Department of Information Engineering (DEI)
Master degree on ICT for Internet and Multimedia Engineering (MIME)

INTERNET – LAB Experience 5: Firewall
USER GUIDE

1 Firewalls and Packet Filters

In this experience, we are going to use Katharà to learn about firewalls. In general, a firewall is a device (which can be either physical, i.e., a separate box on the network edge, or logical, i.e., a process on a machine in the network) that implements access control between networks. A typical usage of firewall is to protect information on a local trusted network, or Intranet, from the unsafe Internet, and allow limited access only to authorized hosts and applications.

"Firewall" is a kind of catch-all term that can be used for devices ranging from proxy servers and application layer firewalls to packet filters (stateless or stateful). Packet filters essentially work by checking incoming and outgoing packets against some pre-set criteria, and only allowing to pass those that match said criteria.

- *Stateless* packet filters inspect packets individually, not as part of an ongoing connection; they typically check the protocol type and the source and destination addresses and ports, dropping (i.e., silently discarding) or rejecting (i.e., discarding with an error message) packets that do not match the whitelist criteria.
- *Stateful* packet filters can implement more complicated rules: the inspection capability now reaches the transport layer, including the possiblity to:
  - keep track of previously seen packets;
  - consider connections and whether packets are part of an ongoing "conversation" or is the beginning of a new one;
  - speed up process by making connection-level decisions with no full packet inspection.

The "conversation" the firewall sees does not require the protocol to be connection-oriented: it is easy to determine a connection for a TCP flow, but a flow of connectionless UDP packets can be recognized and treated as a whole by using alternative parameters such as the interval between a packet and its reply. Some application-layer protocols, such as FTP, use multiple flows, and information about the flow might be contained in the transport layer payload; in this case, the firewall needs to perform deep packet inspection, checking the content of packets to understand the ongoing application-layer "conversations". The firewall also needs to know how these protocols work in depth, or it would have no way to decode and understand the relevant information.

Network Address and Port Translation (NAPT) is a fix to the shortage of IPv4 public addresses: multiple hosts can share the same public IP address, as the edge router handles the translation between the public and private addresses. This is also useful for packet filters: since private addresses are inherently *not-routable* from the outside, private LANs have a "natural protection" against access from outside. NAPT also allows to change the source and/or destination address and port of packets, which is extremely helpful to keep "conversation" state.

The Linux tool `iptables` is a powerful packet filter, and it is the one we are going to use.

2 Using `iptables`

`iptables` is a user-space tool that acts on its kernel-space counterpart, called *NetFilter*, to implement a stateful packet filter with NAPT. In the latest Linux distributions, the new tool `nftables` essentially replaces `iptables`' functionality, with some additional capabilities.

NetFilter is structured on two levels: there are four *tables*, which contain *chains*. The tables are called `filter`, `nat`, `mangle` and `raw`, and each contains some pre-defined chains, with the possibility to add user-defined ones.

- The `filter` table is used for basic packet filtering and contains three pre-defined chains:
  - `INPUT`
  - `OUTPUT`
  - `FORWARD`

  Every chain contains a list of rules, defining some matching criteria and a target. The packet is treated accordingly to the first matching rule, or a default policy if it matches no rules. The most common targets are:
  - `ACCEPT` accepts the packet
  - `DROP` silently discards it
  - `REJECT` discards it and sends an error message to the source
  - `user-defined-chain` redirects it to a specified user-defined chain
  - `RETURN` returns from the user-defined chain
- The `nat` table is used for NAPT operations and contains three pre-defined chains:
  - `PREROUTING`
  - `OUTPUT`
  - `POSTROUTING`

  In this table, targets are packet manipulation operations, which often change the source and destination address and port. The most common targets are:
  - `DNAT` changes the destination address and port of a packet
  - `SNAT` changes the source address and peply, or Forwardort of a packet
  - `MASQUERADE` works like `SNAT`, but it supports firewalls with dyanmic public IPs
  - `REDIRECT` redirects the packet to the firewall
- The `mangle` table is used for packet alteration (e.g., changing the Time To Live field) and contains three pre-defined chains:
  - `PREROUTING`
  - `INPUT`

  - `OUTPUT`
  - `FORWARD`
  - `POSTROUTING`

  We are not going to use this table, so we won't give any further details on it.

- The `raw` table is used to set up exceptions to the "conversation" tracking system and contains two chains:
  - `PREROUTING`
  - `OUTPUT`

  We are not going to use this table, so we won't give any further details on it.

Figure 1 shows the path a packet takes through the list of tables and chains before being sent, received or forwarded. As you can see from the figure, the forwarded packets (i.e., packets for which the host acts as an intermediate hop, and is neither the sender nor the receiver) follow a different path from packets to or from the host itself. This allows routers to be isolated from the environment while still performing their job. In the following description, we skip the `mangle` and `raw` tables for simplicity, as they are used less often.

2.1 Input Stream Packets

Input stream packets have the host that `iptables` is running on as a destination.

- As the packet enters the network interface of the firewall, it is sent to the `PREROUTING` chain of the `nat` table, which is typically used for Destination Network Address Translation (DNAT) operations
- The packet is then routed. This happens after the DNAT operations, so the packet contains its "true" destination address. The destination is one of the local processes, as this is an input packet
- The packet enters the `INPUT` chain of the `filter` table, and is filtered.
- If the filter does not stop it, it can reach its destination (a local process).

2.2 Output Stream Packets

Output stream packets are sent by the host that `iptables` is running on.

- As the packet is generated by a local process, it is assigned to an interface (and, consequently, the correct IP address)
- The packet enters the `OUTPUT` chain of the `filter` table, and is filtered.
- After the packet is routed, it enters the `POSTROUTING` chain of the `nat` table, which is used for Source Network Address Translation (SNAT) operations. Note that filtering after SNAT does not make sense, as all packets have the public address and not the one that the process that created them sees.
- If the filters do not stop it, the packet can exit on one of the network interfaces.

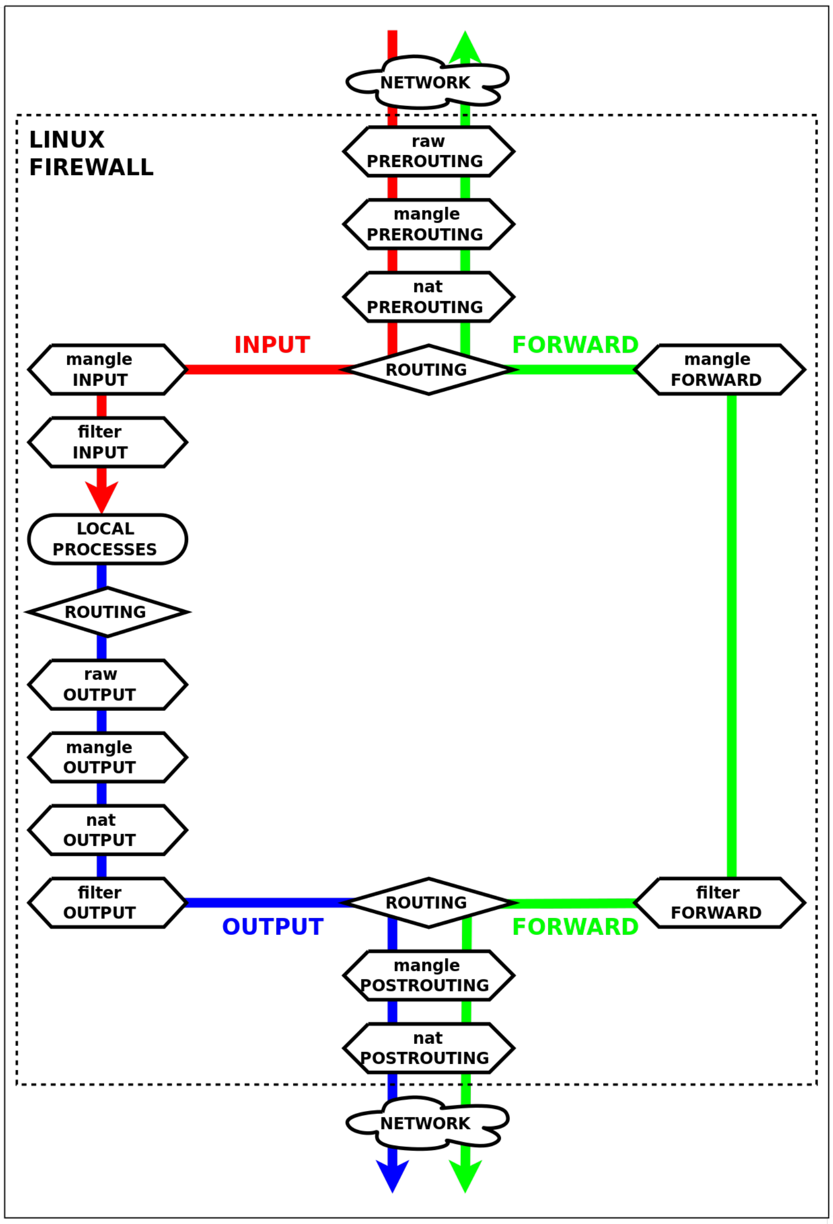

Figure 1: Flow diagram through the `iptables` firewall.

2.3 Forward Stream Packets

Output stream packets are sent by the host that `iptables` is running on.

- The packet first enters one of the firewall network interfaces
- Then enters the `PREROUTING` chain of the `nat` table tipically used for destination network address and port translation (DNAT) operations
- Now it's time to route the packet. Note DNAT operations happens before routing decision and filtering so the packet can be routed and successively filtered accordingly to its "true" destination address. Since we have supposed the packet to belong to the forward stream it will be forwarded
- Then the packet enters the `FORWARD` chain of the `filter` table where all filtering happens.
- If allowed the packet reach the `POSTROUTING` chain of the `nat` table tipically used for source network address and port translation (SNAT) operations. Note that filtering happens

before SNAT operations take place so the packet contains its ”true” source address (after SNAT operations all packets have the same source address so it is useless filter them based on that parameter).

- At the end the packet exits on one of the network interfaces

3 iptables Commands and Usage

As described above, tables contain chains, which contain rutles. Each rule can be inserted in a chain by using an `iptables` command. The main possible operations are:

- To append a rule

```
iptables [--table table-name] --append chain-name matching-criteria --jump target
```

- To delete a rule

```
iptables [--table table-name] --delete chain-name matching-criteria --jump target
```

- To insert a rule in a specific position in the chain

```
iptables [--table table-name] --insert chain-name [rulenum] matching-criteria --jump target
```

- To replace an existing rule with a new one

```
iptables [--table table-name] --replace chain-name rulenum matching-criteria --jump target
```

- To show the content of a chain

```
iptables [--table table-name] --list [chain-name]
```

- To empty a chain

```
iptables [--table table-name] --flush [chain-name]
```

- To create a user-defined chain

```
iptables [--table table-name] --new-chain [chain-name]
```

- To delete an empty user-defined chain

```
iptables [--table table-name] --delete-chain [chain-name]
```

- To change the default policy of a chain

```
iptables [--table table-name] --policy chain-name target
```

- To rename a user-defined chain

```
iptables [--table table-name] --rename-chain old-chain-name new-chain-name
```

3.1 Matching Criteria

Packets can be matched against a wide set of rules, which can be combined to form complex matching criteria tailored to the needs of the network. The most common criteria are:

- The input interface (`eth0`, `wlan0`, etc., and `lo` for the loopback)

  ```
  --in-interface interface-name
  ```

- The output interface (`eth0`, `wlan0`, etc., and `lo` for the loopback)

  ```
  --out-interface interface-name
  ```

- The source IP address

  ```
  --source ip-address
  --source network-address/netmask
  ```

- The destination IP address

  ```
  --destination ip-address
  --destination network-address/netmask
  ```

- The packet protocol (`tcp`, `udp`, `icmp`, etc.)

  ```
  --protocol protocol-name
  ```

  Each protocol has its own criteria:

- For the `icmp` protocol, it is possible to specify the message type (`echo-request`, `echo-reply`, etc.)

  ```
  --protocol icmp --icmp-type message-type
  ```

- For the `tcp` and `udp` protocol, it is possible to specify the source and destination ports:

  ```
  --protocol udp --source-port port-number
  --protocol udp --dest-port port-number
  ```

- TCP also has flags whose values can be specified:

  ```
  --protocol tcp --tcp-flags list-of-examined-flags list-of-set flags
  Example:  --protocol tcp --tcp-flags SYN,ACK,FIN,RST SYN
  ```

  Finally, the conversation state matching criteria are also very useful:

- To allow to match packets starting a new conversation:

  ```
  --match state --state NEW
  ```

- To allow to match packets in an existing and established conversation:

  ```
  --match state --state ESTABLISHED
  ```

- To allow to match packets starting a new conversation related to an established one (e.g., for the FTP protocol):

  ```
  --match state --state RELATED
  ```

It is important to remember that the conversation state has nothing to do with the TCP connection state, and the `RELATED` state does not exist for standard TCP packets.

3.2 Targets

The targets are the actions that `iptables` performs on packets that match the criteria. The most common targets for rules in filtering chains are:

- `ACCEPT` to accept the packet
- `DROP` to silently drop the packet
- `REJECT` to drop the packet and send a message back to the sender

  ```
  --jump REJECT [--reject-with type]
  ```

  The type of the rejection sets the type of message that is sent back, and it can be `icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-port-unreachable`...

- `LOG` to record all packets that match the criteria defined in the rule. In this case, packets don't exit the chain, as `LOG` is a non-terminating target. The `LOG` target allows many options, and the most common are `--log-level` and `--log-prefix`, which allow to control the importance and the prefix of the logged messages.

As mentioned above, `iptables` can also perform Network Address and Port Translation (NAPT). NAPT rules must be contained in the chains contained in the `nat` tables, usually `PREROUTING` and `POSTROUTING`. As previously noted, the rules in `PREROUTING` are normally used to perform DNAT so that the `INPUT` and `FORWARD` chains can act on the "true" destination addresses of the packets, and the opposite goes for `POSTROUTING`, which is used for SNAT. The most common targets in NATting chains are:

- `DNAT` to perform DNAT operations:

  ```
  --jump DNAT --to-destination ipaddr[:port]
  ```

- `DNAT` to perform SNAT operations:

  ```
  --jump SNAT --to-source ipaddr[:port]
  ```

- MASQUERADE to perform NAPT operations when the firewall's public IP is assigned dynamically

By performing SNAT after routing from a private LAN, `iptables` can recognize the different machines on the private network, allowing a more detailed filtering. For example, one machine on the network could be allowed access to the internet, while another one might not; this would be impossible if the NAT operation was performed before the packets reached the `FORWARD` filter chain.