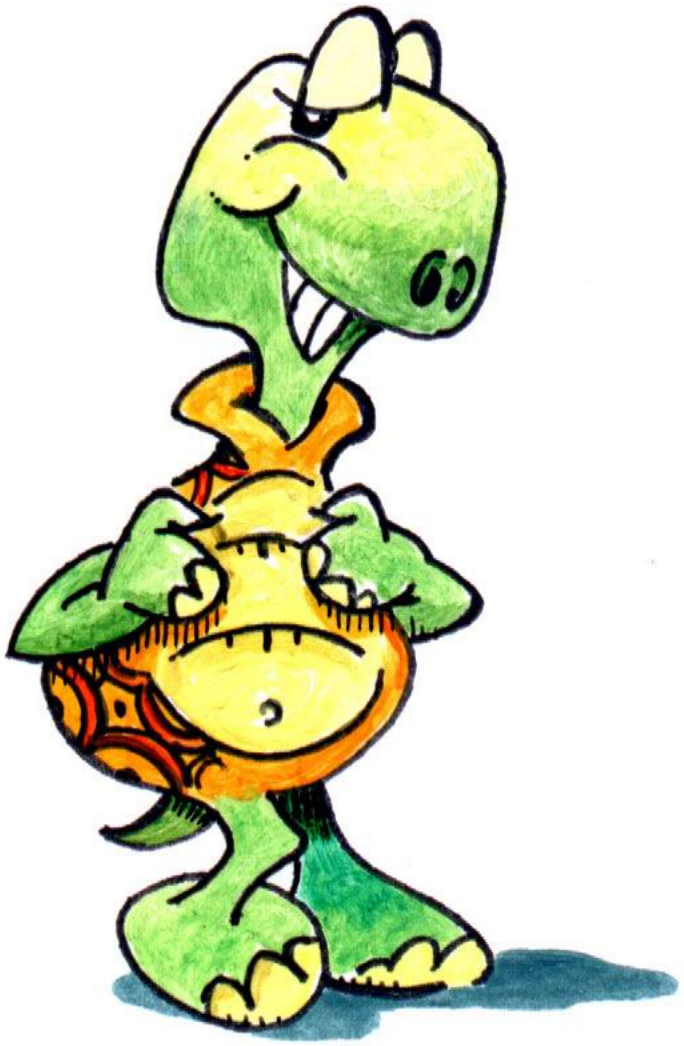
Manuale Turtle Firewall



Andrea Frigido
Friweb snc
Translator: Emanuele Tatti

Manuale Turtle Firewall

by Andrea Frigido
Translator: Emanuele Tatti

Published 2002

Table of Contents

Chapter 1. Overview

1.1. What is Turtle Firewall?

Turtle Firewall is a software which can configure a firewall based on netfilter, so it's compatible with the Linux Kernel versions 2.4.x and 2.5.x.

Turtle Firewall starts up like every other Linux service during the boot time and its configuration is extremely simple thanks to the web interface implemented as a module for webmin.

1.2. Requirements

- Linux kernel 2.4.x o 2.5.x.[1]
- iptables, available in its default path.
- netfilter modules, including ip_conntrack, ip_conntrack_ftp e ip_nat_ftp, built-in or loaded as modules with insmod.
- Perl 5.6[2].
- Expat library (usually included in all distros).
- Perl moduleXML::Parser[3].
- Webmin[4].

Notes

1. http://www.kernel.org
2. http://www.perl.com
3. http://search.cpan.org
4. http://www.webmin.com

Chapter 2. Elements of a system protected by a firewall

2.1. Firewall items

A firewall is a system made up of several network interfaces connected to different branches of different networks. Every interface allows the firewall to communicate with the zone the interface is connected to. A *zone* can be made up of one or more subnets or even all Internet.

Every firewall interface can communicate with a zone. If an host belonging to the zone X wants to communicate with another host that belong to the zone Y it has to pass through the firewall. If it is able to avoid the obstacle, we haven't built a good network as our firewall cannot do its task well.

Having said this, we can set in the firewall a series of rules to determine which packets can pass from a zone to another. These rules can consider a series of characteristics of the packet that has to be filtered: zone/subnet/source host, zone/subnet/destination host, protocol (tcp/udp/icmp), source and destination port, ecc.

Using Turtle Firewall is possible to define the elements that can be source or destination of a connection, assign them a name and then use the name to define the firewall rules.

If you want to view the list of the firewall elements, click on the *Items* icon in the main menu of the Turtle Firewall Webmin module. Now you can see, divided into four tables, ZONE, NET, HOST and GROUP items.

2.2. Zone

A zone is defined by an arbitrary name and the interface used by the firewall to communicate with the hosts of that zone.

Usually a common network with a firewall is made up of 3 zones: the internal net (usually called *good* because we are there, the good :-)) which we should assure a high security level to, the external net that is generally Internet (usually called *bad* because there are the wicked there, who want *"certainly"* enter our *good* zone ;-( ) and finally the zone where we have our public servers, like a web server, which cannot be completely armoured like the good zone because we have to make public services accessible to Internet users (usually called *dmz*, demilitarized zone). Of course a network can be much more complex, so there could be a lot of dmz or good zones but if you want to set a good firewall you have to follow essentially these rules.

Clicking on *"Create new zone"* you will see a short form with which you'll be able to set the name you want to assign to the new zone (good, bad, dmz, my_zone, etc.) and the network interface to use to reach it (eth0, eth1, ppp0, etc.).

Now you can click on *Create* and the new zone will be recorded permanently.

In this way, clicking on the zone name, you can modify the network interface the zone is associated to.

2.3. Net

A net is identified by a network address and a netmask which determine the network dimension.

The netmask can be expressed by four numbers (0-255) separated by a dot (ex. *255.255.255.0*) or simply by the number of binary digits that are part of the network address (ex. *24*).

For more information visit this URL: http://netfilter.samba.org/documentation/HOWTO//networking-concepts-HOWTO-4.html#ss4.1

Likewise we saw for the zones, if you want to create a new net you have to click on *"Create new net"*.

Now you can set a name for the net (ex. internal_net, subnet1, etc.), a network address (ex. 192.168.0.0), a netmask (es. 255.255.0.0) and, in the end, set the zone this subnet belong to (ex. good).

You should remember that you cannot define a net until you have declared the zone it belong to.

2.4. Host

A host can be either a single computer or a device (router or others) reachable by our firewall; so it has an address (*ip*) and belongs to a *zone*.

If you want to create a new host you have to click on *"Create new host"*, assign it a name (ex. PC_Andrea, host1, WebServer, etc.), set an IP address (ex. 192.168.1.123) and, finally, the zone the host belong to.

2.5. Group

With this particular item you can group a series of elements you have previously defined; these elements can be zones, nets or hosts. A Group is identified by a name and it can be used in the firewall rules like a normal system element. This feature makes very simple to define rules which are common to more hosts (even nets or zones), thus the firewall configuration is more synthetic, readable and simple to maintain. If, for example, we want to give ssh access towards a server to our 3 system admins, it will be sufficient to create a group called *"administrators"* that will be made up of the 3 admin hosts, and then apply only a ssh rule to the *"administrators"* group.

To create a new Group you have to click on *"Create new group"*, assign it a name (ex. servers, privileged_host, etc.) and finally, checking a series of check-box, set which items will belong to the Group we are defining.

2.6. The item FIREWALL

Turtle Firewall considers, apart from the zones we have defined, a further default zone called FIREWALL which identifies our linux-box where the firewall itself has been installed. Every packet which is not in transit through the firewall but leaves from it or is addressed to it, involves the use of the "FIREWALL" zone. FIREWALL isn't a host because a firewall doens't have only one ip (it has an IP per interface) therefore it's more appropriate to consider it a zone apart.

It's extremely important to set rules that protect our firewall. The common sense should drive us to make the Firewall completely isolated, denying the access to the FIREWALL zone to all, but often we want to modify the firewall configuration while we are sitting comfortably in our position (laziness is the most frequent cause of compromising a security system ;-)) and therefore I suggest to make available a number of services as minor as possible to a number of hosts as minor as possible; only the ssh service to the network administrator host is a good compromise.

Note: Turtle Firewall doesn't allow you to define 2 elements with the same name, even if they are different items. For example, there cannot be a host which has the name of a zone. For this reason you cannot define an element called "FIREWALL".

Notes

1. http://netfilter.samba.org/documentation/HOWTO//networking-concepts-HOWTO-4.html#ss4.1

Chapter 3. Firewall rules

Turtle Firewall default policy forbids every connection the Firewall monitors. You can add to this default policy some rules which determine the connections you want to allow. Of course there are other ways to configure a firewall but this is considered to be more careful so I chose it for Turtle Firewall.

To allow a simple connection between 2 hosts (A, B), consider for example a http connection, you have to define two filter rules: at first one which allows the packets that go from the *A* host to the *B* one, the one which allows the return of reply packets from *B* to *A*.

Turtle Firewall simplifies this operation defining a series of *services*and making itself responsible for setting all the filtering rules needed to guarantee those *services*. This way a filter rule of Turtle Firewall is simply to use because it's based on an Internet service rather than on the characteristics of single packets. To indicate that you want to allow A to communicate with B through the http protocol, it will be sufficient to define only a Turtle Firewall rule in which you'll specify the name of the source item (*A*), the name of the destination one (*B*) and the*service* to use (http). Remember that if you define a http rule from A to B, B cannot make use of the same service but it can only accept and reply to the http requests from A.

To view the list of active rules you have to click on the "Rules" icon from main menu of the Turtle Firewall module.

To create a new rule, click on "*Create new rule*" from the "*Rules*" section. At this point set the communication source item, the destination item, the service to use (http, ssh, ftp, etc.), in case a port (needed only for generic services such as tcp or udp) and, finally, select the "*Active*" check-box that makes the rule active.

Chapter 4. NAT (Network Address Translation) Rules and Masquerading

4.1. Overview

With the new 2.4.x Kernel and iptables, NAT and Masquerade are handled by the Netfilter module in the kernel. With these rules we can change source and destination item of the connection, and this is useful when we want to redirect all the traffic destinated to host 'X' to another host 'Y'.

If, for example, our provider assigned us some public addresses but our web server is situated in a dmz with a private address, to make this server accessible from outside we need to set a NAT rule which redirects all the connections destinated to the public address we chose for our web server towards its private address (real). The client which establishes the connection won't notice anything and will believe really to communicate with a web server whose IP address is public.

Masquerading is a particular case of NATting and it is applied when we want that connections between 2 zones seem, to the destination zone, as if they came from our firewall. Masquerading is commonly used for connections towards Internet: we don't want that our internal hosts are identified by Internet hosts so we tell our firewall to masquerade all communications toward outside. The connection will appear, to Internet hosts, to come from our firewall.

To view a list of NAT and Masquerading rules, click on *"NAT and Masquerading"* icon from main menu of Turtle Firewall module.

4.2. Nat (Network Address Translation)

A NAT rule has only 2 attribute: *virtual*and *real*. *Virtual* indicates the virtual item, usually a host with a public IP which has to be redirected towards a real item (usually a host situated in dmz). Hosts indicated by "virtual" and "real" have, of course, to be previously defined.

To create a new NAT rule you have to select *"Create new NAT"* Nat and Masquerading section. Now set the virtual item, then the real one towards which the traffic must be redirected.

4.3. Masquerade

Masquerade provides that all packets which, through the firewall, go out towards a determined zone appear as sent by the firewall itself.

To crete a new Masquerade rule you have to click on *"Create new Masquerade"*. Now it's sufficient to set to zone that you want to apply packets masquerading to.

Note: At the moment this feature is limited ;-(, in the future I would like to extend it introducing different masqueradings with other attributes such as source and destination hosts.

Chapter 5. Services (fwservices.xml)

5.1. Overview

There isn't any web interface to define a service, so you have to work directly on the fwservices.xml file which is XML-based.

We already saw that Turtle Firewall rules are based on services (http, ftp, etc.), now we will see how to define our services that we will be able to use and set in the firewall. In this case the configuration provides the use of a separated XML file which allows to define the characteristics that every packet must have to be considered valid for a determinate service. If you want to define a new service you should use the logic followed to define a new firewall rule. Essentially you have to define a certain number of filters which allow the packets traffic, if a packet doesn't satisfy any filter it will be rejected.

The XML definitions file must have the root tag *services* (<services>) that has inside the *service*tags which define the single service. A service tag has two attributes: *name* and *description*, the former to define the name, the latter for a short description of the service. A service tag contains the *filter* tags, which indicate how the valid packets for that service are selected.

5.2. FILTER

The filter tag has a series of attributes which are used to define the characteristics that a packet must have to be considered valid. If the packet doesn't satisfy the parameters defined by the filter, the control passes to the following filter and so on. If no filter is verified, the packet is rejected.

Attributes:

- *direction*: defines the direction of the packet in transit. It can assume the *go* or the *back* value. Using go only the packets from an ipotetic source host to a destination host are considered, on the contrary using back the reply packets are considered.
- *p*: protocol, it can assume one of the following values: tcp, udp or icmp.
- *sport*: the port used by the source host socket.
- *dport*: destination port of the packet.
- *icmptype*: ICMP message type (you should use it only with p="icmp", see iptables doc.).
- *state*: packet state (see "state" instruction of iptables).
- *jump*: forced jump to a chain or a issue.

  It can assume the following values: *ACCEPT* (tha packet is accepted, it's the default chain if direction is set to *go*), *DROP* (the packet is not considerated valid, usually this jump is not used directly because it's not reported to the log file), *BACK* (the control passes to a special chain that allows only reply packets in transit of connections already established, it's the default chain if direction is set to *back*), *ICMP-ACC* (another special chain that has to be used with p="icmp", allows only standard icmp messages considered secure).

Note: If you set sport or dport as "PORT", the port will be set during the script generation as the value indicated by the *port* attribute of the *rule* tag in the firewall rule you have defined. This way you can define services with parameters.

Syntax:

```
<filter direction="go/back" p="tcp/udp/icmp" sport="nPortaSrc" dport="nPortaDst"
  icmptype="tipoMsgIcmp" jump="ACCEPT/DROP/BACK/ICMP-ACC"/>
```

Example 5-1. Definition of 3 services

```
<services>
  <service name="http" description="Servizio www o http">
    <filter direction="go" p="tcp" dport="www"/>
    <filter direction="back" p="tcp" sport="www"/>
  </service>

  <service name="tcp" description="Servizio TCP generico">
    <filter direction="go" p="tcp" dport="PORT"/>
    <filter direction="back" p="tcp" sport="PORT"/>
  </service>

  <service name="ping" description="icmp message echo-request and echo-
reply">
    <filter direction="go" p="icmp" ICMPTYPE="echo-request"/>
    <filter direction="back" p="icmp" ICMPTYPE="echo-reply"/>
  </service>
</services>
```

5.3. Default services (fwservices.xml)

fwservices.xml is used by Turtle Firewall as its source of services definitions. It contains the definitions of most commonly used services on the web and of course it can be integrated with your own services modifying, like we already saw, directly the XML code.

Default services

- *all*: All services opened
- *tcp*: Generic TCP protocol
- *udp*: Generic UDP protocol
- *ftp*: File Transfer Protocol
- *dns*: Domain Name Service
- *www*: World Wide Web HTTP
- *http*: World Wide Web HTTP
- *https*: HTTP protocol over TLS/SSL
- *auth*: Authentication Service
- *smtp*: Simple Mail Transfer Protocol
- *pop3*: Post Office Protocol version 3
- *imap*: Internet Message Access Protocol
- *ssh*: Secure Shell Protocol
- *ntp*: Network Time Protocol
- *icmp_acc*: Essential ICMP messages
- *ping*: ICMP messages echo-request and echo-reply
- *netbios_ns*: NETBIOS Name Service

- *netbios*: NETBIOS complete
- *netbios_ssn*: NETBIOS Session Service
- *cvs*: CVS Server Service
- *nntp*: NNTP Network News Transport Protocol
- *telnet*: Telnet Protocol
- *webmin*: Webmin (port 10000)
- *h323*: H323 Protocol (NetMeeting), Experimental
- *ipsec-ESP*: VPN IPSec protocol with IKE and ESP
- *ipsec-AH*: VPN IPSec protocol with IKE and AH
- *ipsec-ESP-AH*: VPN IPSec protocol with IKE, ESP and AH
- *afp-over-tcp*: AFP (Apple Filing Protocol) over TCP
- *nfs*: NFS (experimental)
- *mysql*: MySQL-Server
- *kazaa*: KaZaa Filesharing
- *pptp*: PPTP VPN Service
- *rdp*: Remote Desktop Protocol
- *pc-anywhere*: PC-Anywhere service (admin to hosts)
- *x11*: X Window System service (client to XServer)