 192.168.10.0/24 -i eth0 -l

# Accept requests/responses from/to your own firewall machine
ipchains -A input   -j ACCEPT -p all -d XXX.XXX.XXX.XXX -i eth0
ipchains -A output  -j ACCEPT -p all -s XXX.XXX.XXX.XXX -i eth0

# Allow outgoing packets source (s) to destination (d)
ipchains -A input   -j ACCEPT -p all -s 192.168.10.0/24 -i eth1
ipchains -A output  -j ACCEPT -p all -s 192.168.10.0/24 -i eth1

# Deny and log (option -l) outside packets from internet which claim to be from your loopback interface
ipchains -A input  -j REJECT -p all -s localhost  -i eth0 -l

ipchains -A forward -s 192.168.10.0/24 -j MASQ
ipchains -A forward -i eth1 -j MASQ

# Enable  packet forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Notes:

- For this example it was assumed that your private network is from 192.168.10.0 to 192.168.10.255
- The `-d 0.0.0.0/0` refers to all or any destination address of packet. (destination in this case is irrelevant and the -d statement may be omitted))
- localhost refers to your loopback interface on 127.0.0.1

Red Hat 7.1 will configure firewall rules as an option during installation. Note that the firewall rules are generated for ipchains. The configuration tool `/usr/bin/gnome-lokkit` was used to perform this setup.

Example of the security configuration: `/etc/sysconfig/ipchains`
This is the configuration file for the script `/etc/rc.d/init.d/ipchains` (which calls `/sbin/ipchains-restore`) which may be invoked during system boot.

```
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: ifup-post will punch the current nameservers through the
#       firewall; such entries will *not* be listed here.
:input ACCEPT
```

```
:forward ACCEPT
:output ACCEPT
-A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT                # Allow WWW http access to web server
-A input -s 0/0 -d 0/0 22 -p tcp -y -j ACCEPT                # Allow SSH (Secure Shell) access
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth0 -j ACCEPT  # Allow DHCP/BOOTPC
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth1 -j ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -s 0/0 -d 0/0 -i eth1 -j ACCEPT  # eth1 internal network access OK. External eth0 goes through firewall rules
-A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT    # This shuts off telnet,FTP,bind...! Use for a workstation only
-A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT
-A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT       # Workstation only or explicitly ports as above with 80, 22
-A input -p udp -s 0/0 -d 0/0 2049 -j REJECT         # Block NFS
-A input -p tcp -s 0/0 -d 0/0 6000:6009 -y -j REJECT # Block remote X-Window connections
-A input -p tcp -s 0/0 -d 0/0 7100 -y -j REJECT      # Block remote font server connections
```

Note: Once ipchains have been invoked for kernel 2.4 (RH 7.1), one may NOT use iptables. You may use one or the other but not both.

Save/restore an tables/ipchains configuration:

- IpTables: iptables-save man page
  ```
  /sbin/iptables-save > /etc/sysconfig/iptables.rules
  /sbin/iptables-restore < /etc/sysconfig/iptables.rules
  ```
- IpChains: ipchains-save man page
  ```
  /sbin/ipchains-save > /etc/sysconfig/ipchains.rules
  /sbin/ipchains-restore < /etc/sysconfig/ipchains.rules
  ```

The system init script looks for the file name `/etc/sysconfig/ipchains` instead of `/etc/sysconfig/ipchains.rules`. This will make the rules accessible to the init script which will invoke the rules upon system boot. See the YoLinux Init process tutorial for more information on init scripts and system boot procedures.

Also see: how to turn off ICMP and look invisible to ping.

---

proc file settings:

- Turning on Linux kernel support for spoof and DOS (Denial Of Service) protection:

  ```
  echo 1 >/proc/sys/net/ipv4/tcp_syncookies
  ```

  Must first be compiled into kernel. (Included in Redhat default kernel) By default the Redhat install has this disabled (set to 0). This helps to prevent against the common 'syn flood attack'. A connecting computer (peer) may not receive reliable error messages from an over loaded server with syncookies enabled.

  For more on SYS cookies see: CERT Advisory CA-96.21

- Turn on Source Address Verification: (Off by default on Red Hat install - set to 0)

  ```
  echo 1 >/proc/sys/net/ipv4/conf/eth0/rp_filter
  OR
  echo 1 >/proc/sys/net/ipv4/conf/all/rp_filter
  ```

  State the interface appropriate for your installation.
  The first example prevents spoofing attacks against your external networks only.

  IP spoofing is a technique where a host sends out packets which claim to be from another host. It is also used to hide the identity of the attacker.

The TCP Man page - Linux Programmer's Manual and `/usr/src/linux/proc.txt` [link] (Kernel 2.4) cover `/proc/sys/net/ipv4/*` file descriptions.

Also see:

- local file:/usr/src/linux/Documentation/proc.txt
- proc man page

IP Forwading Notes:

Choose one of the following to allow the Linux kernel to forward IP packets:

1. Immediately allow the forwarding of packets. The configuration is not preserved on reboot but sets a flag in the kernel itself.

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

2. Another method is to alter the Linux kernel config file: `/etc/sysctl.conf`
   Set the following value:

   ```
   net.ipv4.ip_forward = 1
   ```

   This will configure the system to allow forwarding of packets upon system boot. It is stored in this configuration file and thus read and set upon system boot. If set to "0" then there will be no forwarding of packets.

3. An alternate method is to alter the network script: `/etc/sysconfig/network`

```
FORWARD_IPV4=true
```

Change the default "false" to "true".

All the above methods will result in a proc file value of "1" to allow TCP packet forwarding. Options 2 and 3 set boot configurations in a configuration file and will not take effect until system boot.
Test the current setting of the kernel: `cat /proc/sys/net/ipv4/ip_forward`

Note: The `/proc` directory is NOT on your hard drive but is present in the running kernel.

CIDR Notation:

The notation `"/24"` refers to the use of the first 24 bits of a 32 IP address. The is the equivalent of using the bitmask `255.255.255.0`. To put it another way, it specifies a range of IP addresses: 0 to 255 for the last octet while the first three remain constant.

Example: 192.168.103.0/24 refers to the IP address range 192.168.103.0 to 192.168.103.255

The notation `"/32"` refers to a single IP address as it implies that all 32 bits of the IP address are significant.

Configuration Tools:

GUI tools and scripts exist to help you with the configuration of ipchains. See:

- Firestarter - Configuration of firewall and real-time hit monitor for the Gnome desktop. Configures ipchains (kernel 2.2) and iptables (kernel 2.4)
- Firewall Builder - iptables, ipfilter and OpenBSD PF. (GTK--)

Included with Red Hat 7.x is the Gnome GUI tool gnome-lokkit. (ipchains)

Tools for iptables configuration:

- Webmin - Linux web admin tool
- Shorewall
- NARC: Netfilter Automatic Rule Configurator

Links and information:

iptables:

- IpTables.org - Netfilter/Iptables home page
- Linux iptables syntax - by Shane Chen
- ipmenu - Console based application for viewing and editing iptables and chains.
- Bastille Linux - Security hardening system (script)
- IPTables Firewall Script - Bob Sully

ipchains:

- Man page for ipchains
- Man page for ifconfig
- Ipchains HOWTO - LDP - Paul Russell
- Linux Firewall Script - ipchains and ipfwadm scripts and configuration. (It's the fanciest I've seen.) - by Craig Zeller
- linas.org: Linux NAT, Load Balancing, and High Availability
- TrinityOS: NAT, MASQ Links
- Config /etc/rc.d/init.d/firewall script file -Web Server
- Config /etc/rc.d/init.d/firewall script file - Mail Server

Relevant networking links:

- Traffic shaping - bandwidth allocation using tc - by Shane Chen
  - tc examples
- **PPP Dialing your ISP - TUTORIAL**
- Man page for resolv.conf
- Man page for pppd
- Man page for chat
- Connecting to an ISP
- Networking overview HOWTO - LDP
- Modem HOWTO - LDP
- Linux Modem sharing mini-HOWTO
- Smoothwall.org - Web managed OS for Firewall, VPN, Dialup, Intrusion detection, DMZ, dynamic DNS, DHCP, port forwarding, ...
- DSLreports.com: Reviews of DSL providers, bandwidth speed measurement, Tools, Info
- Modem Sharing

Linux Based Routers:

- Leaf - Linux Embeded Application firewall
- Eigerstein

SOCKS Proxy Servers:

I can no longer find the NEC reference implementation but here are some other SOCKS proxy server options for Linux:

- sSOCKS5
- Polipo - SOKS 5, web caching, IPV6 suport
- DeleGate - proprietary software
- DeleGate
- DeleGate

One may also configure ssh to provide SOCKS5 proxy capability:

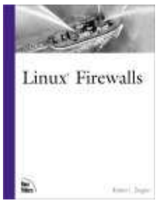
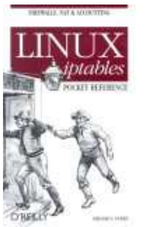
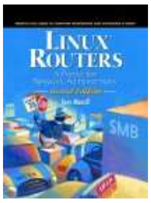
```
ssh -f -N -D 0.0.0.0:1080 localhost
```

Where:

- -D: port forwarding on port 1080. The IP address 0.0.0.0 specifies the socket option INADDR_ANY which means that it is listeneing for connections from any IP address.
- -N: stays idle and does not allow for the execution of commands on localhost
- -f: Run in the background as a daemon

Iptables can be used to further restrict IP sources accessing port 1080 and add further security constraints.

Books:

	"Linux Firewalls" by Robert L. Ziegler, Carl Constaintine ISBN #0735710996, New Riders 10/2001 Second edition. (Focus: iptables) This is the most up to date version of this book. It highlights the recent changes to the Linux 2.4 kernel (including iptables). It also includes coverage of VPS's and SSH. Scripts and examples are provided for almost any condition or purpose one may require. I highly recommend this book for anyone concerned about internet security.	Buy it now! amazon.com
	"Linux Firewalls" Robert L. Ziegler ISBN #0-7357-0900-9, New Riders 11/1999 First edition. (Focus: ipchains) Most complete Linux firewall/security book in publication. Covers ipchains, bind and a complete review of possible firewall configurations. A newer version of this book has been released. See the above book.	Buy it now! amazon.com
	"Red Hat Linux Firewalls" Bill McCarty ISBN #0764524631, John Wiley and Sons Red Hat Press	Buy it now! amazon.com
	"Linux iptables Pocket Reference" by Gregor N. Purdy ISBN #0596005695, O'Reilly; 1 edition (November, 2004)	Buy it now! amazon.com
	"LINUX Routers: A Primer for Network Administrators" by Tony Mancill ISBN #0130090263, Prentice Hall PTR; 2 edition (June 15, 2002)	Buy it now! amazon.com