Network Security and Firewalling

Network security

Vulnerabilities	protocol stack		Countermeasures
DoS Sniffing MiM passive MiM active spoofing	application	application	application gateway, nids, authentication, cryptography
	presentation		
	session	TCP	**stateful firewall**, nids, cryptography
	transport		
	network	IP	**screening router**, nids nat, cryptography
	link	link	vlan, switch configuration, authentication, cryptography
	physical	physical	isolation, cryptography

Network vulnerabilities

- The network could be exploited as vector for the attack
  - The attack could start in a remote physical location
  - The human factor is crucial: email, web, etc.
- The network protocols could have vulnerabilities in their logic
  - Non-authenticated protocols
  - Non-encrypted protocols
  - DoS
  - DDoS

Firewalls

- Network security device or software that acts as a barrier between two (or more) domains
- Could be of two types depending of the packet structure they analyze:
  - Network firewall: layer3+4
    - stateful vs. stateless
  - Application firewall: layer7
    - a.k.a deep packet inspection o applicative content inspection
- Could have a lot more functionalities
  - Often called Unified Thread Management (UTF)

Unified Threat Management (UTM)

- Evolved firewall
- Many functionalities in a single device
  - firewall
  - network intrusion detection/prevention
  - antivirus
  - anti-spam
  - VPN termination
  - applicative content inspection
  - load balancing

stateless firewall

- “packet filtering” rules
  - based on the headers of the L4 packet
    - tuple <saddr,sport,daddr,dport>
- They do not mantain any state but filter some specific rules
  - e.g., drop all packets with protocol UDP
  - e.g., drop all packets with protocol UDP and destination port 53

stateful firewall

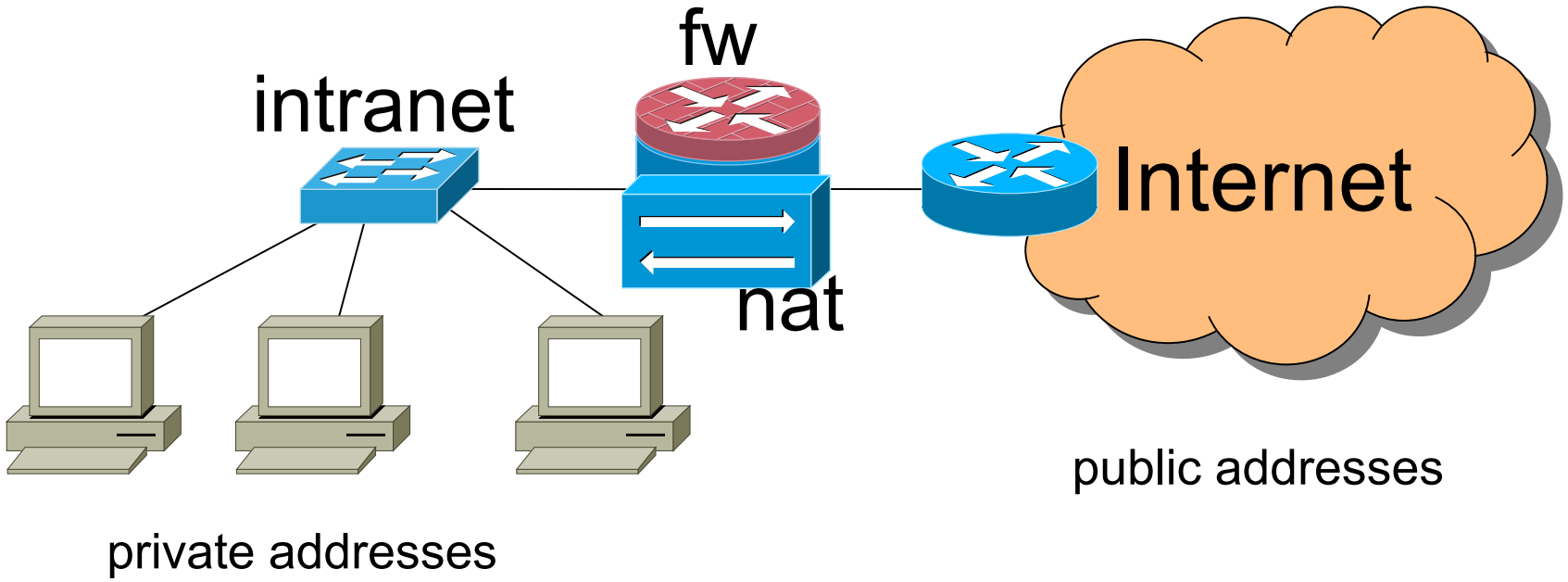
- Stateful firewalls keep track of the connection status
  - TCP is implemented with sequence numbers
  - UDP is implemented just with the tuple <saddr,sport,daddr,dport>
- Able to filter on the connection state
  - NEW: only the first packet of the connection
    - e.g., the TCP SYN packet
  - ESTABLISHED: all the other packets of the connection

Basic policy

- If the first packet (connection state: NEW) is accepted all the other packets of the same connection (connection state: ESTABLISHED) should be accepted
  - If this rule is not set, it is easy to lose access to a server ☺

firewall, nat e intranet

- Use case:
  - Internal network shielded from the Internet by the firewall
  - S-NAT to allow private addresses inside the intranet

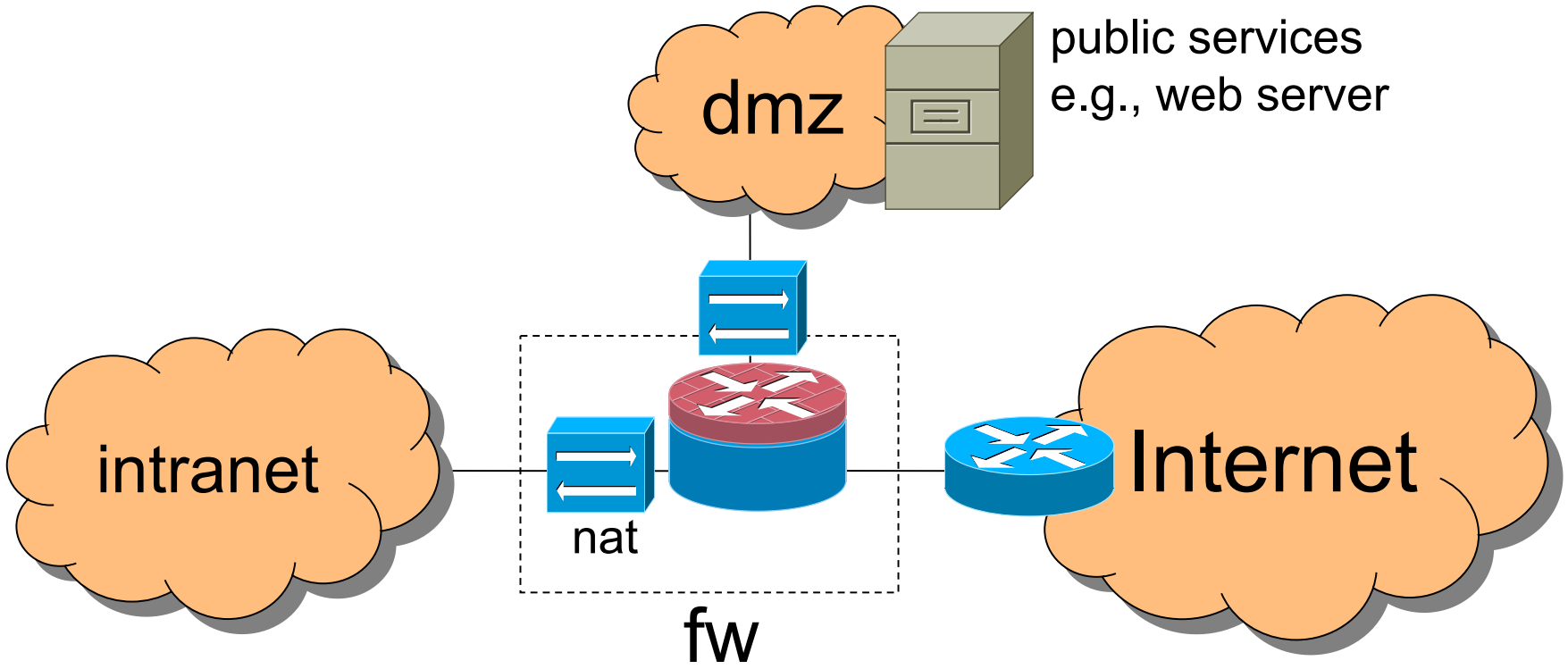


Basic firewall configuration

- Two network interfaces
  - One towards the (trusted) internal network
  - One towards the (untrusted) external network
- Rules:
  - Outbound traffic is always allowed
    - **This allows to establish new connections**
  - Only inbound traffic related to ESTABLISHED connections is allowed
  - S-NAT to change all the source IPs to the IP of the firewall/gateway
- This can be considered a safe default

intranet + DMZ

- Servers are located inside a third network area, separated from the rest.
- DMZ: demilitarized zone or perimeter network
  - Firewall regulates accesses to the DMZ according to the specified policies



DoS and DDoS network attacks

Denial of Service Attacks

- A Denial-of-Service (DoS) attack is a malicious attempt to disrupt the normal functioning of a targeted server, service, or network by overwhelming it with a flood of illegitimate traffic.
- The goal of a DoS attack is to make the targeted system or network unavailable to its intended users, causing a denial of service.

Example of DoS Attacks

- Flooding Attacks
  - TCP/IP Flooding
  - UDP Flooding
  - ICMP Flooding (Ping Flood)
- Logic Attacks
  - Application Layer Attacks
  - SYN/ACK Attacks
- Amplification Attacks

SYN-flood e SYN-proxy

- **SYN-flood**: flood target with SYN packets with source address **spoofing**
- **SYN-proxy**: a firewall to protect from SYN flooding
- Basic behaviour:
  - When a SYN is received, the firewall replies back, but do not forward the SYN to the server it was destinated to
  - Then, when the ACK is received, the firewall establishes the TCP session to the server and begin proxing the two TCP connections
  - The firewall scales well on all the SYN received
    - It just stores IPs and sequence numbers

Slowloris DoS Attack

- Slowloris is a specific attack designed to allow a single machine to take down a server without using a lot of bandwidth.
- It overwhelms a targeted server by opening and maintaining many simultaneous HTTP connections between the attacker and the target.
- Since the webservers usually dedicate a thread to each connection, the attack is targeted at topping the number of threads available on the server so it cannot handle more requests
- Each connection is slowed down (there are many parameters that helps slowing down a TCP connection) so each response takes a lot of time to complete

Amplification Attacks

- Amplification attacks are a type of cyberattack in which an attacker exploits vulnerabilities in certain network protocols or services to generate a larger volume of data and direct it toward a target.
- Amplification attacks take advantage of the fact that a small request can trigger a disproportionately larger response.
- Two common types of amplification attacks are:
  - **DNS Amplification**
  - **NTP Amplification**

Distributed DoS Attacks

- Unlike a traditional Denial-of-Service (DoS) attack, which is conducted from a single source, a DDoS attack involves a distributed network of devices, often referred to as a botnet.
- Key characteristics of DDoS attacks include:
  - Botnets
  - Traffic Volume
  - Variety of Attack Vectors
  - IP Spoofing
  - Duration and Persistence

Firewalls and DDoS Attacks

- DDoS attacks saturate the network bandwidth
  - Firewalls might not be able to help
    - The target of DDoS is the whole network infrastructure.
    - Once traffic reaches the firewall, it's too late
  - anycast and CDNs can offer mitigations against DDoSs but it can be expensive
- DDoS against server resources (if **no spoofing is done)**
  - keep track of session activity
    - aging and number of open sessions
    - However, this can become an attack vector
  - Mitigate using **white/black-lists**
    - A sample pool of unknown IPs are monitored
    - IP addresses which keep an unjustifiable number of sessions open for too long are blacklisted
    - Known IPs are whitelisted
      - Use *bloom filters*

Linux Netfilter

Linux Netfilter

- Netfilter is a framework within the Linux kernel that provides hooks or entry points for various networking-related operations.
  - Packet filter
  - Stateful firewall
  - NAT
  - And much more…

Linux Netfilter: Tables

- Tables are organizational structures that group rules based on the type of packet processing they define.
- Each table is associated with a specific type of packet handling, and within each table, there are chains that represent different points in the packet's traversal through the network stack.
- The main Netfilter tables are:
  - filter
    - chains: INPUT, OUTPUT, FORWARD
  - nat
    - chains: PREROUTING, POSTROUTING, OUTPUT
  - mangle (packet marking)
    - chains: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING
- Each table is defined by a Kernel module

Linux Netfilter: Chains

- Each Netfilter table has a sequence of chains
- There are default chains but the user can create other (user-defined) chains
- Each chain contains rules
- Each chain has a default **target**, if a packet do not match any rule

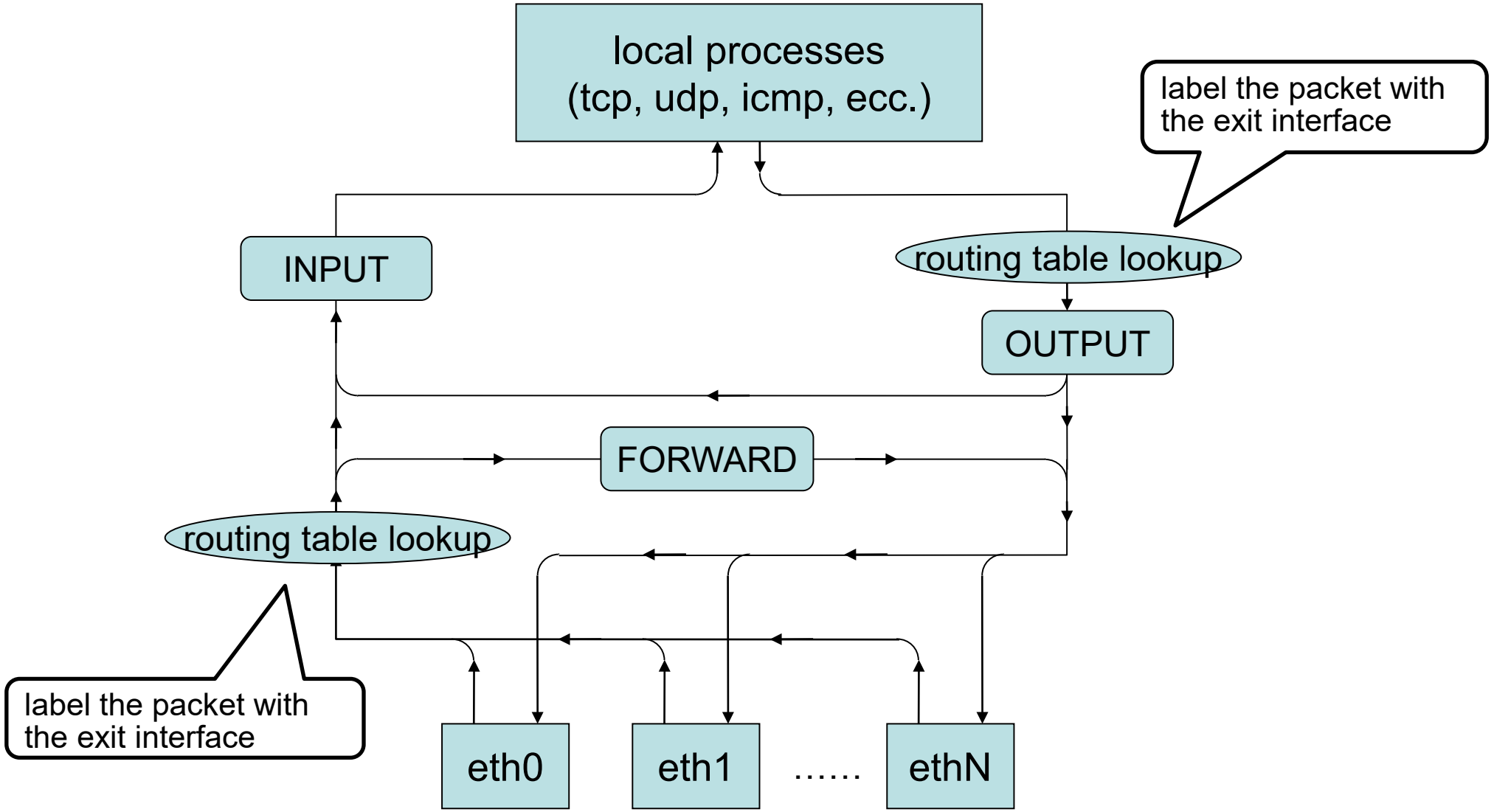
Linux Netfilter: Rules

- Each rule has
  - some **parameters**, to match some packets
  - a **target**, that specify what to do with the matched packet
- Main parameters (many other exists)
  - protocol (-p ip/tcp/udp/icmp)
  - source/destination address (-s/-d [!]x.x.x.x/x)
  - source/destination port (--sport/--dport port)
  - input/output interface (-i/-o ethN)
  - tcp flags syn=1 e ack=0 (--syn)

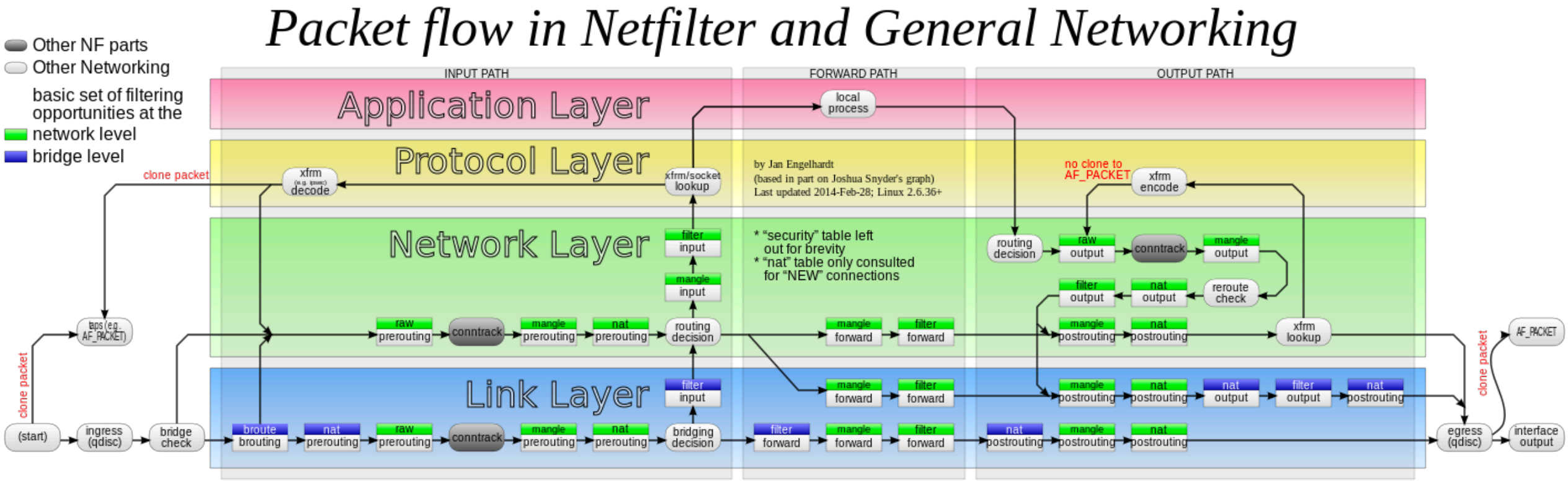
Linux Netfilter: Targets

- Possible targets
  - ACCEPT
  - DROP
  - REJECT (drop packet and send ICMP to sender)
  - <chain name> (jump to specified chain, useful with user-defined chains)
  - RETURN (return to the calling chain)

Linux Netfilter: a simple flow graph



Linux Netfilter: the complete flow graph



Linux Netfilter: Commands

- iptables
  - manages the running configuration
- iptables-save
  - print the running configuration to stdout
- iptables-restore
  - loads the configuration from a dump made with iptables-save

Linux Netfilter: Commands

- iptables if not specified uses the default table (filter)
  - -t <table> use table <table>
  - -L show the running configuration
  - --flush delete the running configuration
  - -A <chain> append to <chain> a rule
- All parameters can be found in the man page

Linux Netfilter: Extended modules

- It is possible to extend Netfilter with modules to extend its basic capabilities
- With option -m is possible to use modules
  - **state**: NEW, ESTABLISHED, RELATED, INVALID
  - **conntrack:** similar to state but extended
  - **mac**: filter basing on Layer2 MAC Addresses
  - **limit**: filter basing on the frequency of arriving packets
    - anti DoS, anti scanning ecc.
  - **iprange**: es. 192.168.1.13-192.168.2.19
  - many more
- With the additional modules, Netfilter is able to match quite every header in the layer 4 packet

Linux Netfilter: Connection Tracking

- One of the most used modules is called Connection Tracking
- This module keeps track of:
  - protocol
  - for tcp e udp: tuple {<src-addr, src-port>, <dst-addr, dst-port>}
  - for icmp echo: tuple {src-addr, dst-addr}
- A packet can be in state
  - NEW: first packet of the connection (e.g., TCP SYN)
  - ESTABLISHED: packet in the middle of a connection
  - RELATED: new connection related to another one established (ftp, icmp errors)
  - INVALID: impossible to match a connection due to the packet contents (e.g., icmp error)

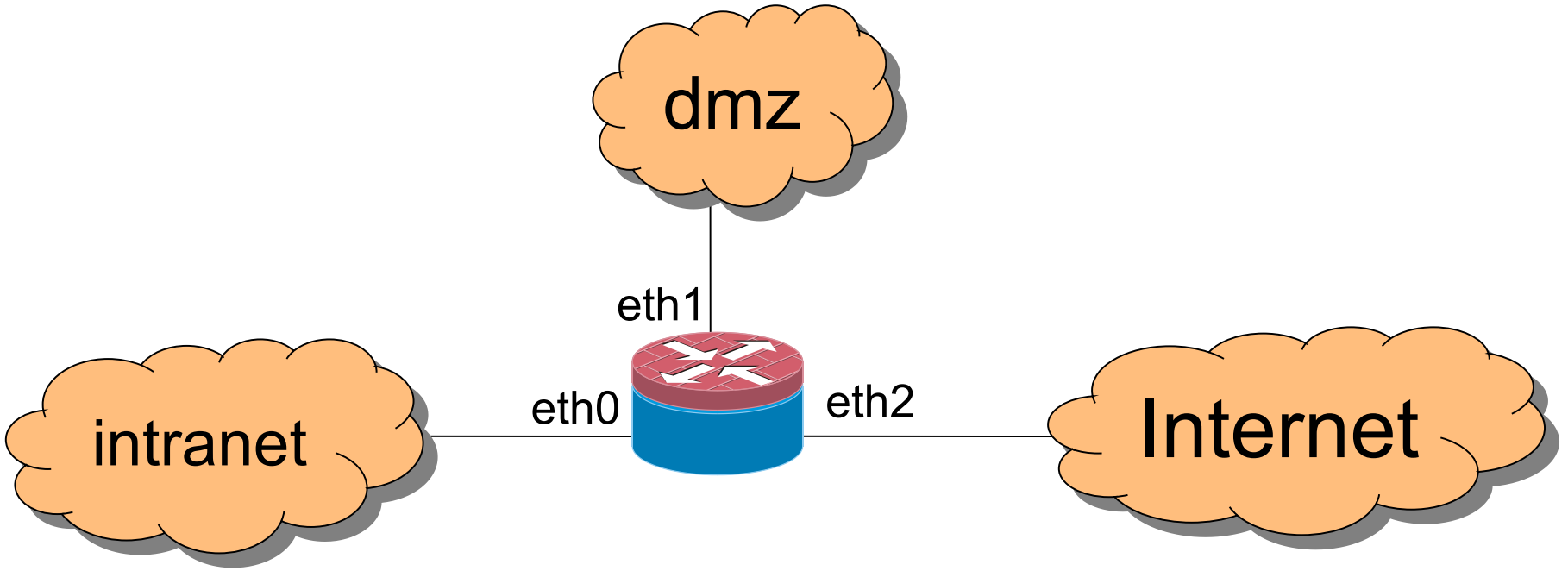
Linux Netfilter: Default Configuration

```
user@vulnbox:# iptables --flush
user@vulnbox:# iptables-save
# Generated by iptables-save v1.8.9 (nf_tables) on Tue Nov 21 20:05:19 2023
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# Completed on Tue Nov 21 20:05:19 2023
```

Linux Netfilter: Basic Configuration

```
user@vulnbox:# iptables --flush
user@vulnbox:# iptables-save
# Generated by iptables-save v1.8.9 (nf_tables) on Tue Nov 21 20:05:19 2023
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Tue Nov 21 20:05:19 2023
```

Linux Netfilter: Basic DMZ




```
*filter
:INPUT ACCEPT [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -i eth0 -o eth1 -m state --state NEW -j ACCEPT
-A FORWARD -i eth0 -o eth2 -m state --state NEW -j ACCEPT
-A FORWARD -i eth2 -o eth1 -m state --state NEW -j ACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Linux Netfilter: nftables

- Nftables is a newer and more unified framework that aims to replace the older tools associated with Netfilter
- nftables is built on top of the Netfilter framework.
  - It does not replace Netfilter but rather provides a more modern and unified way to interact with it.
- The command `iptables` is being replaced by `iptables-nft`
- `iptables-nft` will be the new command to configure Netfilter
- nftables supports a more consistent syntax and data model compared to the various tools it replaces.
- It allows users to define rulesets that are applied to packets, similar to iptables but with a more versatile and expressive language.