

Quick HOWTO : Ch14 : Linux Firewalls Using iptables

Visit our forums! 130,000 comments on Linux and many other topics!

Ads by Google
[Router Setup](#)
[Router Config](#)
[Network Setup](#)

From Linux Home Networking

Contents

- 1 Sponsors
- 2 Introduction
- 3 What Is iptables?
- 4 Download And Install The Iptables Package
- 5 How To Start iptables
- 6 Determining The Status of iptables
- 7 Packet Processing In iptables
 - 7.1 Table 14-1 Processing For Packets Routed By The Firewall
 - 7.2 Figure 14-1 Iptables Packet Flow Diagram
- 8 Targets And Jumps
 - 8.1 Table 14-2 Descriptions Of The Most Commonly Used Targets
- 9 Important Iptables Command Switch Operations
 - 9.1 Table 14-2 General Iptables Match Criteria
 - 9.2 Table 14-4 Common TCP and UDP Match Criteria
 - 9.3 Table 14-5 Common ICMP (Ping) Match Criteria
 - 9.4 Table 14-6 Common Extended Match Criteria
- 10 Using User Defined Chains
 - 10.1 Table 14.7 Custom Queues Example Listing
- 11 Saving Your iptables Scripts
- 12 Fedora's iptables Rule Generator
- 13 Recovering From A Lost Script
- 14 Loading Kernel Modules Needed By iptables
- 15 Sample iptables Scripts
 - 15.1 Basic Operating System Defense
 - 15.2 Advanced iptables Initialization
 - 15.3 Allowing DNS Access To Your Firewall
 - 15.4 Allowing WWW And SSH Access To Your Firewall
 - 15.5 Allowing Your Firewall To Access The Internet
 - 15.6 Allow Your Home Network To Access The Firewall
 - 15.7 Masquerading Many To One NAT
 - 15.8 Port Forwarding Type NAT (DHCP/DSL)
 - 15.9 Static NAT
- 16 Troubleshooting iptables
 - 16.1 Checking The Firewall Logs
 - 16.2 iptables Won't Start
- 17 Conclusion

Sponsors

[Serial Port Data Logger](#)
Standalone x86 computer, code in C Up to
serial ports, OEM \$99+
[www.tem.com](#)

[Linux LDAP Server](#)
High performance, advanced security with
replication & mgmt GUIs - Eval
[www.isode.com](#)

[Cisco ASA 5505 Tutorial](#)
Step-by-Step Configuration of Vpn,
Cisco ASA 5505

Introduction

Network security is a primary consideration in any decision to host a website as the threats are becoming more widespread and persistent every day. One means of providing additional protection is to invest in a firewall. Though prices are always falling, in some cases you may be able to create a comparable unit using the Linux iptables package on an existing server for little or no additional expenditure.

This chapter shows how to convert a Linux server into:

- A firewall while simultaneously being your home website's mail, web, and DNS server.
- A router that will use NAT and port forwarding to both protect your home network and have another web server on your home network while sharing the public IP address of your firewall.

Creating an iptables firewall script requires many steps, but with the aid of the sample tutorials, you should be able to complete a configuration relatively quickly.

What Is iptables?

Originally the most popular firewall NAT package running on Linux was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called iptables, giving it such improvements as:

- Better integration with the Linux kernel with the capability of loading iptables-specific kernel modules designed for improved speed and reliability.
- Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and, in certain cases, will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services.
- Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses.
- System logging that provides the option of adjusting the level of detail of the reporting.
- Better network address translation.
- Support for transparent integration with such Web proxy programs as Squid.
- A rate-limiting feature that helps iptables block some types of denial-of-service (DoS) attacks.

Considered a faster and more secure alternative to ipchains, iptables has become the default firewall package installed under Red Hat and Fedora Linux.

Download And Install The Iptables Package

Before you begin, you need to make sure that the iptables software RPM is installed (See Chapter 6, "Installing Linux Software", if you need a refresher.) When searching for the RPMs, remember that the filename usually starts with the software package name by a version number, as in iptables-1.2.9-1.0.i386.rpm.

How To Start iptables

You can start, stop, and restart iptables after booting by using the commands

```
root@bigboy tmp# service iptables start
root@bigboy tmp# service iptables stop
root@bigboy tmp# service iptables restart
```

To get iptables configured to start at boot, use the chkconfig command.

```
root@bigboy tmp# chkconfig iptables on
```

Determining The Status of iptables

You can determine whether iptables is running or not via the service iptables status command. Fedora Core will give a simple status message. For example:

```
root@bigboy tmp# service iptables status
iptables stopped
root@bigboy tmp#
```

Packet Processing In iptables

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation or filtering chain.

There are three tables in total. The first is the mangle table, which is responsible for the alteration of quality of service bits in the TCP header. This is hardly used in a home or SOHO environment.

The second table is the filter queue, which is responsible for packet filtering. It has three built-in chains in which you can place your firewall policy rules. These are the:

- Forward chain: Filters packets to servers protected by the firewall
- Input chain: Filters packets destined for the firewall
- Output chain: Filters packets originating from the firewall

The third table is the nat queue, which is responsible for network address translation. It has two built-in chains; these are:

- Pre-routing chain: NATs packets when the destination address of the packet needs to be changed
- Post-routing chain: NATs packets when the source address of the packet needs to be changed

Table 14-1 Processing For Packets Routed By The Firewall

Queue Type	Queue Function	Packet Transformation Chain in Queue	Chain Function
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall
		INPUT	Filters packets destined to the firewall
		OUTPUT	Filters packets originating from the firewall
Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as destination NAT or DNAT .
		POSTROUTING	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as source NAT , or SNAT .
		OUTPUT	Network address translation for packets generated by the firewall. (Rarely used in SOHO environment)
Mangle	TCP header modification	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	Modification of the TCP packet quality of service bits before routing occurs. (Rarely used in SOHO environment)

You need to specify the table and the chain for each firewall rule you create. There is an exception: Most rules are related to filtering, so iptables assumes that any chain that's defined without an associated table will be a part of the filter table. The filter table is therefore the default.

To help make this clearer, take a look at the way packets are handled by iptables. In Figure 14.1, a TCP packet from the Internet arrives at the firewall's interface on Network A to create a data connection.

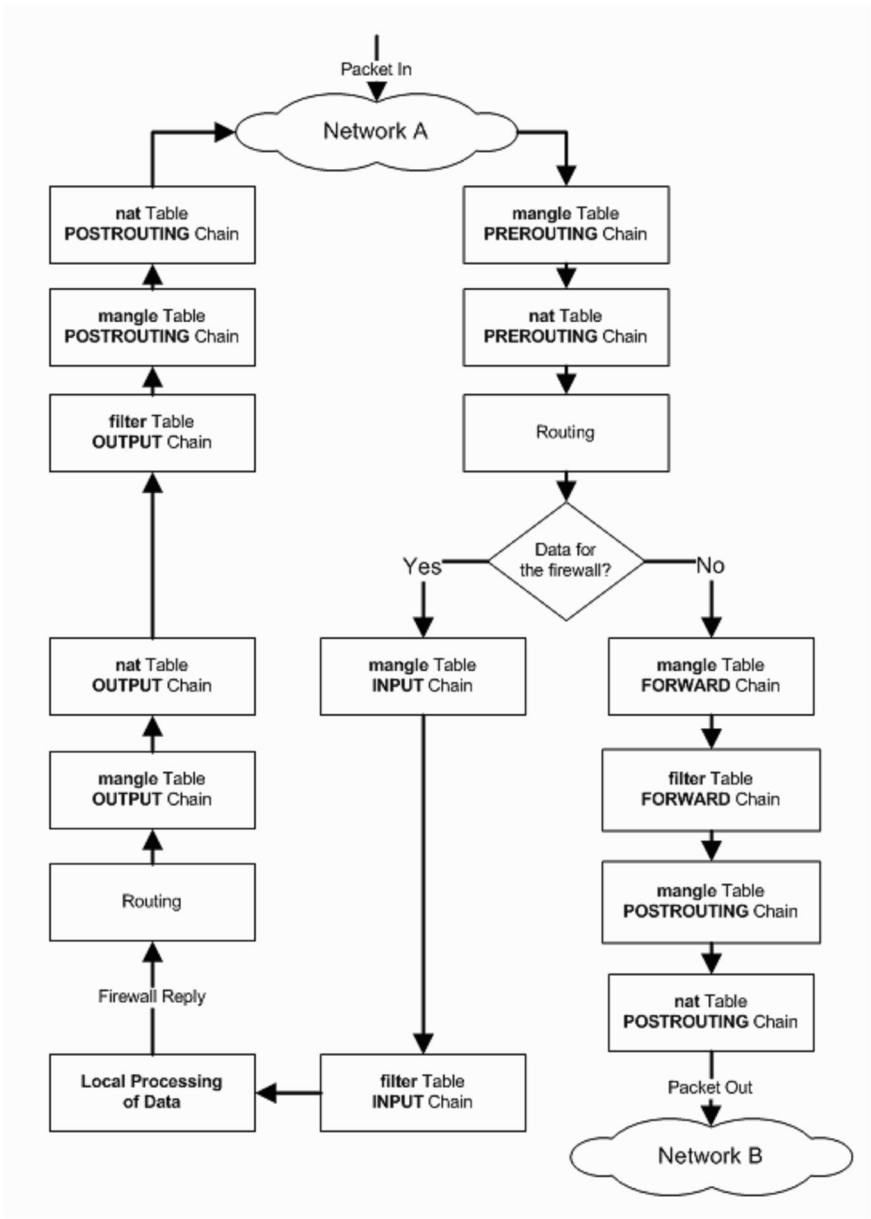
The packet is first examined by your rules in the mangle table's PREROUTING chain, if any. It is then inspected by the rules in the nat table's PREROUTING chain to see whether the packet requires DNAT. It is then routed.

If the packet is destined for a protected network, then it is filtered by the rules in the FORWARD chain of the filter table and, if necessary, the packet undergoes SNAT in the POSTROUTING chain before arriving at Network B. When the destination server decides to reply, the packet undergoes the same sequence of steps. Both the FORWARD and POSTROUTING chains may be configured to implement quality of service (QoS) features in their mangle tables, but this is not usually done in SOHO environments.

If the packet is destined for the firewall itself, then it passes through the mangle table of the INPUT chain, if configured, before being filtered by the rules in the INPUT chain of the filter table. If it successfully passes these tests, then it is processed by the intended application on the firewall.

At some point, the firewall needs to reply. This reply is routed and inspected by the rules in the OUTPUT chain of the mangle table, if any. Next, the rules in the OUTPUT chain of the nat table determine whether DNAT is required, and the rules in the OUTPUT chain of the filter table are then inspected to help restrict unauthorized packets. Finally, before the packet is sent back to the Internet, SNAT and QoS mangling is done by the POSTROUTING chain.

Figure 14-1 Iptables Packet Flow Diagram



It is now time to discuss the ways in which you add rules to these chains.

Targets And Jumps

Each firewall rule inspects each IP packet and then tries to identify it as the target of some sort of operation. Once a target is identified, the packet needs to jump over to it for further processing. Table 14.2 lists the built-in targets that iptables uses.

Table 14-2 Descriptions Of The Most Commonly Used Targets

target	Description	Most Common Options
ACCEPT	<ul style="list-style-type: none">iptables stops further processingThe packet is handed over to the end application or the operating system for processing	N/A
DROP	<ul style="list-style-type: none">iptables stops further processingThe packet is blocked	N/A
LOG	<ul style="list-style-type: none">The packet information is sent to the syslog daemon for loggingiptables continues processing with the next rule in the tableAs you can't log and drop at the same time, it is common to have two similar rules in sequence. The first will log the packet, the second will drop it.	<div>--log-prefix "string"</div> <p>Tells iptables to prefix all log messages with a user-defined string. Frequently used to tell why the logged packet was dropped.</p>
REJECT	<ul style="list-style-type: none">Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked	<div>--reject-with qualifier</div> <p>The qualifier tells what type of reject message is returned. Qualifiers include:</p>

		<div>icmp-port-unreachable(default) icmp-net-unreachable icmp-host-unreachable icmp-prot-unreachable icmp-net-prohibited icmp-host-prohibited tcp-reset echo-reply</div>
DNAT	<div><ul style="list-style-type: none">Used to do destination network address translation ie. rewriting the destination IP address of the packet</div>	<div>--to-destination ipaddress</div> <div>Tells iptables what the destination IP address should be</div>
SNAT	<div><ul style="list-style-type: none">Used to do source network address translation rewriting the source IP address of the packetThe source IP address is user defined</div>	<div>--to-source <address>[:<address>][:<port>-<port>]</div> <div>Specifies the source IP address and ports to be used by SNAT.</div>
MASQUERADE	<div><ul style="list-style-type: none">Used to do Source Network Address TranslationBy default the source IP address is the same as that used by the firewall's interface</div>	<div>--to-ports <port>[-<port>]</div> <div>Specifies the range of source ports to which the original source port can be mapped</div>

Important Iptables Command Switch Operations

Each line of an iptables script not only has a jump, but they also have a number of command line options that are used to append rules to chains that match your defined packet characteristics such as the source IP address and TCP port. There are also options that can be used to just clear a chain so you can start all over again. Tables 14.2 through 14.6 list the most common options.

Table 14-2 General Iptables Match Criteria

iptables command switch	Description
-t <table>	If you don't specify a table, then the <code>filter</code> table is assumed. As discussed before, the possible built-in tables include <code>filter</code> , <code>nat</code> , <code>mangle</code> .
-j <target>	Jump to the specified target chain when the packet matches the current rule.
-A	Append rule to end of a chain
-F	Flush. Deletes all the rules in the selected table.
-p <protocol-type>	Match protocol. Types include <code>icmp</code> , <code>tcp</code> , <code>udp</code> , and <code>all</code> .
-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters
-o <interface-name>	Match "output" interface on which the packet exits

In this command switch example:

```
iptables-A INPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT
```

iptables is being configured to allow the firewall to accept TCP packets coming in on interface `eth0` from any IP address destined for the firewall's IP address of 192.168.1.1. The 0/0 representation of an IP address means any.

Table 14-4 Common TCP and UDP Match Criteria

Switch	Description
-p tcp --sport <port>	TCP source port. Can be a single value or a range in the format: <i>start-port-number</i> <i>end-port-number</i> .
-p tcp --dport <port>	TCP destination port. Can be a single value or a range in the format: <i>starting-port</i> <i>ending-port</i> .
-p tcp --syn	Used to identify a new TCP connection request. <code>! --syn</code> means, not a new connection request.
-p udp --sport <port>	UDP source port. Can be a single value or a range in the format: <i>starting-port</i> <i>ending-port</i> .
-p udp --dport <port>	UDP destination port. Can be a single value or a range in the format: <i>starting-port</i> <i>ending-port</i> .

In this example:

```
iptables-A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP \
--sport 1024:65535 --dport 80 -j ACCEPT
```

iptables is being configured to allow the firewall to accept TCP packets for routing when they enter on interface `eth0` from any IP address and are destined for an IP address of 192.168.1.58 that is reachable via interface `eth1`. The source port is in the range 1024 to 65535 and the destination port is port 80 (www/http).

Table 14-5 Common ICMP (Ping) Match Criteria

Matches used with --icmp-type	Description
--icmp-type <type>	The most commonly used types are <code>echo-reply</code> and <code>echo-request</code> .

In this example:

```
iptables-A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables-A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

iptables is being configured to allow the firewall to send ICMP echo requests (pings) and in turn, accept the expected ICMP echo replies.

Consider another example:

```
iptables-A INPUT -p icmp --icmp-type echo-request \
-m limit --limit 1/s -i eth0 -j ACCEPT
```

The `limit` feature in iptables specifies the maximum average number of matches to allow per second. You can specify time intervals in the format /second, /minute, /hour, or /day, or you can use abbreviations so that 3/second is the same as 3/s.

In this example, ICMP echo requests are restricted to no more than one per second. When tuned correctly, this feature allows you to filter unusually high volumes of traffic that characterize denial of service (DOS) attacks and Internet worms.

```
iptables-A INPUT -p tcp --syn -m limit --limit 5/s -i eth0 -j ACCEPT
```

You can expand on the `limit` feature of iptables to reduce your vulnerability to certain types of denial of service attack. Here a defense for SYN flood attack was created by limiting the acceptance of TCP segments with the SYN bit set to no more than five per second.

Table 14-6 Common Extended Match Criteria

Switch	Description
-m multiport --sports <port, port>	A variety of TCP/UDP source ports separated by commas. Unlike when <code>-m</code> isn't used, they do not have to be within a range.
-m multiport --dports <port, port>	A variety of TCP/UDP destination ports separated by commas. Unlike when <code>-m</code> isn't used, they do not have to be within a range.
-m multiport --ports <port, port>	A variety of TCP/UDP ports separated by commas. Source and destination ports are assumed to be the same and they do not have to be within a range.
-m --state <state>	<p>The most frequently tested states are:</p> <p>ESTABLISHED The packet is part of a connection that has seen packets in both directions.</p> <p>NEW The packet is the start of a new connection.</p> <p>RELATED The packet is starting a new secondary connection. This is a common feature of such protocols such as an FTP data transfer or an ICMP error.</p> <p>INVALID The packet couldn't be identified. Could be due to insufficient system resources or ICMP errors that don't match an existing data flow.</p>

This is an expansion on the previous example:

```
iptables-A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP \
--sport 1024:65535 -m multiport --dports 80,443 -j ACCEPT

iptables-A FORWARD -d 0/0 -o eth0 -s 192.168.1.58 -i eth1 -p TCP \
-m state --state ESTABLISHED -j ACCEPT
```

Here iptables is being configured to allow the firewall to accept TCP packets to be routed when they enter on interface `eth0` from any IP address destined for IP address of 192.168.1.58 that is reachable via interface `eth1`. The source port is in the range 1024 to 65535 and the destination ports are port 80 (www/http) and 443 (https). The return packets from 192.168.1.58 are allowed to be accepted too. Instead of stating the source and destination ports, you can simply allow packets related to established connections using the `-m state` and `--state ESTABLISHED` options.

Using User Defined Chains

As you may remember you can configure iptables to have user-defined chains. This feature is frequently used to help streamline the processing of packets. For example, instead of using a single built-in chain for all protocols, you can use the chain to determine the protocol type for the packet and then hand off the actual final processing to a user-defined, protocol-specific chain in the filter table. In other words, you can replace a long chain with a stubby main chain pointing to multiple stubby chains, thereby shortening the total length of all chains the packet has to pass through. For example:

```
iptables -A INPUT -i eth0 -d 206.229.110.2 -j fast-input-queue
iptables -A OUTPUT -o eth0 -s 206.229.110.2 -j fast-output-queue

iptables -A fast-input-queue -p icmp -j icmp-queue-in
iptables -A fast-output-queue -p icmp -j icmp-queue-out

iptables -A icmp-queue-out -p icmp --icmp-type echo-request \
-m state --state NEW -j ACCEPT

iptables -A icmp-queue-in -p icmp --icmp-type echo-reply -j ACCEPT
```

Here six queues help assist in improving processing speed. Table 14.7 summarizes the function of each.

Table 14.7 Custom Queues Example Listing

Chain	Description
INPUT	The regular built-in INPUT chain in iptables
OUTPUT	The regular built-in OUTPUT chain in iptables
fast-input-queue	Input chain dedicated to identifying specific protocols and shunting the packets to protocol-specific chains
fast-output-queue	Output chain dedicated to identifying specific protocols and shunting the packets to protocol-specific chains
icmp-queue-out	Output queue dedicated to ICMP
icmp-queue-in	Input queue dedicated to ICMP

Saving Your iptables Scripts

The service iptables save command permanently saves the iptables configuration in the /etc/sysconfig/iptables file. When the system reboots, the iptables restore program reads the configuration and makes it the active configuration.

The format of the /etc/sysconfig/iptables file is slightly different from that of the script shown in this chapter. The initialization of built-in chains is automatic, and the string "iptables" is omitted from the rule statements.

Here is a sample /etc/sysconfig/iptables configuration that allows ICMP, IPsec (ESP and AH packets), already established connections, and inbound SSH.

```
root@bigboy tmp# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.2.9 on Mon Nov 8 11:00:07 2004
filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [144:12748]
:RH-Firewall1-INPUT - [0:0]
-A INPUT -j RH-Firewall1-INPUT
-A FORWARD -j RH-Firewall1-INPUT
-A RH-Firewall1-INPUT -i lo -j ACCEPT
-A RH-Firewall1-INPUT -p icmp -m icmp --icmp-type 255 -j ACCEPT
-A RH-Firewall1-INPUT -p esp -j ACCEPT
-A RH-Firewall1-INPUT -p ah -j ACCEPT
-A RH-Firewall1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A RH-Firewall1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A RH-Firewall1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Mon Nov 8 11:00:07 2004
root@bigboy tmp#
```

It is never a good idea to edit this script directly because it is always overwritten by the save command, and it doesn't save any comments at all, which can also make it extremely difficult to follow. For these reasons, you're better off writing and applying a customized script and then using the service iptables save command to make the changes permanent.

Fedora's iptables Rule Generator

Fedora comes with a program called `lokkitt` that you can use to generate a very rudimentary firewall rule set. It prompts for the level of security and then gives you the option of doing simple customizations. It is a good place for beginners to start on a test system so that they can see a general rule structure.

Like the service iptables save command, `lokkitt` saves the firewall rules in a new /etc/sysconfig/iptables file for use on the next reboot.

Once you have become familiar with the iptables syntax, it's best to write scripts that you can comment and then save it to /etc/sysconfig/iptables. It makes them much more manageable and readable.

Recovering From A Lost Script

Sometimes the script you created to generate iptables rules may get corrupted or lost, or you might inherit a new system from an administrator and cannot find the original script used to protect it. In these situations, you can use the iptables save and iptables restore commands to assist you with the continued management of the server.

Unlike the service iptables save command, which actually saves a permanent copy of the firewall's active configuration in the /etc/sysconfig/iptables file, iptables save displays the active configuration to the screen in /etc/sysconfig/iptables format. If you redirect the iptables save screen output to a file with the > symbol, then you can edit the output and reload the updated rules when they meet your new criteria with the iptables restore command.

This example exports the iptables save output to a text file named `firewall.config`.

```
root@bigboy tmp# iptables save > firewall.config
root@bigboy tmp# cat firewall.config
# Generated by iptables-save v1.2.9 on Mon Nov 8 11:00:07 2004
filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [144:12748]
:RH-Firewall1-INPUT - [0:0]
-A INPUT -j RH-Firewall1-INPUT
-A FORWARD -j RH-Firewall1-INPUT
-A RH-Firewall1-INPUT -i lo -j ACCEPT
-A RH-Firewall1-INPUT -p icmp -m icmp --icmp-type 255 -j ACCEPT
-A RH-Firewall1-INPUT -p esp -j ACCEPT
-A RH-Firewall1-INPUT -p ah -j ACCEPT
-A RH-Firewall1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A RH-Firewall1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A RH-Firewall1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Mon Nov 8 11:00:07 2004
root@bigboy tmp#
```

After editing the `firewall.config` file with the commands you need, you can reload it into the active firewall rule set with the iptables restore command.

```
root@bigboy tmp# iptables restore < firewall.config
```

Finally, you should permanently save the active configuration so that it will be loaded automatically when the system reboots:

```
root@bigboy tmp# service iptables save
```

If desired, you can eventually convert this `firewall.config` file into a regular iptables script so that it becomes more easily recognizable and manageable.

Loading Kernel Modules Needed By iptables

The iptables application requires you to load certain kernel modules to activate some of its functions. Whenever any type of NAT is required, the iptable nat module needs to be loaded. The ip_conntrack_ftp module needs to be added for FTP support and should always be loaded with the ip_conntrack module, which tracks TCP connection states. As most scripts probably will keep track of connection states, the ip_conntrack module will be needed in any case. The ip_nat_ftp module also needs to be loaded for FTP servers behind a NAT firewall.

Unfortunately, the /etc/sysconfig/iptables file doesn't support the loading of modules, so you'll have to add the statements to your /etc/rc.local file, which is run at the end of every reboot.

The script samples in this chapter include these statements only as a reminder to place them in the /etc/rc.local file.

```
# File: /etc/rc.local
# Module to track the state of connections
modprobe ip_conntrack

# Load the iptables active FTP module, requires ip_conntrack
modprobe ip_conntrack_ftp

# Load iptables NAT module when required
modprobe iptable_nat

# Module required for active an FTP server using NAT
modprobe ip_nat_ftp
```

Sample iptables Scripts

This section provides some sample scripts you can use to get iptables working for you. Pay special attention to the logging example at the end.

The basic initialization script snippets should also be included in all your scripts to ensure the correct initialization of your chains should you decide to restart your script after startup. This chapter also includes other snippets that will help you get basic functionality. It should be a good guide to get you started.

Note: Once you feel more confident, you can use Appendix II "Codes, Scripts, and Configuration", to find detailed scripts. The appendix shows you how to allow your firewall to:

- Be used as a Linux Web, mail, and DNS server
- Be the NAT router for your home network
- Prevent various types of attacks using corrupted TCP, UDP, and ICMP packets
- Provide outbound passive FTP access from the firewall

There are also simpler code snippets in the Appendix II "Codes, Scripts, and Configuration", for inbound and outbound FTP connections to and from your firewall.

Basic Operating System Defense

You can do several things before employing your firewall script to improve the resilience of your firewall to attack. For example, the Linux operating system has a number of built-in protection mechanisms that you should activate by modifying the system kernel parameters in the /proc filesystem via the /etc/sysctl.conf file. Using of /etc/sysctl.conf to modify kernel parameters is explained in more detail in , Appendix I "Miscellaneous Linux Topics".

Here is a sample configuration:

```
# File: /etc/sysctl.conf
#-----
# Disables routing triangulation responses to queries out
```

```
# the same interface not another. Helps to maintain state
# Also protects against IP spoofing
-----
net/ipv4/conf/all/rp_filter = 1
-----
#
# Enable logging of packets with malformed IP addresses
#
-----
net/ipv4/conf/all/log_martians = 1
-----
#
# Disabled redirects
#
-----
net/ipv4/conf/all/send_redirects = 0
-----
#
# Disable source routed packets
#
-----
net/ipv4/conf/all/accept_source_route = 0
-----
#
# Disable acceptance of ICMP redirects
#
-----
net/ipv4/conf/all/accept_redirects = 0
-----
#
# Turn on protection from Denial of Service (DOS) attacks
#
-----
net/ipv4/tcp_syncookies = 1
-----
#
# Disable responding to ping broadcasts
#
-----
net/ipv4/icmp_echo_ignore_broadcasts = 1
-----
#
# Enable IP routing. Required if your firewall is protecting a
# network. NAT included.
#
-----
net/ipv4/ip_forward = 1
```

Advanced iptables Initialization

You may also want to add some more advanced initialization steps to your script, including checks for Internet traffic from RFC1918 private addresses. The sample script snippet below outlines how to do this. More complex initialization would include checks for attacks using invalid TCP flags and directed broadcasts which are beyond the scope of this book.

The script also uses multiple user-defined chains to make the script shorter and faster as the chains can be repeatedly accessed. This removes the need to repeat the same statements over and over again.

You can take even more precautions to further protect your network. The complete firewall script in Appendix II "Codes, Scripts and Configuration", outlines most of them.

```
#####
#
# Define networks. NOTE! You may want to put these "EXTERNAL"
# definitions at the top of your script.
#
#####
EXTERNAL_INT="eth0"      # External Internet interface
EXTERNAL_IP="97.158.253.25" # Internet interface IP address
#
# Initialize our user-defined chains
#
-----
iptables -N valid_src
iptables -N valid_dst
#
# Verify valid source and destination addresses for all packets
#
-----
iptables -A INPUT -i $EXTERNAL_INT -j valid_src
iptables -A FORWARD -i $EXTERNAL_INT -j valid_src
iptables -A OUTPUT -o $EXTERNAL_INT -j valid_dst
iptables -A FORWARD -o $EXTERNAL_INT -j valid_dst
#####
#
# Source and Destination Address Sanity Checks
#
# Drop packets from networks covered in RFC 1918 (private nets)
# Drop packets from external interface IP
#
#####
iptables -A valid_src -s 10.0.0.0/8 -j DROP
iptables -A valid_src -s 172.16.0.0/12 -j DROP
iptables -A valid_src -s 192.168.0.0/16 -j DROP
iptables -A valid_src -s 224.0.0.0/4 -j DROP
iptables -A valid_src -s 240.0.0.0/5 -j DROP
iptables -A valid_src -s 127.0.0.0/8 -j DROP
iptables -A valid_src -s 0.0.0.0/8 -j DROP
iptables -A valid_src -d 255.255.255.255 -j DROP
iptables -A valid_src -s 169.254.0.0/16 -j DROP
iptables -A valid_src -s $EXTERNAL_IP -j DROP
iptables -A valid_dst -d 224.0.0.0/4 -j DROP
```

Allowing DNS Access To Your Firewall

You'll almost certainly want your firewall to make DNS queries to the Internet. This is not because it is required for the basic functionality of the firewall, but because of Fedora Linux's yum RPM updater which will help to keep the server up to date with the latest security patches. The following statements will apply not only for firewalls acting as DNS clients but also for firewalls working in a caching or regular DNS server role.

```
-----
#
# Allow outbound DNS queries from the FW and the reply too
#
# - Interface eth0 is the internet interface
#
# Zone transfers use TCP and not UDP. Most home networks
# / websites using a single DNS server won't require TCP statements
#
-----
iptables -A OUTPUT -p udp -o eth0 --dport 53 --sport 1024:65535 \
-j ACCEPT
iptables -A INPUT -p udp -i eth0 --sport 53 --dport 1024:65535 \
-j ACCEPT
```

Allowing WWW And SSH Access To Your Firewall

This sample snippet is for a firewall that doubles as a web server that is managed remotely by its system administrator via secure shell (SSH) sessions. Inbound packets destined for ports 80 and 22 are allowed there by making the first steps in establishing a connection. It isn't necessary to specify these ports for the return leg as outbound packets for all established connections are allowed. Connections initiated by persons logged into the Web server will be denied as outbound NEW connection packets aren't allowed.

```
-----
#
# Allow previously established connections
#
# - Interface eth0 is the internet interface
#
-----
iptables -A OUTPUT -o eth0 -m state --state ESTABLISHED,RELATED \
-j ACCEPT
#
# Allow port 80 (www) and 22 (SSH) connections to the firewall
#
-----
iptables -A INPUT -p tcp -i eth0 --dport 22 --sport 1024:65535 \
-m state --state NEW -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 80 --sport 1024:65535 \
-m state --state NEW -j ACCEPT
```

Allowing Your Firewall To Access The Internet

This iptables script enables a user on the firewall to use a Web browser to surf the Internet. HTTP traffic uses TCP port 80, and HTTPS uses port 443.

Note: HTTPS (secure HTTP) is used for credit card transactions frequently as well as by Red Hat Linux servers running update. FTP and HTTP are frequently used with yum.

```
-----
#
# Allow port 80 (www) and 443 (https) connections from the firewall
#
-----
iptables -A OUTPUT -j ACCEPT -m state \
--state NEW,ESTABLISHED,RELATED -o eth0 -p tcp \
-m multiport --dports 80,443 --sport 1024:65535
#
# Allow previously established connections
#
# - Interface eth0 is the internet interface
#
-----
iptables -A INPUT -j ACCEPT -m state --state ESTABLISHED,RELATED \
-i eth0 -p tcp
```

If you want all TCP traffic originating from the firewall to be accepted, then remove the line:

```
-m multiport --dports 80,443 --sport 1024:65535
```

Allow Your Home Network To Access The Firewall

In this example, eth1 is directly connected to a home network using IP addresses from the 192.168.1.0 network. All traffic between this network and the firewall is simplistically assumed to be trusted and allowed.

Further rules will be needed for the interface connected to the Internet to allow only specific ports, types of connections and possibly even remote servers to have access to your firewall and home network.

```
-----
#
# Allow all bidirectional traffic from your firewall to the
# protected network
#
# - Interface eth1 is the private network interface
#
-----
iptables -A INPUT -j ACCEPT -p all -s 192.168.1.0/24 -i eth1
iptables -A OUTPUT -j ACCEPT -p all -d 192.168.1.0/24 -o eth1
```

Masquerading (Many to One NAT)

As explained in Chapter 2, "Introduction to Networking", masquerading is another name for what many call many to one NAT. In other words, traffic from all devices on one or more protected networks will appear as if it originated from a single IP address on the Internet side of the firewall.

Note: The masquerade IP address always defaults to the IP address of the firewall's main interface. The advantage of this is that you never have to specify the NAT IP address. This makes it much easier to configure iptables NAT with DHCP.

You can configure many to one NAT to an IP alias using the POSTROUTING chain and not the MASQUERADE statement. An example of this can be seen in the static NAT section that follows.

Keep in mind that iptables requires the iptables_nat module to be loaded with the modprobe command for the masquerade feature to work. Masquerading also depends on the Linux operating system being configured to support routing between the Internet and private network interfaces of the firewall. This is done by enabling IP forwarding or routing by giving the file /proc/sys/net/ipv4/ip_forward the value 1 as opposed to the default disabled value of 0.

Once masquerading has been achieved using the POSTROUTING chain of the nat table, you will have to configure iptables to allow packets to flow between the two interfaces. To do this, use the FORWARD chain of the filter table. More specifically, packets related to NEW and ESTABLISHED connections will be allowed outbound to the Internet, but only packets related to ESTABLISHED connections will be allowed inbound. This helps to protect the home network from anyone trying to initiate connections from the Internet.

```
-----
# Load the NAT module
#
# Note: It is best to use the /etc/rc.local example in this
# chapter. This value will not be retained in the
# /etc/sysconfig/iptables file. Included only as a reminder
#
-----
modprobe iptable_nat
-----
# Enable routing by modifying the ip_forward /proc filesystem file
#
# Note: It is best to use the /etc/sysctl.conf example in this
# chapter. This value will not be retained in the
# /etc/sysconfig/iptables file. Included only as a reminder
#
-----
echo 1 > /proc/sys/net/ipv4/ip_forward
-----
# Allow masquerading
# - Interface eth0 is the Internet interface
# - Interface eth1 is the private network interface
#
-----
iptables -A POSTROUTING -t nat -o eth0 -s 192.168.1.0/24 -d 0/0 \
-j MASQUERADE
-----
# Prior to masquerading, the packets are routed via the filter
# table's FORWARD chain
# Allowed outbound: New, established and related connections
# Allowed inbound: Established and related connections
#
-----
iptables -A FORWARD -t filter -o eth0 -m state \
--state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -t filter -i eth0 -m state \
--state ESTABLISHED,RELATED -j ACCEPT
-----
```

Note: If you configure your firewall to do masquerading then it should be used as the default gateway for all your servers on the network.

Port Forwarding Type NAT (DHCP DSL)

In many cases, servers may get a single DHCP public IP address from their ISPs. If a Linux firewall is also your DHCP server, then you will have to use port forwarding. Here the combination of the firewall's single IP address, the remote server's IP address, and the source/destination port of the traffic can be used to uniquely identify a traffic flow. All traffic that matches a particular combination of these factors may then be forwarded to a single server on the private network.

Port forwarding is handled by the PREROUTING chain of the nat table. As in masquerading, the iptables_nat module has to be loaded and routing has to be enabled for port forwarding to work. Routing too must be allowed in iptables with the FORWARD chain; this includes all NEW inbound connections from the Internet matching the port forwarding port plus all future packets related to the ESTABLISHED connection in both directions.

```
-----
# Load the NAT module
#
# Note: It is best to use the /etc/rc.local example in this
# chapter. This value will not be retained in the
# /etc/sysconfig/iptables file. Included only as a reminder
#
-----
modprobe iptable_nat
-----
# Get the IP address of the Internet interface eth0 (Linux only)
#
# You'll have to use a different expression to get the IP address
# for other operating systems which have a different configuration
# or enter the IP address manually in the PREROUTING statement
#
# This is best when your firewall gets its IP address using DHCP.
# The external IP address could just be hard-coded ("typed in
# normally")
#
-----
external_int="eth0"
external_ip=""
ifconfig $external_int | grep 'inet addr' | \
awk '{print $2}' | sed -e 's/.*://'
-----
# Enable routing by modifying the ip_forward /proc filesystem file
#
# Note: It is best to use the /etc/sysctl.conf example in this
# chapter. This value will not be retained in the
# /etc/sysconfig/iptables file. Included only as a reminder
#
-----
echo 1 > /proc/sys/net/ipv4/ip_forward
-----
# Allow port forwarding for traffic destined to port 80 of the
# firewall's IP address to be forwarded to port 8080 on server
# 192.168.1.200
# - Interface eth0 is the Internet interface
# - Interface eth1 is the private network interface
#
-----
iptables -t nat -A PREROUTING -p tcp -i eth0 -d $external_ip \
--dport 80 --sport 1024:65535 -j DNAT --to 192.168.1.200:8080
-----
# After DNAT, the packets are routed via the filter table's
# FORWARD chain
# Connection on port 80 to the target machine on the private
# network must be allowed
#
-----
iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.200 \
--dport 8080 --sport 1024:65535 -m state --state NEW -j ACCEPT
iptables -A FORWARD -t filter -i eth0 -m state \
--state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -t filter -i eth0 -m state \
--state ESTABLISHED,RELATED -j ACCEPT
-----
```

Static NAT

In this example, all traffic to a particular public IP address, not just to a particular port, is translated to a single server on the protected subnet. Because the firewall has more than one IP address, I can't recommend MASQUERADE; it will force masquerading as the IP address of the primary interface and not as any of the alias IP addresses the firewall may have. Instead, use SNAT to specify the alias IP address to be used for connections initiated by all other servers in the protected network.

Note: Although the nat table NATs all traffic to the target servers (192.168.1.100 to 102), only connections on ports 80, 443 and 22 are allowed through by the FORWARD chain. Also notice how you have to specify a separate -m multiport option whenever you need to match multiple non-sequential ports for both source and destination.

In this example, the firewall

- Uses one to one NAT to make the server 192.168.1.100 on your home network appear on the Internet as IP addresses 97.158.253.26.
- Creates a many to one NAT for the 192.168.1.0 home network in which all the servers appear on the Internet as IP address 97.158.253.29. This is different from masquerading.

You will have to create alias IP addresses for each of these Internet IPs for one to one NAT to work.

```
-----
# Load the NAT module
#
# Note: It is best to use the /etc/rc.local example in this
# chapter. This value will not be retained in the
# /etc/sysconfig/iptables file. Included only as a reminder
#
-----
modprobe iptable_nat
-----
# Enable routing by modifying the ip_forward /proc filesystem file
#
# Note: It is best to use the /etc/sysctl.conf example in this
# chapter. This value will not be retained in the
# /etc/sysconfig/iptables file. Included only as a reminder
#
-----
echo 1 > /proc/sys/net/ipv4/ip_forward
-----
# NAT ALL traffic
# #####
# REMEMBER to create aliases for all the Internet IP addresses below
# #####
#
# TO:          FROM:          MAP TO SERVER
# 97.158.253.26 Anywhere          192.168.1.100 (1:1 NAT - Inbound)
# Anywhere     192.168.1.100  97.158.253.26 (1:1 NAT - Outbound)
# Anywhere     192.168.1.0/24  97.158.253.29 (FW IP)
#
# SNAT is used to NAT all other outbound connections initiated
# from the protected network to appear to come from
# IP address 97.158.253.29
#
# POSTROUTING
# NATs source IP addresses. Frequently used to NAT connections from
# your home network to the Internet
#
# PREROUTING
# NATs destination IP addresses. Frequently used to NAT
# connections from the Internet to your home network
#
# - Interface eth0 is the Internet interface
# - Interface eth1 is the private network interface
#
-----
# PREROUTING statements for 1:1 NAT
# (Connections originating from the Internet)
iptables -t nat -A PREROUTING -d 97.158.253.26 -i eth0 \
-j DNAT --to-destination 192.168.1.100
#
# POSTROUTING statements for 1:1 NAT
# (Connections originating from the home network server)
iptables -t nat -A POSTROUTING -s 192.168.1.100 -o eth0 \
-j SNAT --to-source 97.158.253.26
#
# POSTROUTING statements for Many:1 NAT
# (Connections originating from the entire home network)
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 \
-j SNAT -o eth0 --to-source 97.158.253.29
-----
```

```
# Allow forwarding to each of the servers configured for 1:1 NAT
# (For connections originating from the Internet. Notice how you
# use the real IP addresses here.)

iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.100 \
-m multiport --dports 80,443,22 \
-m state --state NEW -j ACCEPT

# Allow forwarding for all New and Established SNAT connections
# originating on the home network AND already established
# DNAT connections

iptables -A FORWARD -t filter -o eth0 -m state \
--state NEW,ESTABLISHED,RELATED -j ACCEPT

# Allow forwarding for all 1:1 NAT connections originating on
# the Internet that have already passed through the NEW forwarding
# statements above

iptables -A FORWARD -t filter -i eth0 -m state \
--state ESTABLISHED,RELATED -j ACCEPT
```

Troubleshooting iptables

A number of tools are at your disposal for troubleshooting iptables firewall scripts. One of the best methods is to log all dropped packets to the `/var/log/messages` file.

Checking The Firewall Logs

You track packets passing through the iptables list of rules using the LOG target. You should be aware that the LOG target

- Logs all traffic that matches the iptables rule in which it is located.
- Automatically writes an entry to the `/var/log/messages` file and then executes the next rule.

If you want to log only unwanted traffic, therefore, you have to add a matching rule with a DROP target immediately after the LOG rule. If you don't, you'll find yourself logging both desired and unwanted traffic with no way of discerning between the two, because by default iptables doesn't state why the packet was logged in its log message.

This example logs a summary of failed packets to the file `/var/log/messages`. You can use the contents of this file to determine which TCP/UDP ports you need to open to provide access to specific traffic that is currently stopped.

```
-----
# Log and drop all other packets to file /var/log/messages
# Without this we could be crawling around in the dark
-----

iptables -A OUTPUT -j LOG
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG

iptables -A OUTPUT -j DROP
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
```

Here are some examples of the output of this file:

- Firewall denies reply to DNS queries (UDP port 53) destined to server 192.168.1.102 on the home network

```
Feb 23 20:33:50 bigboy kernel: IN=wlan0 OUT= MAO=00:06:25:09:69:80:a0:c5:e1:3e:88:08:00 SRC=192.42.93.30 DST=192.168.1.102 LEN=220 TOS=0x00 PREO=0x00 TTL=54 ID=30485 PROTO=UDP SPT=53 DPT=32820 LEN=200
```

- Firewall denies Windows NetBIOS traffic (UDP port 138)

```
Feb 23 20:43:08 bigboy kernel: IN=wlan0 OUT= MAO=ff:ff:ff:ff:ff:ff:00:06:25:09:6a:b5:08:00 SRC=192.168.1.100 DST=192.168.1.255 LEN=241 TOS=0x00 PREO=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=138 DPT=138 LEN=221
```

- Firewall denies Network Time Protocol (NTP UDP port 123)

```
Feb 23 20:58:48 bigboy kernel: IN= OUT=wlan0 SRC=192.168.1.102 DST=207.200.81.113 LEN=76 TOS=0x10 PREO=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=123 DPT=123 LEN=56
```

The traffic in all these examples isn't destined for the firewall. Therefore, you should check your INPUT, OUTPUT, FORWARD and NAT related statements. If the firewall's IP address is involved then you should focus on the INPUT and OUTPUT statements.

If nothing shows up in the logs, then follow the steps in Chapter 4, "Simple Network Troubleshooting", to determine whether the data is reaching your firewall at all and, if it is not, the location on your network that could be causing the problem.

As a general rule, you won't be able to access the public NAT IP addresses from servers on your home network. Basic NAT testing requires you to ask a friend to try to connect to your home network from the Internet.

You can then use the logging output in `/var/log/messages` to make sure that the translations are occurring correctly and iptables isn't dropping the packets after translation occurs.

iptables Won't Start

The iptables startup script expects to find the `/etc/sysconfig/iptables` before it starts. If none exists, then symptoms include the firewall status always being stopped and the `/etc/init.d/iptables` script running without the typical [OK] or [FAILED] messages.

If you have just installed iptables and have never applied a policy, then you will face this problem. Unfortunately running the `service iptables save` command before restarting won't help either. You have to create this file:

```
[root@bigboy tmp]# service iptables start
[root@bigboy tmp]#


[root@bigboy tmp]# touch /etc/sysconfig/iptables
[root@bigboy tmp]# chmod 600 /etc/sysconfig/iptables

[root@bigboy tmp]# service iptables start
Applying iptables firewall rules: [ OK ]
[root@bigboy tmp]#
```

Conclusion

A firewall is a critical part of any establishment that connects to an unprotected network such as the Internet, but a firewall is never sufficient. Web site security involves not just protection from corrupted packets or maliciously overwhelming volumes of traffic, but also involves daily data backups to help recovery from device failures, regular application patching, enforced password policies, restricted and monitored physical access to your servers, reliable power and cooling, secured cabling, redundant hardware, and, probably most importantly, well-trained and motivated employees. Security should be viewed as anything that contributes to the desired risk-free functioning of your site, and it is well worth the money to invest in and learn from a book that specializes in the topic.

Retrieved from "http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_-_Ch14_-_Linux_Firewalls_Using_iptables"



Ads by Google

[Network Setup](#)


Amazon.com



Tivox STB-T8 Digital to Analog TV Converter

Tivox

New \$49.99




Glenn Beck's Common Sense

Glenn Beck

New \$6.59

Best \$5.82



The Tivoli Theatre

Audrey

New \$

Best \$

- This page was last modified on 19 May 2009, at 19:33.
- Content is available under Attribution-NonCommercial-NoDerivs 2.5 .

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_-_Ch14_-_Linux_Firewalls_Using_iptables

30.07.2009 22:43:30