Training: Applying firewall rules with iptables

Created by Pedro Geraldo on [2019-05-29](190529-training-iptables-firewall-rules.docx)

Table of Contents

[Introduction 1](#_Toc11682233)

[Prerequisites 1](#_Toc11682234)

[Packet Filter 1](#_Toc11682235)

[Advantages 1](#_Toc11682236)

[Control: 1](#_Toc11682237)

[Security: 1](#_Toc11682238)

[Watchfulness: 1](#_Toc11682239)

[Firewall Chains 1](#_Toc11682240)

[Using iptables 1](#_Toc11682241)

[Validation 1](#_Toc11682242)

[References 1](#_Toc11682243)

[End of document 1](#_Toc11682244)

# Introduction

The iptables tool allows us to insert and delete rules from the (Unix) kernel's packet filtering table. This means that whatever you configure, it will be lost upon reboot (there is a way of making them permanent)! Note that iptables will be replaced by nftables as of <<Debian Buster>> (current one is <<Debian Stretch>>).

## Prerequisites

Make sure to have followed the [Training: Connexion to servers](https://docs.switzernet.com/3/support/101014-training-connection-sip/).

# Packet Filter

A packet filter is a piece of software which looks at the header of packets as they pass through and decides the fate of the entire packet. It might decide to DROP the packet (i.e., discard the packet as if it had never received it), ACCEPT the packet (i.e., let the packet go through), or something more complicated.

## Advantages

### Control:

When you are using a Linux box to connect your internal network to another network (say, the Internet) you have an opportunity to allow certain types of traffic and disallow others. For example, the header of a packet contains the destination address of the packet, so you can prevent packets going to a certain part of the outside network.

### Security:

When your Linux box is the only thing between the chaos of the Internet and your nice, orderly network, it's nice to know you can restrict what comes tromping in your door. For example, you might allow anything to go out from your network, but you might be worried about the well-known `Ping of Death' coming in from malicious outsiders. As another example, you might not want outsiders telnetting to your Linux box, even though all your accounts have passwords. Maybe you want (like most people) to be an observer on the Internet, and not a server (willing or otherwise). Simply don't let anyone connect in, by having the packet filter reject incoming packets used to set up connections.

### Watchfulness:

Sometimes a badly configured machine on the local network will decide to spew packets to the outside world. It's nice to tell the packet filter to let you know if anything abnormal occurs; maybe you can do something about it, or maybe you're just curious by nature.

## Firewall Chains

The kernel starts with three lists of rules in the “filter” table. These lists are called **firewall chains** or just **chains**. The three chains are called **INPUT**, **OUTPUT** and **FORWARD**.

Incoming

Routing Decision

**FORWARD**

Outgoing

**INPUT**

Local Process

**OUTPUT**

Figure 1 - Representation of how the chains are arranged

A chain is a checklist of rules. Each rule says, “if the packet header looks like this, then here's what to do with the packet”. If the rule doesn't match the packet, then the next rule in the chain is consulted. Finally, if there are no more rules to consult, then the kernel looks at the chain policy to decide what to do. In a security-conscious system, this policy usually tells the kernel to DROP the packet.

## Using iptables

For now, we will focus on configuring the default chains. In a chain we can add, replace, delete and list the rules mentioned previously. The commands are as follows:

**iptables** [**-t** table] {**-A**|**-C**|**-D**} chain rule-specification target

**iptables** [**-t** table] **-I** chain [rulenum] rule-specification target

**iptables** [**-t** table] **-R** chain rulenum rule-specification target

**iptables** [**-t** table] **-D** chain rulenum

**iptables** [**-t** table] **-S** [chain [rulenum]]

**iptables** [**-t** table] {**-F**|**-L**|**-Z**} [chain [rulenum]] [options...]

What do these mean?

### TABLES

 To specify the packet matching table which the command should operate on, we invoke the following option:

{-**t**|--**table**} *table*

If the kernel is configured with automatic module loading, an attempt will be made to load the appropriate module for that table if it is not already there

There are currently five independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

* **filter**:
	+ This is the default table (if no -t option is passed).
	+ It contains the built-in chains **INPUT** (for packets destined to local sockets), **FORWARD** (for packets being routed through the box), and **OUTPUT** (for locally-generated packets).
* **nat**:
	+ This table is consulted when a packet that creates a new connection is encountered.
	+ It consists of four built-in chains: **PREROUTING** (for altering packets as soon as they come in), **INPUT** (for altering packets destined for local sockets), **OUTPUT** (for altering locally-generated packets before routing), and **POSTROUTING** (for altering packets as they are about to go out).
* **mangle**:
	+ This table is used for specialized packet alteration.
	+ Until kernel 2.4.17 it had two built-in chains: **PREROUTING** (for altering incoming packets before routing) and **OUTPUT** (for altering locally-generated packets before routing).
	+ Since kernel 2.4.18, three other built-in chains are also supported: **INPUT** (for packets coming into the box itself), **FORWARD** (for altering packets being routed through the box), and **POSTROUTING** (for altering packets as they are about to go out).
* **raw**:
	+ This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target.
	+ It registers at the netfilter hooks with higher priority and is thus called before ip\_conntrack, or any other IP tables.
	+ It provides the following built-in chains: **PREROUTING** (for packets arriving via any network interface) **OUTPUT** (for packets generated by local processes)
* **security**:
	+ This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the **SECMARK** and **CONNSECMARK** targets.
	+ Mandatory Access Control is implemented by Linux Security Modules such as SELinux.
	+ The security table is called after the filter table, allowing any Discretionary Access Control (DAC) rules in the filter table to take effect before MAC rules.
	+ This table provides the following built-in chains: **INPUT** (for packets coming into the box itself), **OUTPUT** (for altering locally-generated packets before routing), and **FORWARD** (for altering packets being routed through the box).

### Action Commands

These options specify the desired action to perform. Only one of them can be specified on the command line unless otherwise stated below. For long versions of the command and option names, you need to use only enough letters to ensure that **iptables** can differentiate it from all other options.

* **-A, --append** *chain rule-specification*
	+ Append one or more rules to the end of the selected chain. When the source and/or destination names resolve to more than one address, a rule will be added for each possible address combination.
* **-D, --delete** *chain rule-specification*
* **-I, --insert** *chain* [*rulenum*] *rule-specification*
	+ Insert one or more rules in the selected chain as the given rule number. So, if the rule number is 1, the rule or rules are inserted at the head of the chain. This is also the default if no rule number is specified.
* **-R, --replace** *chain rulenum rule-specification*
	+ Replace a rule in the selected chain. If the source and/or destination names resolve to multiple addresses, the command will fail. Rules are numbered starting at 1.
* **-D, --delete** *chain rulenum*
	+ Delete one or more rules from the selected chain. There are two versions of this command: the rule can be specified as a number in the chain (starting at 1 for the first rule) or a rule to match.
* **-L, --list** [*chain*]
	+ List all rules in the selected chain. If no chain is selected, all chains are listed. As every other iptables command, it applies to the specified table (filter is the default), so NAT rules get listed by
* **-F, --flush** [*chain*]
	+ Flush the selected chain (all the chains in the table if none is given). This is equivalent to deleting all the rules one by one.
* **-Z, --zero** [*chain*]
	+ Zero the packet and byte counters in all chains. It is legal to specify the **-L, --list** (list) option as well, to see the counters immediately before they are cleared. (See above.)

Hopefully this will shine some light about what these commands mean, but what about the rules themselves?

### Rule specifications

The following parameters make up a rule specification (as used in the add, delete, insert, replace and append commands).

* **-p, --protocol** [!] *protocol*
	+ The protocol of the rule or of the packet to check. The specified protocol can be one of *tcp*, *udp*, *icmp*, or *all*, or it can be a numeric value, representing one of these protocols or a different one. A protocol name from /etc/protocols is also allowed. A "!" argument before the protocol inverts the test. The number zero is equivalent to *all*. Protocol *all* will match with all protocols and is taken as default when this option is omitted.
* **-s, --source** [!] *address*[/*mask*]
	+ Source specification. *Address* can be either a network name, a hostname (please note that specifying any name to be resolved with a remote query such as DNS is a really bad idea), a network IP address (with /mask), or a plain IP address. The *mask* can be either a network mask or a plain number, specifying the number of 1's at the left side of the network mask. Thus, a mask of *24* is equivalent to *255.255.255.0*. A "!" argument before the address specification inverts the sense of the address. The flag **--src** is an alias for this option.
* **-d, --destination** [!] *address*[/*mask*]
	+ Destination specification. See the description of the **-s** (source) flag for a detailed description of the syntax. The flag **--dst** is an alias for this option.
* **-i, --in-interface** [!] *name*
	+ Name of an interface via which a packet was received (only for packets entering the **INPUT**, **FORWARD** and **PREROUTING** chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

These are not all but are the most common. Keep in mind that you can stack them (using several one after another) and setting up multiple one’s rules to form a hierarchy of rules (as explained above in [Firewall Chains](#_Firewall_Chains)).

### TARGETS

When a firewall rule matches we can determine what happens to those packets is specified by the value of the target, which can be the name of a user-defined chain, one of the targets described in [**iptables-extensions**](http://ipset.netfilter.org/iptables-extensions.man.html)(8), or one of the special values **ACCEPT**, **DROP** or **RETURN**.

* **ACCEPT** means to let the packet through.
* **DROP** means to drop the packet on the floor.
* **RETURN** means stop traversing this chain and resume at the next rule in the previous (calling) chain.

If the end of a built-in chain is reached or a rule in a built-in chain with target **RETURN** is matched, the target specified by the chain policy determines the fate of the packet.

# Validation

For validation, first ask permission to use two servers, and then create a document with a header and footer according to the [guidelines](https://docs.switzernet.com/3/company/181003-how-to-work-with-documentations/) and place screenshots with the results of the tasks in a dedicated chapter named “Validation”:

1. Access the Server1 with [ssh](https://docs.switzernet.com/3/support/101014-training-connection-sip/) and issue a date command for timestamp and execute a ping to the server Server2.
2. From your local computer, access the Server2 (also with ssh) and apply a rule to block incoming traffic from the Server1.
3. Redo task 1 and prove the rule is applied successfully.
4. Redo task 2 but this time remove the rule that blocks traffic from the Server1.
5. Redo task 1 and prove the rule is removed successfully.

# References

<https://docs.switzernet.com/3/support/101014-training-connection-sip/>

<http://ipset.netfilter.org/iptables.man.html>

<https://netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.txt>

<https://docs.switzernet.com/3/company/181003-how-to-work-with-documentations/>

# End of document

\* \* \*

© 4z.com