IP TABLES

A Beginner’s Tutorial

Author	**:**	Tony Hill
Date	**:**	24th March 2013
Version	**:**	v1-3
Linux OS	**:**	Ubuntu 12.04.2 LTS (precise)
Kernel	**:**	v3.2.0-39
Iptables	**:**	v1.4.12

INDEX

1 Introduction

This paper shows how to use iptables to set up a secure firewall on your Linux home computer(s). It contains plenty of configuration examples and command output. If you follow the examples you will be able to build and deploy a robust and flexible firewall of your own.

Having configured the firewall there are instructions on how to create a script to start it automatically at boot-time using the *"/etc/init.d update-rc"* mechanism.

2 Firewall Overview

Essentially, there are two types of firewall - external and internal. Corporate firewalls are usually dedicated external devices with complex rule-sets, whereas internal (personal) firewalls run on your computer and are generally much simpler to configure.

The basic job of both types of firewall is the same – an external firewall prevents unwanted "outside" traffic from entering your network whereas an internal firewall prevents unwanted "inside" traffic from entering your computer; together with any "outside" traffic that the external firewall may have allowed through either deliberately or by misconfiguration.

An underlying principle of all firewalls is as follows:

The outbound traffic that you generate is good because you sent it so you know what it is. Inbound responses to that traffic must, for the most part, also be good. Unsolicited inbound traffic may be bad and should be stopped!

Clearly there are some exceptions to this principle but I am assuming that it holds true for the purpose of this tutorial.

Most corporate firewalls spend their day trying to detect and prevent Denial of Service attacks. They not only enforce connection rules but also look out for anomalous protocol behaviour and use deep packet inspection to find virus signatures and other naughty code. Iptables is very good at the connection rules thing, but is not a virus scanner or deep packet inspection tool.

Note: *Fire-walling and virus protection are two distinct functions. A personal firewall per se does not protect against viruses, although most commercially available personal firewall packages are accompanied by a virus scanner of some description.*

Firewalls correlate related outbound and inbound traffic into "flows". Related traffic is any bi-directional traffic stream that has corresponding source and destination IP addresses, protocol types, source and destination port numbers and, in the case of TCP, sequence numbers and acknowledgements. The firewall maintains a connection table to track the state of each flow and check for correct protocol behaviour. Firewalls that maintain connection tables are referred to as "stateful" firewalls. Iptables is a stateful firewall.

3 Iptables Overview

Iptables is a suite of powerful directives that hook into the Linux kernel at various stages of the packet processing lifecycle. Figure-1 below shows where iptables sits in relation to the kernel and at which points the hooks into the kernel are provisioned.

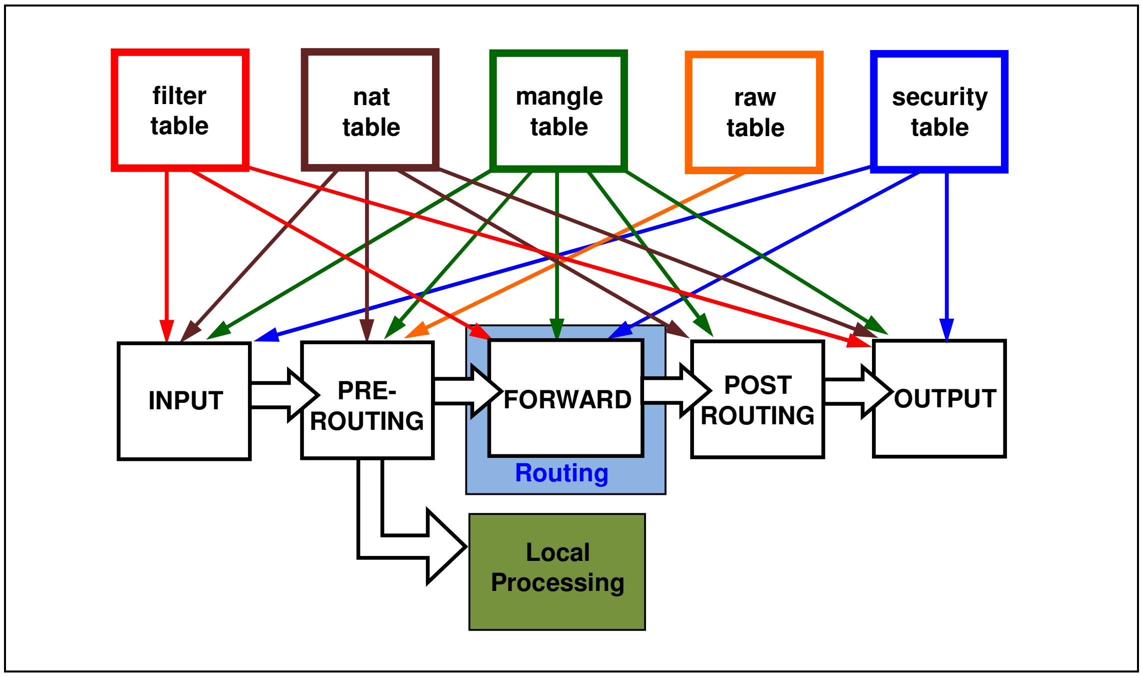

Figure-1: How Iptables Hooks into the Kernel

Iptables is used to create and manage rules that provide, amongst other things, packet manipulation, connection tracking, NAT, ToS / DSCP matching and re-writing, and connection rate-limiting.

Netfilter is the Linux kernel’s native packet filtering subsystem which is not available to the user other than through system primitives. Iptables provides a user interface to configure the netfilter subsystem. Most third party Linux firewalls that you download, and install, such as UFW and Firewall Builder, are simply front-ends to iptables. Understanding how to configure iptables natively allows you to implement more granular and comprehensive packet filtering and manipulation policies than any of the third party applications.

4 Iptables Structure and Terminology

Iptables allows an administrator to populate tables with chains of rules that manipulate packets at different stages of the kernel's packet processing life-cycle.

Each table has a distinct function. For example, the ***filter table*** (the default table) provides commands to filter and accept or drop packets, the ***NAT table*** provides commands to translate (modify) source or destination IP addresses, and the ***mangle table*** provides commands to modify packet headers.

Each table contains entities called “chains” under which specific packet rules (policies) are configured. For example, the *filter table* contains built-in chains called INPUT, FORWARD and OUTPUT. A packet drop rule configured underneath the INPUT chain directs the kernel to DROP packets that are received on a particular interface.

Table-1 below lists the “iptables tables”, their function and the built-in chains they contain. This tutorial focuses predominantly on the *filter table* but the principles apply equally well to all of the tables in the iptables subsystem.

Table	Function	Chain
filter (default)	Packet filtering / firewall	**INPUT**
		FORWARD
		OUTPUT
NAT	Network Address Translation	**PREROUTING**
		INPUT
		OUTPUT
		POSTROUTING
mangle	Packet modification	**PREROUTING**
		INPUT
		FORWARD
		OUTPUT
		POSTROUTING
security	Mandatory Access Control	**INPUT**
		FORWARD
		OUTPUT
raw	Bypass “**conntrack**” for corner cases	**PREROUTING**
		OUTPUT

Table-1: Iptables Tables & Chains

Figure-2 below shows another representation of the stages at which the various rule chains hook into the kernel packet processing subsystem together with the tables with which the chains are associated. This diagram depicts two types of packet flow:

– Packets entering interfaces one and two that terminate in an application within the computer (local packets). The application returns these packets to the interface over which they arrived if the application issues a response to the sender.

– Packets entering interface one are forwarded direct to interface two, and vice-versa. These packets are not handed over for local processing. The computer is set up to forward packets, which are simply routed through the computer.