FIREWALLS

Firewalls have nothing to do with cryptography, they offer a different kind of security.

The idea is to separate the local network from the internet in such a way that every data entering the local network will be analysed by the firewall. Commonly it is the incoming data that has to be analysed, packets outgoing the network are not so important. Actually also the inner data outgoing can be analysed, because some schemas and patterns can be recognized. There are so many topics on firewalls, we will be focusing on few details.

The local network can be made such that the firewall affects all the traffic coming from the outside world except for the so called Demilitarized Zone (DMZ), which is a zone in our local network accessible from outside, and it is useful when our local network provides some Server (which cannot be reached if a firewall is placed between it and the internet).

They can be applied more then one firewall, a common behaviour is to have one general firewall for the whole network, and other special firewalls for certain machines, so PCs can have their personal firewall.

Firewall can be cathegorized in 3 different types each one working at a different level of the network. Of course the higher is the level, the higher is the security offered by the firewall (we will see why with fragmentation attack). In fact a firewall that is working at a low level it will inspect single packets, but inspecting single packets it will not be able to recognize malicious data at application level.

Firewall types

PACKET FILTERING: analysing datagrams, in particular headers, and deciding whether datagrams can pass or not. We will be analyzing this kind of firewall.

PROXY GATEWAY: the firewall acts like a gateway, in the sense that all incoming traffic goes to the firewall first and all outgoing traffic appears to come from the firewall. It can be cathegorized in:

- Application-Level: every application has its own proxy firewall
- Circuit-Level: firewall working at lower levels, invisible to the application level

PERSONAL FIREWALL: firewall setted on the single PC

PACKET FILTERING

We will focus on this kind of firewall. For each packet, the firewall decides whether the packet has to proceed or not. An interesting thing about firewalls is the mode of operating: *stateless* or *stateful*. Stateless means every packet is anaylized one independently from the other, Stateful means the firewall maintains some informations about packets transiting through it, and the most used case of stateful firewall is *session filtering*. Stateful is hard to implement, too many packets to analyse. Usually the header of the packet is analyzed, in order to inspect the IP source, but it can be possible that further analysis is needed, so the firewall may inspect the payload of the packet in order to get the TCP port and take further decisions. So resuming the PACKET FILTERING is based on filtering *IP numbers* and *TCP ports*.

Beeing Stateful means that the firewall keeps information about some session, for example a TCP session. In such a way the firewall is able to understand what every packet means within that session. Therefore we can apply a kind of *session filtering*. It will mean that the firewall takes different decisions depending on the particular packet in the context of a session. Of couse such a firewall must be placed at an higher level w.r.t. IP level.

An example of filtering packets is for example analysing all packets coming from 192.168.4.* where '*' means any number. If we know a certain subnet is dangerous, let's say the subnet 10.25.68.0/24, we can add a rule, write something like "10.25.68.* BLOCK" in order to block all the traffic coming from that subnet. Of course we can specifty further details, for example we can block specific IP address togheter with a specific port.

However PACKET FILTERING is vulnerable to *ip spoofing* since the attacker can change its IP bypassing the rules of the firewall. A well known attack against firewalls is *fragmentation attack*, the idea is simple, fragment a packet into two subpackets such that they seem innoque alone, but once reassembled on the upper layer they constitute a malicious fragment. Also *DoS attacks* can be performed, like the one against the 3-way handshaking of TCP. The *packet filtering* firewall will not be able to recognize such an attack.

In LINUX distributions there is a useful tool that provides firewall features, we are talking about iptables. It perform stateful analysis on session filtering, and performs also other features like NAT tables. We will focus only on its packet filtering feature, where the analysis is stateful.

IPTABLES

Iptables offers firewall features providing a sort of session filtering, but it works at packet level, not application level. Iptables works on *tables*, as the name suggests.

An important concept of iptables is the concept of **chain**: a chain is a list of rules, processed in a sequential way, which determine whether a packet has to be accepted or dropped. We will focus only on three different kind of chains: **INPUT**, **FORWARD** and **OUTPUT**. The meaning is the following:

- A chain called INPUT is a list of rules that consider incoming packets
- A chain called FORWARD is alist of rules that consider all those packets that were routed to us but are not to be delivered for us so they just have to be forwarded
- A chain called OUTPUT is just the converse of the INPUT chain, so it deals with outgoing packets

As we can imagine, when using iptables on a PC (so we are implementing a personal firewall), then INPUT and OUTPUT chains are meaningful, FORWARD chain is not important. On the countrary, when iptables is working on a gateway, the FORWARD chain is the most important.

Working with iptables, a decision associated to a specific rule is called **target**, that specifies what to do if that packet has matched with that specific rule. The most common targets are **accept**, **drop**, **queue**, **return**.

Accept: let the packet through

Drop: drop the message without leaving a feedback, no ICMP messages sent.

Of course custom targets can be written. A custom target will be implemented as a sub-chain, so other rules with other targets. Here the **return** target makes sense, where it means that we have to come back to the upper chain and continue analyzing. It is like a custom target is a sorth of routine that we write, like the ones we write in the programming lenguages.

Then there is the concept of **external module**, that can be imported for implementing filters for a specific protocol, and the we can change rules whithin the module. This because of course it requires too much effort to configure iptables from scratch, so we can rely on built patterns and then we modify them according to our preferences.

EXAMPLES

First of all, let's see the command line usage of IPTABLES.

-A	Used to *append* a chain-rule specification (es. -A INPUT, -A OUTPUT, -A FORWARD)
-p	Specifies the protocol (es. -p TCP, -p UDP, …)
-i	Specifies the input interface (es. -i eth0)
-o	Specifies the ouput interface (es. -o eth1)
-s	Specifies the source address (es. -s 192.168.1.104)
-d	Specifies the destination address(es. 192.168.1.124)
--sport	Specifies the range of source ports (es.--sport 1024:65535)
--dport	Specifies the range of destination ports (es. --dport 1024:65535)
-j	Specifies the target (es. -j ACCEPT, -j REJECT, -j DROP)

Example 1

Assume that eth0 is the interface of a router towards public Internet. Block all incoming traffic from eth0

> iptables -A FORWARD -i eth0 -j DROP

So we have specified to add another rule with -A, specifying the kind of chain FORWARD, and specifying the interface eth0 with -i and specifying to DROP all the packets on this interface. Notice that with DROP we will not send back any response, just drop the packet.

Example 2

Accept packet from outside if they refer to a TCP connection started within the network

> iptables -A FORWARD -i eth0 -m state --state ESTABLISHED -j ACCEPT

Where ESABLISHED allows to decide whether the connection originated from the inside or outside

Example 3

Allow firewall to accept TCP packets for routing when they enter on interface eth0 from any IP address and are destinated 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

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

Example 4

Allow firewall to send ICMP echo-requests and in turn accept the expected ICMP echo-replies

> iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

> iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT