Lab 6 - Firewall

Kathara Firewalls.pdf

Firewall: what it is

Network security system that monitors, controls, filters and modifies incoming and outgoing network traffic based on predetermined security rules.

The backend lib in linux is **NETFILTER,** the **userspace for controlling it is iptables** that manage the NETFILER hooks (entry points of linux kernel that allows pkt managing). ipatables follow the structure: **MATCHING RULE + CONFIGURED ACTIO**N (ex: protocol+accept)

IPtables

Filter

Function: Packet filtering

- **Packet Transformation Chains**:
  - **FORWARD**: Filters packets to servers accessible by another NIC (Network Interace Card) on the firewall.
  - **INPUT**: Filters packets destined to the firewall.
  - **OUTPUT**: Filters packets originating from the firewall.

Nat (Network Address Translation)

Function: Address translation occurs before routing.

- **Packet Transformation Chains**:
  - **PREROUTING**: Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used for **Destination NAT (DNAT)**.
  - **POSTROUTING**: Translates the source IP address after routing (Source NAT or SNAT).
  - **OUTPUT**: Network address translation for packets generated by the firewall (rarely used in SOHO environments).

Mangle

Function: TCP header modification.

- **Packet Transformation Chains**:

- **PREROUTING**: Modifies packet headers before routing.
- **POSTROUTING**: Modifies packet headers after routing.
- **OUTPUT**: Modifies headers for packets originating from the firewall.
- **INPUT**: Modifies headers for packets destined to the firewall.
- **FORWARD**: Modifies headers for packets being routed through the firewall.

IPtables: rules

Flag for iptables command

- **-p / --protocol**: Specifies the protocol of the packet (e.g., tcp, udp, icmp); can be negated with**!**
- **-s / --source**: Source IP address (with optional /mask).
- **-d / --destination**: Destination IP address (with optional /mask).
- **-i / --in-interface**: Specifies the input interface for the packet; valid for INPUT, FORWARD, PREROUTING chains.
- **-o / --out-interface**: Specifies the output interface for the packet; valid for FORWARD, OUTPUT, POSTROUTING chains.
- **-j** : Specifies the target action for matching packets (e.g., ACCEPT, DROP, REJECT).

Append, Delete, Insert Rules

- **Append (-A)**: Aggiunge una regola alla fine della catena.

```
iptables [-t table] -A chain rule-specification
```

- **Delete (-D)**: Elimina una regola da una catena (usando il numero della regola o la specifica).

```
iptables [-t table] -D chain rulenum
```

- **Insert (-I)**: Inserisce una regola in una posizione specifica nella catena.

```
iptables [-t table] -I chain [rulenum] rule-specification
```

Flush, List Rules

- **Flush (-F)**: Rimuove tutte le regole da una catena o da tutte le catene.

```
iptables [-t table] -F [chain]
```

- **List (-L)**: Elenca tutte le regole di una catena.

```
iptables [-t table] -L [chain] [options...]
```

Create, Delete, Rename Chains, and Set Default Policy

- **Create (-N)**: Crea una nuova catena personalizzata.

```
iptables [-t table] -N chain
```

- **Delete (-X)**: Elimina una catena personalizzata (deve essere vuota).

```
iptables [-t table] -X [chain
```

- **Rename (-E)**: Rinomina una catena.

```
iptables [-t table] -E old-chain-name new-chain-nam
```

- **Set Default Policy (-P)**: Imposta una politica predefinita per una catena (es. ACCEPT, DROP).

```
iptables [-t table] -P chain target
```

Example of commands

- **Clean any existing rule on the filter table**:

```
iptables -F
```

- **Accept forwarding from LAN to everywhere**:

```
iptables -A FORWARD -s 192.168.1.0/24 -j ACCEPT
```

- **Accept forwarding from internet to DMZ**:

```
iptables -A FORWARD -i eth2 -d 10.0.0.0/24 -j ACCEPT
```

- **Accept forwarding from pc1 to LAN**:

```
iptables -A FORWARD -s 10.0.0.20 -d 192.168.1.0/24 -j ACCE
PT
```

- **Default policy DROP for forwarding packets**:

```
iptables -P FORWARD DROP
```

Rule order

We want to drop input packets directed **to the firewall itself** coming from DMZ, except for pc1.

```
iptables -A INPUT -s 10.0.0.0/24 -j DROP
iptables -A INPUT -s 10.0.0.20 -j ACCEPT
```

This does not work!

The second rule is appended to the end of the INPUT chain. Therefore, the first rule is executed before the second one. When a packet arrives from `10.0.0.20`, it

matches the first rule and gets dropped.

Rules are matched in order!

Protocol matching

Accept input DNS packets from LAN (DNS runs on udp port 53):

```
iptables -A INPUT -s 192.168.1.0/24 -p udp --dport 53 -j ACCE
PT
```

Accept input HTTP packets from LAN (HTTP runs on tcp port 80):

```
iptables -A INPUT -s 192.168.1.0/24 -p tcp --dport 80 -j ACCE
PT
```

Accept ICMP traffic (e.g., pings) **from anywhere**:

```
iptables -A INPUT -p icmp -j ACCEPT
```

Notes:

- `--dport` matches the destination port.
- `--sport` matches the source port.
- `--tcp-flags` matches TCP header flags.
- `--syn` matches packets with the SYN flag set

Matching rules examples

forward SSH traffic

Allow forwarding of SSH traffic from DMZ to Internet:

```
iptables -A FORWARD -s 10.0.0.0/24 -o eth2 -p tcp --dport 22
-j ACCEPT
iptables -A FORWARD -i eth2 -d 10.0.0.0/24 -p tcp --sport 22
-j ACCEPT
```

- **iptables -A FORWARD -s 10.0.0.0/24 -o eth2 -p tcp --dport 22 -j ACCEPT**: Allows TCP traffic **originating from the 10.0.0.0/24 network**, **routed through the eth2 output interface**, **destined for TCP port 22 (SSH)**, and accepts and forwards the packets if all conditions are met.
- **iptables -A FORWARD -i eth2 -d 10.0.0.0/24 -p tcp --sport 22 -j ACCEPT**: Allows TCP traffic **entering eth2**, **destined 10.0.0.0/24 network**, **originating from TCP port 22 (SSH)** on the source, and accepts and forwards the packets if all conditions are met.

We need two iptables rules, because we have to match the SSH packets going from DMZ to Internet and the ones going from Internet to DMZ (so ingoing and outgoing traffic of the ssh communication, otherwise “only one will speak”)

Here’s the textual version of the images, structured similarly to the format above, with explanations.

Other way with ssh

```
iptables -A FORWARD -s 10.0.0.0/24 -o eth2 -p tcp --dport 22
-j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j A
CCEPT
```

1. **iptables -A FORWARD -s 10.0.0.0/24 -o eth2 -p tcp --dport 22 -j ACCEPT**:Allows TCP traffic **originating from the 10.0.0.0/24 network**, routed through the **eth2** output interface, destined for **TCP port 22 (SSH)**. This rule specifically matches and accepts the first SSH packets initiated by a host in the **DMZ**.

2. **iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT**: Allows any packets belonging to an **already established connection** or related to one (e.g., TCP acknowledgment packets). This ensures the return traffic (e.g., packets from the Internet back to the DMZ) is also forwarded.

NAT (Network Address Translation)

What is it

NAT (Network Address Translation) is a process where a router replaces a private source IP address with its public IP address when sending packets to the internet. The router reverses this process for incoming packets, forwarding them to the correct private address.
It work on that way (assume private IP: `192.168.1.15` (not routable), Router's public IP): `203.0.113.1`.

1. You send a packet to a website. The router replaces **192.168.1.15** with **203.0.113.1**.
2. The website replies to **203.0.113.1**. The router replaces the destination IP back to **192.168.1.15** and delivers it to you.

This allows devices in a private LAN to communicate with the internet using a single public IP.

NAT prevents new connections from being initiated from outside the LAN, protecting hosts from exposure to the public internet.

However, if an attack starts from inside the LAN, NAT does not provide protection.

For example, a malicious program can be installed on a PC within the LAN. This program initiates a connection to a remote malicious host on the internet, and NAT allows the connection because it originates from inside the LAN. Once the connection is established, it becomes bidirectional, allowing the remote host to communicate directly with the PC.

Source NAT with iptables

Purpose: Source NAT (SNAT) modifies the **source IP address** of outgoing packets.

- **Command**: `iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE`
- **Explanation**:
  - `t nat`: Specifies the NAT table.
  - `A POSTROUTING`: Applies the rule after the routing decision has been made.
  - `o eth2`: Targets packets exiting through the **eth2** network interface.
  - `j MASQUERADE`: Replaces the source IP address with the firewall's IP address.

This allows devices in a private LAN to communicate with external networks by masquerading the private IPs.

Additional Rule:

To block **new connections** coming from the outside:

```
iptables -A FORWARD -i eth2 -m state --state NEW -j DROP
```

Destination NAT (DNAT)

Purpose: Destination NAT modifies the **destination IP address** of incoming packets, often used for port forwarding.

- **Command**: `iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 80 -j DNAT --to-destination 10.0.0.100:8080`
- **Explanation**:
  - `t nat`: Specifies the NAT table.
  - `A PREROUTING`: Applies the rule before the routing decision.
  - `i eth2`: Targets packets entering through the **eth2** network interface.
  - `p tcp --dport 80`: Matches incoming HTTP traffic on port **80**.
  - `j DNAT`: Redirects the packets.
  - `-to-destination 10.0.0.100:8080`: Forwards packets to **10.0.0.100** on port **8080**.

Example: Port forwarding on a home router to redirect HTTP traffic (port 80) to a web server listening on port 8080.

Basic iptables Troubleshooting

Purpose: To identify rules that are dropping or affecting traffic.

- **Command**: `iptables -t mytable -L -v -n`
  - `t mytable`: Specifies the table to inspect (e.g., `filter`, `nat`).
  - `L`: Lists all rules in the table.
  - `v`: Provides verbose output, showing packet counters.
  - `n`: Displays IP addresses and ports numerically (no DNS resolution).

Usage:

This command helps identify how many packets match specific rules, making it easier to troubleshoot which rule is dropping traffic. The packet counters help pinpoint issues in the firewall configuration.