Firewalls

Security 1 (CM0475, CM0493) 2020-21
Università Ca' Foscari Venezia

Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università
Ca'Foscari
Venezia

Motivations

Networking is **complex** and **pervasive**

- Local Area Networks (**LANs**) connecting **PCs**, **servers**, ...
- Wide Area Networks (**WANs**) connecting geographically **distributed** LANs
- **Internet** connectivity
- **Cloud** computing
- Internet of Things (**IoT**), **Industry 4.0**, ...

Motivations

Host-based vs network-based defence

Multitude of Operating Systems.
E.g., Windows, Linux, MacOS, …

Host-based defence: security flaws are fixed on **every** system

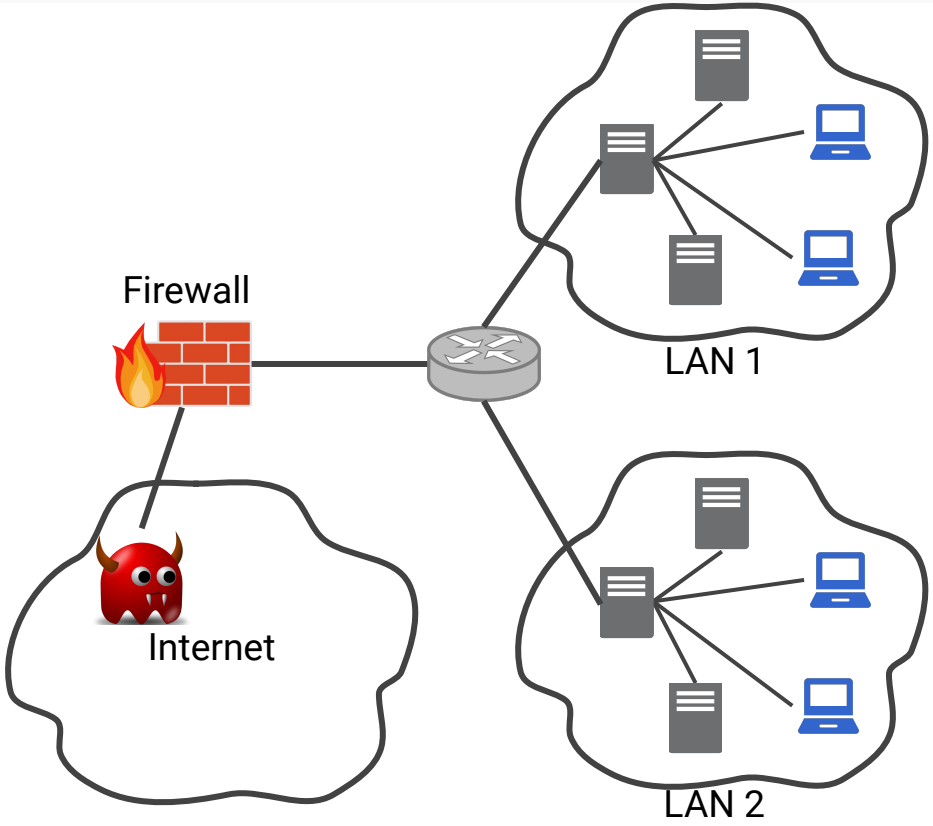
Network-based defence: firewalls prevent attacks to **all systems**

⇒ **Single point** for audit / security
⇒ Extra layer (***defence in depth***)

Firewall characteristics

A firewall **monitors** and **filters** all network traffic:

1. ***Choke point***: all traffic must pass through the firewall
2. ***Policy***: only *authorized* traffic is allowed to pass
3. ***Hardened system***: firewall should be **immune** to attacks



Firewall policies

Address and protocol: filters traffic based on **IPs** and **ports**, **direction** of flow (in/out), and **protocol**. Typically used to limit access to services

Application protocols: usually in the form of an **application-level gateway**
Example: spam filter, Web Application Firewall (WAF)

Identity: filter traffic based on user identity: requires some form of **authentication**
Example: Virtual Private Networks (**VPNs**)

Activity: filter traffic based on **network activity**.
Example: DoS filtering based on the rate of requests

Types of firewalls

- Packet filtering
- Stateful packet filtering
- Application-level gateway
- Circuit-level proxy

Packet filtering firewall

Applies **rules** to packets to decide whether to **accept** or **discard** them

Rules are based on:

IP addresses: packet **source** and **destination** addresses
Example: an internal server which is only reachable from a specific subnetwork

Ports: TCP and UDP port numbers corresponding to **services** (e.g. 80 HTTP, 443 HTTPS, 22 ssh, …)

Protocol: the IP **protocol** (e.g., TCP, UDP, ICMP, …)

Interface: what interfaces the packet is traversing; interfaces indicate what **subnetwork** the packet is coming from / going to

Rules and default policy

Rules are inspected one after the other and when a rule ***matches***, its **action is taken** (accept/discard)

If no rule matches then a **default action** is taken:

- **Default discard**: what is not explicitly accepted is **discarded**
- **Default accept**: what is not explicitly discarded is **accepted**

Default discard policy:

👍 **More conservative**: permitted services are explicitly added
👎 Might **reduce usability**

Default accept policy:

👍 **Not visible** to users
👎 Requires careful **update of rules** to prevent new threats

Packet filtering example

Rule	Src address	Dst address	Protocol	Dst port	Action
1	External	Internal	TCP	80	Accept
2	Internal	External	TCP	80	Accept
3	Any	Any	Any	Any	Discard

Explanation

- Rule 1 accepts **incoming** connection to port 80 (Web)
- Rule 2 accepts **outgoing** connection to port 80 (Web)
- Rule 3 encodes a **default discard policy**, by discarding any packet that does not match rules 1 and 2

Packet filtering example

Rule	Src address	Dst address	Protocol	Dst port	Action
1	External	Internal	TCP	80	Accept
2	Internal	External	TCP	80	Accept
3	Any	Any	Any	Any	Discard

When a browser connects to a server on port 80 its **source port p** is a number assigned dynamically between 1024 and 65535 (< 1024 are “reserved”)

What happens to the **server answer**?

⇒ It is **discarded** at it is directed to port **p ≠ 80** !

Packet filtering example: fix 1

Rule	Src address	Dst address	Protocol	Src port	Dst port	Action
1	External	Internal	TCP	Any	80	Accept
2	Internal	External	TCP	Any	80	Accept
3	Internal	External	TCP	80	Any	Accept
4	External	Internal	TCP	80	Any	Accept
5	Any	Any	Any	Any	Any	Discard

Works but **not so secure** …. why?

Attacker forges a connection with **source port 80** and **bypasses** the firewall!

Packet filtering example: fix 2

Rule	Src address	Dst address	Protocol	Src port	Dst port	Flag	Action
1	External	Internal	TCP	Any	80		Accept
2	Internal	External	TCP	Any	80		Accept
3	Internal	External	TCP	80	Any	ACK	Accept
4	External	Internal	TCP	80	Any	ACK	Accept
5	Any	Any	Any	Any	Any		Discard

ACK flag is automatically set to any “answers” in a TCP connection

⇒ Attacker cannot start **new connections** with **source port 80**

Attacks on packet filtering

Packet filtering is **fast** and **simple** but is subject to the following attacks

Application-layer attacks: packet filtering cannot stop attacks on **application** protocols

Fragmentation attacks: extreme fragmentation might **split headers** so that filtering does not work
Solution: discard connections with excessive fragmentation

IP address spoofing: Attacker sends packets from a spoofed **internal** IP address
Solution: discard packets with source IPs that do not match the interface

Source routing: possibility to **force routing** through a certain host. Used in IP spoofing attacks to get answers
Solution: discard packets with *source routing* enabled

Stateful packet filtering

Packet filtering that keep the **state** of **established** connections.

- Simpler configurations that enable **bidirectional** traffic
- Can identify **related** connections on other ports (by inspecting limited application data); **Example**: VoIP protocols
- Network Address Translation (**NAT**)

Src address	Dst address	Protocol	Src port	Dst port	State	Action
~~Internal~~	~~External~~	~~TCP~~	~~80~~	~~Any~~		~~Accept~~
~~External~~	~~Internal~~	~~TCP~~	~~80~~	~~Any~~		~~Accept~~
Any	Any	TCP	Any	Any	Established	Accept

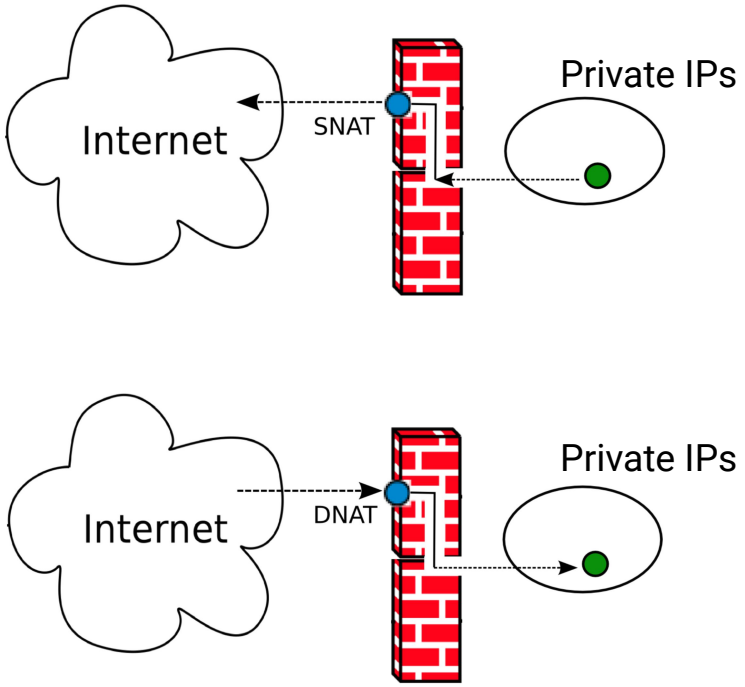
Network Address Translation (NAT)

Network Address Translation (**NAT**): is typically necessary in LANs with private IP addresses

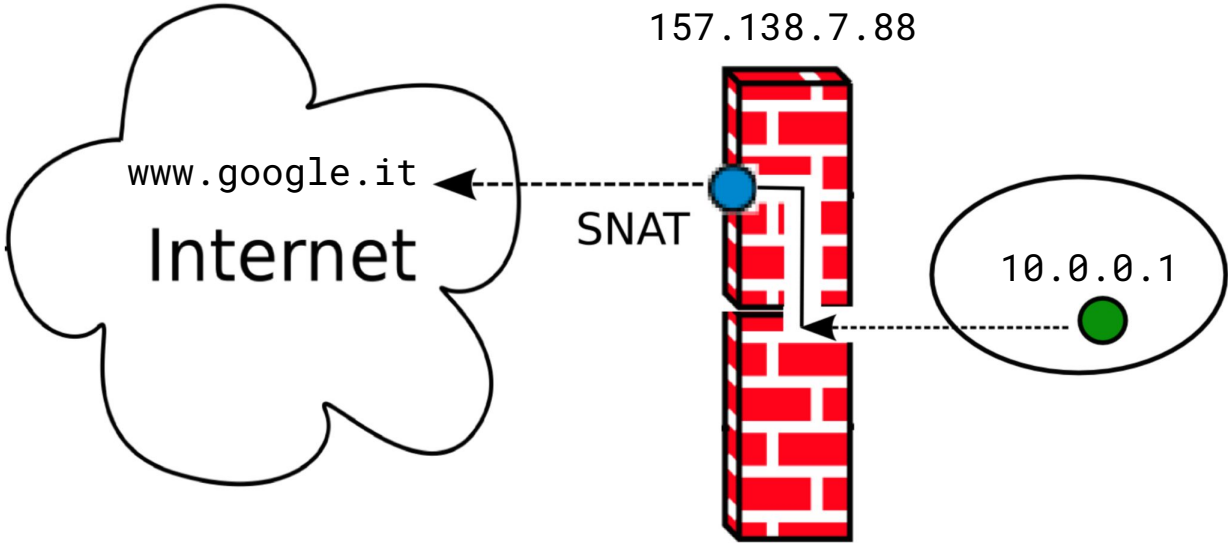
Source NAT: **outgoing** traffic needs a public IP **source** address

Destination NAT: **incoming** traffic needs a public IP **destination** address

NAT can be implemented **transparently** in stateful firewalls

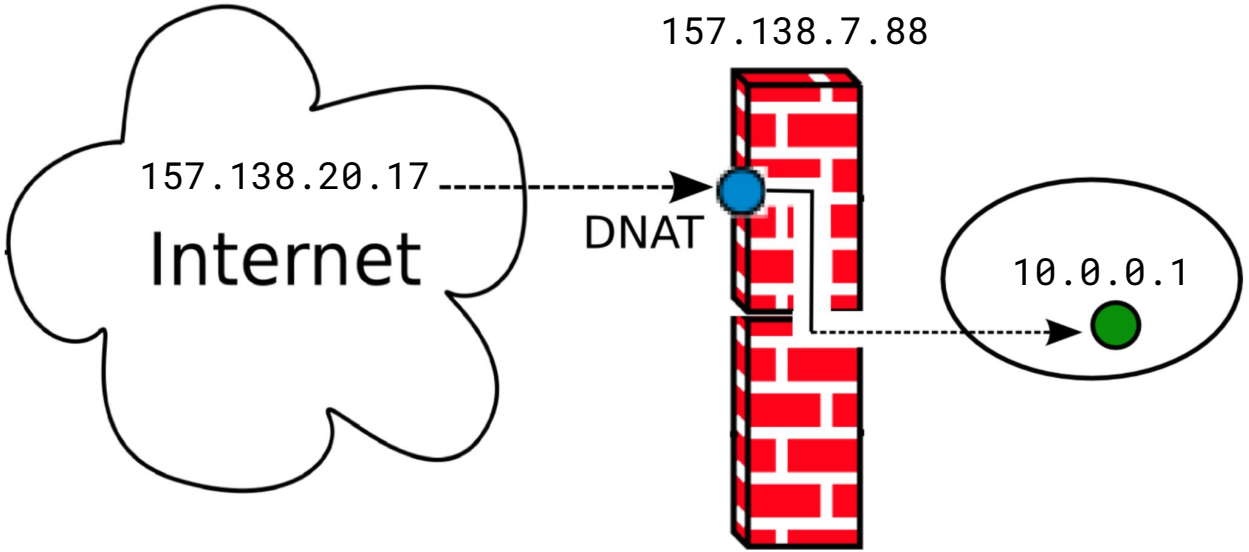


Example: Source NAT



`www.google.it` answers to `157.138.7.88`. The (stateful) firewall transparently **translates** the **destination** address into `10.0.0.1`

Example: Destination NAT



`10.0.0.1` answers to `157.138.20.17`. The stateful firewall transparently **translates** the **source** address into `157.138.7.88`

Application-level gateway (ALG)

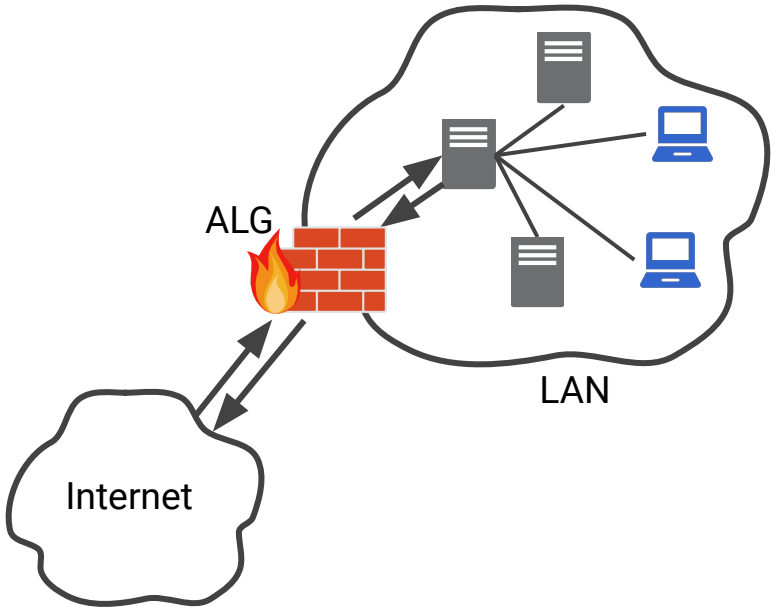
Relay of application-level traffic

1. User **authenticates** to the ALG
2. ALG **connects** to the application
3. ALG **forwards** (part of) user commands to the application

Can **reduce application commands** to a suitable subset

Performs **application-level filtering**
es. Web Application Firewall (WAF)

👎 Introduces processing overhead



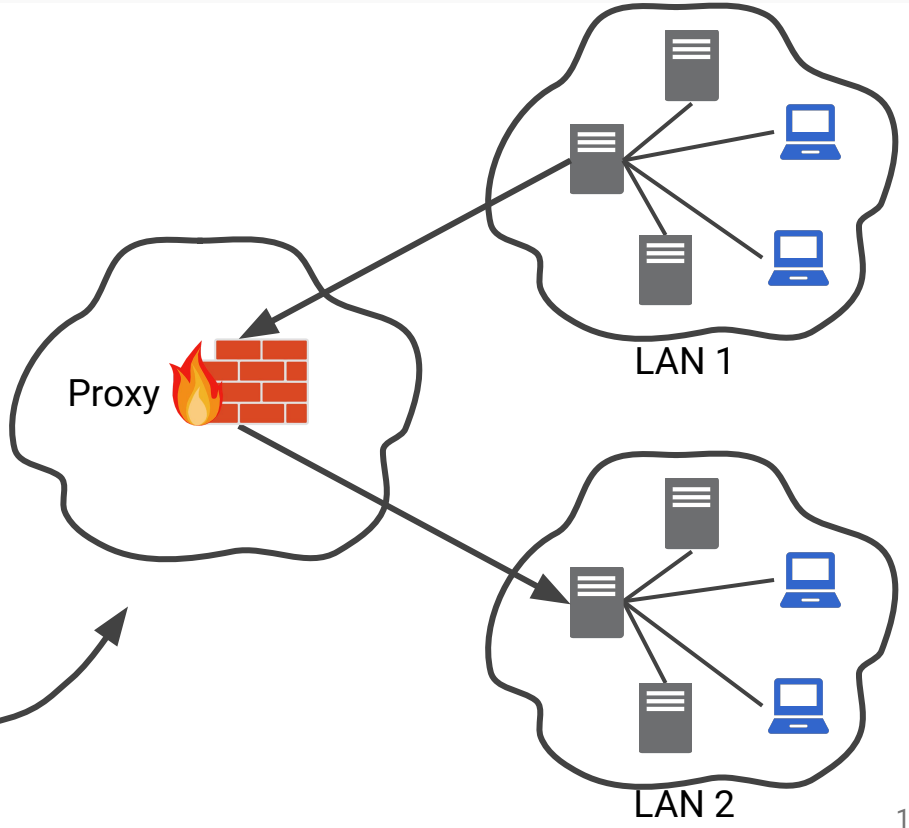
Circuit-level proxy

Relay of TCP/UDP connections

1. User **authenticates** to the proxy
2. Proxy **connects** to the server
3. Proxy **forwards** packets bidirectionally

Once connection is established it does **not** perform filtering (example: SOCKS protocol)

Also used to **hide** IP address



Firewall basing (1)

Bastion host is a **critical network security point**

Usually implements **application-level** and/or **circuit-level** gateways

- **hardened** system
- audit **log**
- **simple** and verified software
- **read-only** file system

Host-based firewall: secures an individual host (e.g, a server)

- filtering tailored to the **specific** applications
- **independent** of network topology (any connection will go through the firewall)
- **extra layer** of protection

Firewall basing (2)

Network device firewall is a firewall on a router or switch

Usually implements **stateful packet filtering**

- **complements** bastion host and host-based firewall
- provides **isolation** between internal LANs

Personal firewall: secures a PC or a home (simple) network

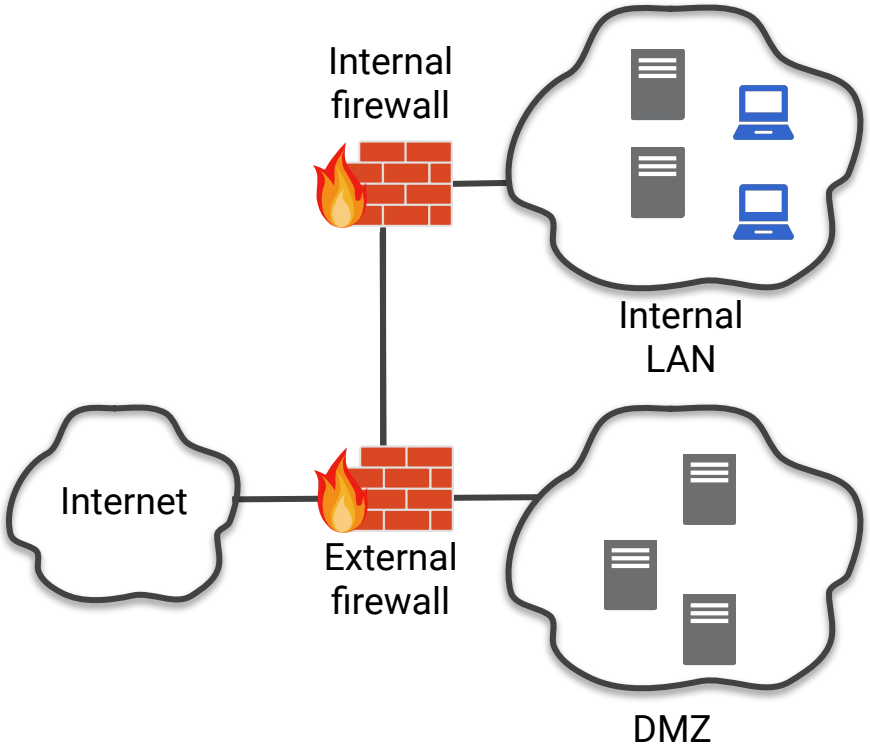
- protect from **external** accesses
- try to detect **suspicious** outgoing traffic (malware)
- usually part of the **operating system**

DMZ networks

Demilitarized Zone (DMZ) is a subnetwork that **needs to be accessed** from the Internet (es. Web and email servers)

DMZ is usually

- protected by an **external** firewall
- **isolated** from internal servers by an internal firewall

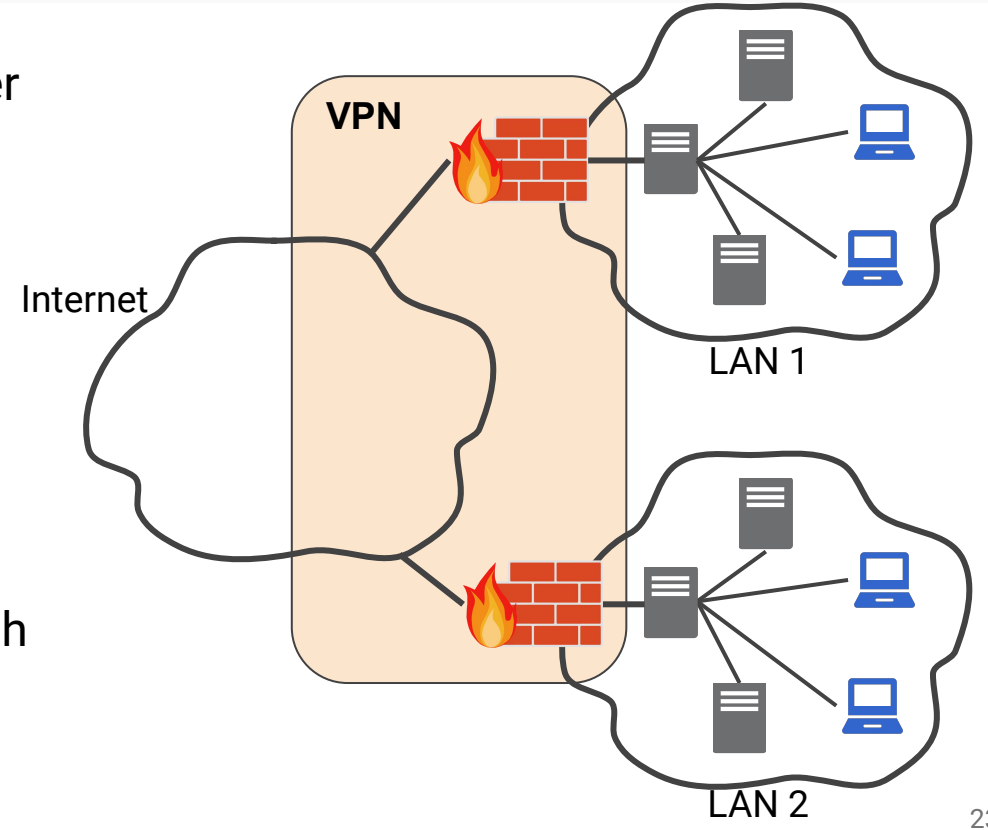


Virtual Private Network (VPN)

VPN is a **secure private network** over an insecure one

Based on **crypto protocols**, for example: IPSec

Multiple (remote) LANs can be **transparently** linked through a VPN: Routers or firewalls **encrypt/decrypt packets** before sending them through the insecure network



Case study: netfilter

Standard **firewall tool** in Linux

Netfilter allows for:

- Packet **filtering**
- Network address **translation** (NAT)
- Packet **mangling** (packet transformation)

Configured through **`iptables`**, a very powerful and flexible tool

`netfilter` is based on ***tables***

Tables group rules depending on the kind of ***action***

The three most commonly used tables are:

- ***filter*** for packet filtering
- ***nat*** for NATs
- ***mangle*** for packet alteration

Chains: lists of rules in netfilter

Chains are lists of rules that are **inspected sequentially**

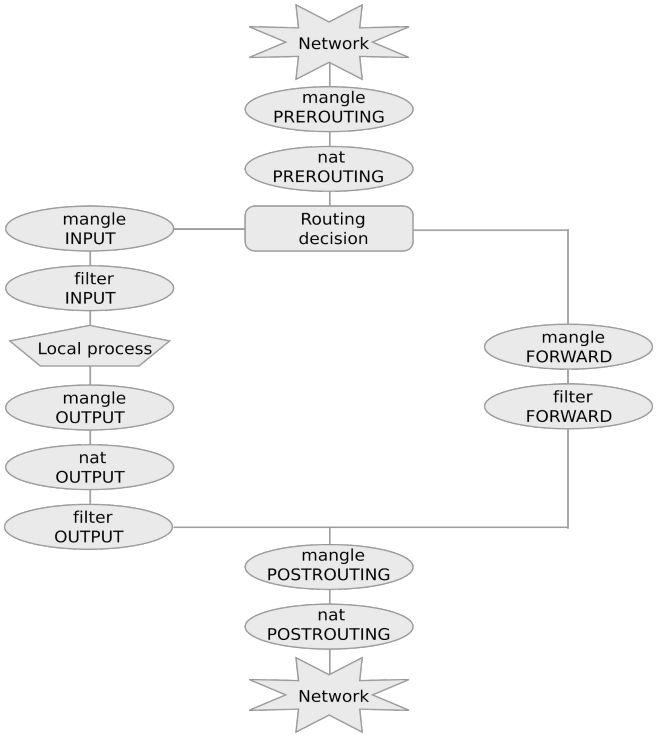
`PREROUTING` p **reaches** the host

`FORWARD` p is **forwarded**

`POSTROUTING` p is about to **leave**

`INPUT` p is **routed to** the host

`OUTPUT` p is **generated** by the host



Rules

Rules (in a chain) are inspected one after the other

- If matched then p is processed along the ***rule target***
- Otherwise the **next rule** in the chain is examined

A **default policy** is triggered if no rule matches

The most commonly used targets are:

- **`ACCEPT`**, for **accepting** the packet
- **`DROP`**, for **discarding** it
- **`DNAT`**, for **destination NAT**
- **`SNAT`** for **source NAT**

Example: list rules and default policy

```
# iptables -t filter -L

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

`-t` specifies the table (filter is the default)

`-L` stands for “list”

Example: blocking incoming traffic but ssh

```
# iptables -A INPUT  -p tcp --dport 22 -j ACCEPT
```

`-A` stands for append
`-p tcp` specifies tcp protocol
`--dport` and `--sport` specify destination and source port
`-j ACCEPT` specifies the **ACCEPT** target

```
# iptables -P INPUT DROP
```

(sets **default policy** for INPUT chain to **DROP**)

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source       destination
  126  8632 ACCEPT     tcp  --  any    any     anywhere     anywhere       tcp dpt:ssh
```

Example: blocking incoming traffic but ssh

What happens if we issue the following?

```
iptables -A INPUT  -p tcp --dport 22 -j DROP
```

⇒ Rules are inspected sequentially: this rule will **never be matched**, since ssh packets will be accepted by the previous one!

Q: Can we connect to a web server (port 80)?

A: Yes, **OUTPUT** policy is **ACCEPT**

Q: What happens to the server **answer**?

A: Answer is **dropped**! Firewall only admits **ssh** incoming connections

Stateful filtering

`netfilter` tracks connections:

- when a new connection starts the packet has state **NEW**
- packets belonging to the same connection has state **ESTABLISHED**
- some protocols start new connections (e.g. ftp). These packets have state **RELATED**
- Network Address Translation is also tracked (NAT)

```
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

⇒ both **ssh** and **established** incoming packets will be accepted