Iptables !

Πίνακας περιεχομένων

1. Install swarmlab-sec (Home PC)

HowTo: See http://docs.swarmlab.io/lab/sec/sec.adoc.html

NOTE

Assuming you're already logged in

2. iptables

iptables is a command line utility for configuring Linux kernel **firewall** implemented within the Netfilter project. The term "iptables" is also commonly used to refer to this kernel-level firewall. It can be configured directly with iptables, or by using one of the many

More: wikipedia

- Console tools

and

- Graphical front-ends.

iptables is used for IPv4 and "ip6tables" is used for ihttps://en.wikipedia.org/wiki/IPv6[IPv6]. Both "iptables" and "ip6tables" have the same syntax, but some options are specific to either IPv4 or IPv6.

2.1. Installation

The Swarmlab.io kernel is compiled with iptables support.

2.2. Front-ends

2.2.1. Console

- Shorewall, High-level tool for configuring Netfilter.

You describe your firewall/gateway requirements using entries in a set of configuration files.

shorewall

- Arno’s Secure firewall for both single and multi-homed machines.

Very easy to configure, handy to manage and highly customizable. Supports: NAT and SNAT, port forwarding, ADSL ethernet modems with both static and dynamically assigned IPs, MAC address filtering, stealth port scan detection, DMZ and DMZ-2-LAN forwarding, protection against SYN/ICMP flooding, extensive user definable logging with rate limiting to prevent log flooding, all IP protocols and VPNs such as IPsec, plugin support to add extra features.|

arno-iptables-firewall

- FireHOL Language to express firewalling rules, not just a script that produces some kind of a firewall. It makes building even sophisticated firewalls easy - the way you want it.

http://firehol.sourceforge.net

- firewalld (firewall-cmd) Daemon and console interface for configuring network and firewall zones as well as setting up and configuring firewall rules.

firewalld

2.2.2. Graphical

- Firewall Builder

firewall configuration and management tool that supports iptables (netfilter), ipfilter, pf, ipfw, Cisco PIX (FWSM, ASA) and Cisco routers extended access lists. The program runs on Linux, FreeBSD, OpenBSD, Windows and macOS and can manage both local and remote firewalls.

fwbuilder

- firewalld

(firewall-config) Daemon and graphical interface for configuring network and firewall zones as well as setting up and configuring firewall rules.

firewalld

- FireStarter

High-level GUI Iptables firewall for Linux systems

firestarter

3. Basic concepts

iptables is used to inspect, modify, forward, redirect, and/or drop IP packets.

- The code for filtering IP packets is already built into the kernel and is organized into a collection of **tables**, each with a specific purpose.
- The tables are made up of a set of predefined **chains**, and the chains contain **rules** which are traversed in order.
- Each rule consists of a predicate of potential matches and a corresponding action (called a **target**) which is executed if the predicate is true; i.e. the conditions are matched.
- If the IP packet reaches the end of a built-in chain, including an empty chain, then the chain's **policy** target determines the final destination of the IP packet.

iptables is the user utility which allows you to work with these chains/rules.

Understanding how iptables works

The key to understanding how iptables works is this chart.

The lowercase word on top is the **table** and the upper case word below is the **chain**.

- Every IP packet that comes in **on any network interface** passes through this flow chart from top to bottom.

All interfaces are handled the same way; it's up to you to define rules that treat them differently.

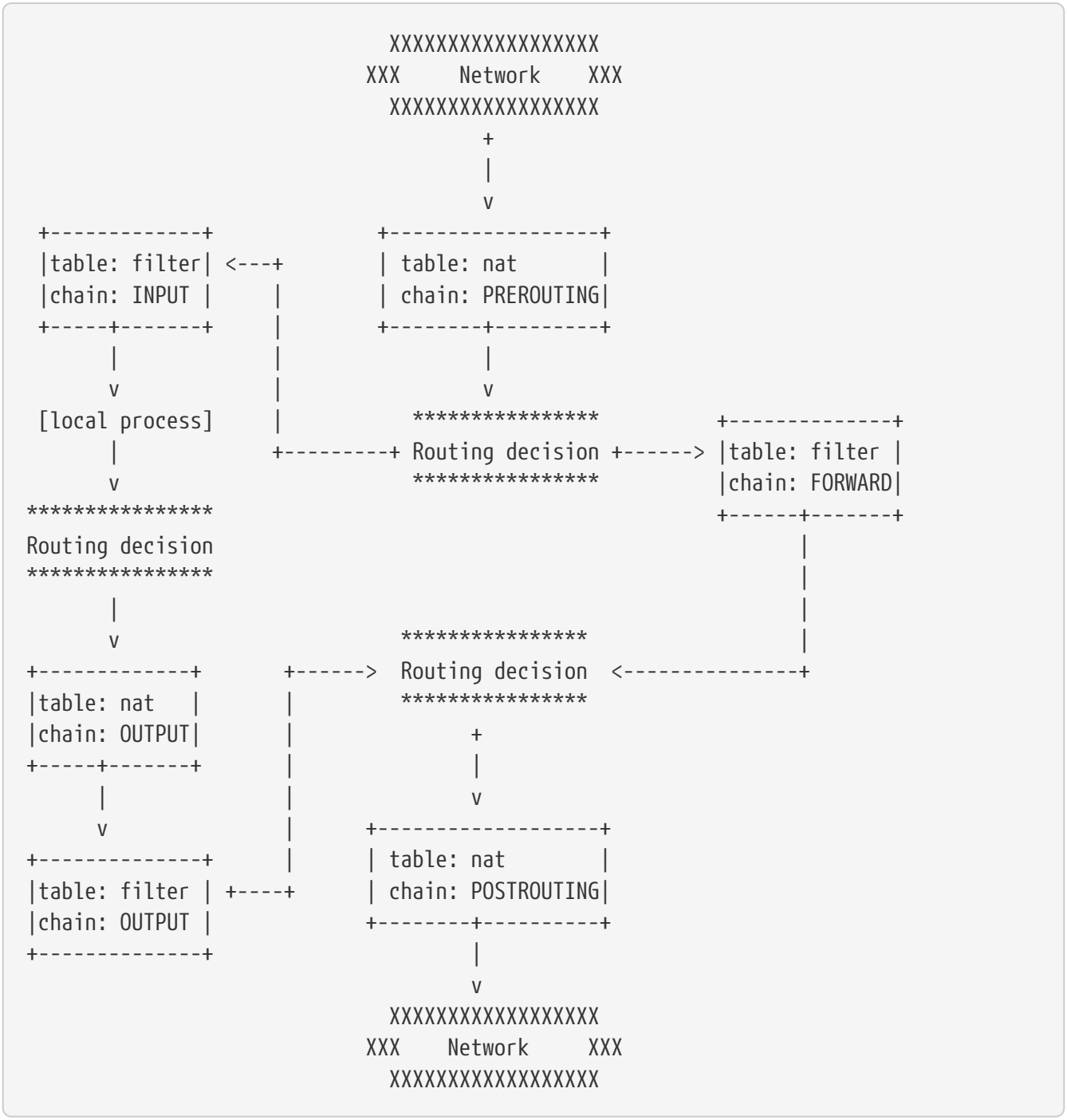
Some packets

- are intended for local processes, hence come in from the top of the chart and stop at **Local Proces**,
- while other packets are generated by local processes; hence start at **Local Process** and proceed downward through the flowchart.

A detailed explanation here.

In the vast majority of use cases you won't need to use the **raw**, **mangle**, or **security** tables at all.

Consequently, the following chart depicts a simplified network packet flow through **iptables**:



3.1. Table

iptables contains five tables:

Chains

Tables consist of **chains**, which are lists of rules which are followed in order.

3.1.1. Filter

This is the default table.

Its built-in chains are:

```
Input: packets going to local sockets
Forward: packets routed through the server
Output: locally generated packets
```

3.1.2. Nat

When a packet creates a new connection, this table is used.

Its built-in chains are:

```
Prerouting: designating packets when they come in
Output: locally generated packets before routing takes place
Postrouting: altering packets on the way out
```

3.1.3. Mangle

Used for special altering of packets.

Its built-in chains are:

```
Prerouting: incoming packets
Postrouting: outgoing packets
Output: locally generated packets that are being altered
Input: packets coming directly into the server
Forward: packets being routed through the server
```

3.1.4. Raw

Primarily used for configuring exemptions from connection tracking.

Its built-in chains are:

```
Prerouting: packets that arrive by the network interface
Output: processes that are locally generated
```

3.1.5. Security

Used for Mandatory Access Control (MAC) rules. After the filter table, the security table is accessed next.

Its built-in chains are:

```
Input: packets entering the server
Output: locally generated packets
Forward: packets passing through the server
```

In most common use cases you will only use two of these: **filter** and **nat**.

3.2. Rules

Packet filtering is based on **rules**, which are specified by multiple **matches** (conditions the packet must satisfy so that the rule can be applied), and one **target** (action taken when the packet matches all conditions).

The typical things a rule might match on are

- what interface the packet came in on (e.g eth0 or eth1),
- what type of packet it is (ICMP, TCP, or UDP),
- or the destination port of the packet.

Targets are specified using the **-j** or **--jump** option.

Targets can be either - user-defined chains (i.e. if these conditions are matched, jump to the following user-defined chain and continue processing there), one of the special built-in targets, - or a target extension.

- Built-in targets are **ACCEPT**, **DROP**, **QUEUE** and **RETURN**
- target extensions are, for example, **REJECT** and **LOG**.

- If the target is a built-in target, the fate of the packet is decided immediately and processing of the packet in current table is stopped.
- If the target is a user-defined chain and the fate of the packet is not decided by this second chain, it will be filtered against the remaining rules of the original chain.

Target extensions can be either **terminating** (as built-in targets) or **non-terminating** (as user-defined chains)

3.3. Traversing Chains

A network packet received on any interface traverses the traffic control chains of tables in the order shown in the this chart

- The first routing decision involves deciding if the final destination of the packet is the local machine (in which case the packet traverses through the **INPUT chains**
- or elsewhere (in which case the packet traverses through the **FORWARD chains**.
- Subsequent routing decisions involve deciding what interface to assign to an outgoing packet.

At each chain in the path, every rule in that chain is evaluated in order and whenever a rule matches, the corresponding target/jump action is executed.

The 3 most commonly used targets are **ACCEPT**, **DROP**, and **jump** to a user-defined chain.

While built-in chains can have default policies, user-defined chains can not.

- If every rule in a chain that you jumped fails to provide a complete match, the packet is dropped back into the calling chain as illustrated here.
- If at any time a complete match is achieved for a rule with a **DROP** target, the packet is dropped and no further processing is done.
- If a packet is **ACCEPT**ed within a chain, it will be **ACCEPT**ed in all superset chains also and it will not traverse any of the superset chains any further.

However, be aware that the packet will continue to traverse all other chains in other tables in the normal fashion.

4. Usage

4.1. Showing the current rules

```
# iptables -nvL

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

If the output looks like the above, then there are no rules (i.e. nothing is blocked) in the default filter table

4.2. Resetting rules

You can flush and reset iptables to default using these commands:

```
# iptables -F
# iptables -X
# iptables -t nat -F
# iptables -t nat -X
# iptables -t mangle -F
# iptables -t mangle -X
# iptables -t raw -F
# iptables -t raw -X
# iptables -t security -F
# iptables -t security -X
# iptables -P INPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT ACCEPT
```

The -F command with no arguments flushes all the chains in its current table. Similarly, -X deletes all empty non-default chains in a table.

Individual chains may be flushed or deleted by following -F and -X with a [chain] argument.

4.3. Editing rules

Rules can be edited by

- appending -A a rule to a chain,
- inserting -I it at a specific position on the chain,
- replacing -R an existing rule,
- or deleting -D it.

The first three commands are exemplified in the following.

First of all, our computer is not a router (unless, of course, it is a router). We want to change the default policy on the FORWARD chain from ACCEPT to DROP.

```
# iptables -P FORWARD DROP
```

4.4. Examples

We are going to use Shorewall as an iptables configuration tool.

See Appendix.

Here are some examples of "raw" iptables command lines.

4.4.1. Block Traffic by PortPermalink

You may use a port to block all traffic coming in on a specific interface.

For example:

```
iptables -A INPUT -j DROP -p tcp --destination-port 110 -i eth0
```

Let's examine what each part of this command does:

- **-A** will add or append the rule to the end of the chain.

  ```
  **INPUT** will add the rule to the table.
  ```

  ```
  **DROP** means the packets are discarded.
  ```

- **-p tcp** means the rule will only drop TCP packets.
- **--destination-port 110** filters packets targeted to port 110.
- **-i eth0** means this rule will impact only packets arriving on the eth0 interface.

4.4.2. Drop Traffic

In order to drop all incoming traffic from a specific IP address, use the iptables command with the following options:

```
iptables -I INPUT -s 198.51.100.0 -j DROP
```

To remove these rules, use the **--delete** or **-D** option:

```
iptables --delete INPUT -s 198.51.100.0 -j DROP
iptables -D INPUT -s 198.51.100.0 -j DROP
```

4.4.3. Block or Allow Traffic by Port Number

One way to create a firewall is to block all traffic to the system and then allow traffic on certain ports.

Below is a sample sequence of commands to illustrate the process:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -m comment --comment "Allow loopback connections" -j ACCEPT
iptables -A INPUT -p icmp -m comment --comment "Allow Ping to work as expected" -j
ACCEPT
iptables -A INPUT -p tcp -m multiport --destination-ports
22,25,53,80,443,465,5222,5269,5280,8999:9003 -j ACCEPT
iptables -A INPUT -p udp -m multiport --destination-ports 53 -j ACCEPT
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

Let's break down the example above.

The **first two** commands add or append rules to the **INPUT chain** in order to allow access on specific ports.

The **-p tcp** and **-p udp** options specify either **UDP** or **TCP** packet types.

The **-m** multiport function matches packets on the basis of their source or destination ports, and can accept the specification of up to 15 ports.

Multiport also accepts **ranges such as 8999:9003** which counts as 2 of the 15 possible ports, but matches ports 8999, 9000, 9001, 9002, and 9003.

The next command **allows all incoming** and **outgoing packets** that are associated with existing connections so that they will not be inadvertently blocked by the firewall.

The final two commands use the **-P** option to describe the **default policy** for these chains. As a result, all packets processed by **INPUT** and **FORWARD** will be dropped by default.

Note that the rules described above only control incoming packets, and do not limit outgoing connections.

4.5. More Examples

```
# Allow all loopback (lo0) traffic and reject traffic
# to localhost that does not originate from lo0.
-A INPUT -i lo -j ACCEPT
-A INPUT ! -i lo -s 127.0.0.0/8 -j REJECT

# Allow ping.
-A INPUT -p icmp -m state --state NEW --icmp-type 8 -j ACCEPT

# Allow SSH connections.
-A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT

# Allow HTTP and HTTPS connections from anywhere
# (the normal ports for web servers).
-A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 443 -m state --state NEW -j ACCEPT

# Allow inbound traffic from established connections.
# This includes ICMP error returns.
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Log what was incoming but denied (optional but useful).
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables_INPUT_denied: " --log
-level 7

# Reject all other inbound.
-A INPUT -j REJECT

# Log any traffic that was sent to you
# for forwarding (optional but useful).
-A FORWARD -m limit --limit 5/min -j LOG --log-prefix "iptables_FORWARD_denied: "
--log-level 7

# Reject all traffic forwarding.
-A FORWARD -j REJECT
```

Appendix A: How to use iptables

Shorewall is not the easiest to use of the available iptables configuration tools but I believe that it is the most flexible and powerful.

It can handle complex and fast changing network environments.

It needs multiple configuration files, even for simple setups.

Suitable for powerusers! - Most likely there are a lot of these among our Students :-)

Shorewall is very popular!

Origin

Reminder

Caminante, no hay camino,
se hace camino al andar.

Wanderer, there is no path,
the path is made by walking.

Antonio Machado Campos de Castilla