# Practical Network Defense

*Master's degree in Cybersecurity 2020-21*

# Network traffic regulation
# NAT

*Angelo Spognardi*
*spognardi@di.uniroma1.it*
*Dipartimento di Informatica*
*Sapienza Università di Roma*

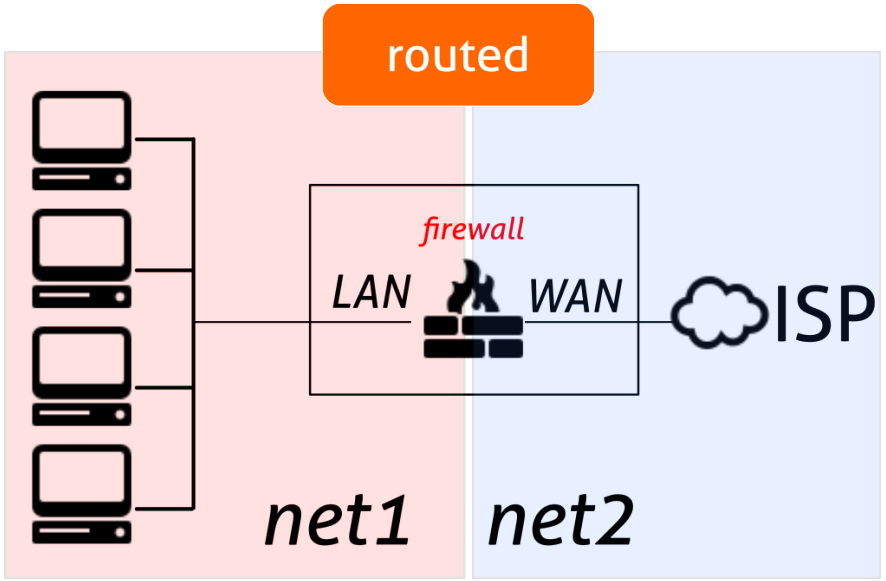
# Today's agenda

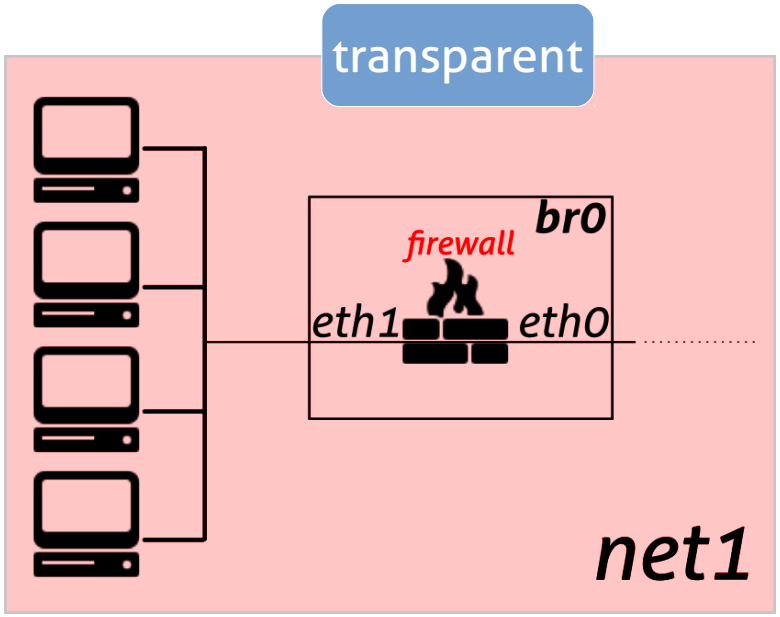
- NAT
- iptables
  - Tables
  - Chain priorities
  - Logging


# NAT
# Network Address Translation

# Routed vs Transparent firewall modes

- In routed mode, a firewall is a hop in the routing process
  - It IS a router responsible of its own (internal) networks
- In transparent mode, a firewall works with data at Layer 2 and is not seen as a router hop to connected devices
  - It can be implemented using bridged NICs



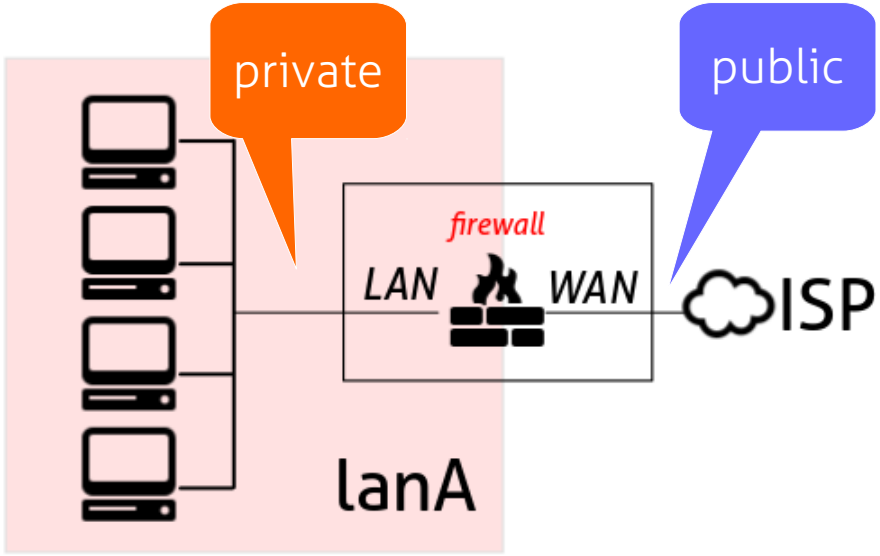
# Routable IP Addressing

- Routable addresses need to be unique on the Internet to be PUBLICLY reachable → **public IP addresses**
- Non-routable addresses → **Special-Purpose Address IP addresses**
  - Private addresses (https://www.iana.org/go/rfc1918):
    - 10.0.0.0 - 10.255.255.255 (10/8 prefix)
    - 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
    - 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)
  - Loopback addresses (https://www.iana.org/go/rfc1122):
    - 127.0.0.0–127.255.255.255 (127/8 prefix)
  - Shared address space (https://www.iana.org/go/rfc6598):
    - 100.64.0.0–100.127.255.255 (100.64/10 prefix)
  - Full list: iana website

# Network Address Translation (NAT)

- Translate the address (f.e.: between incompatible IP addressing)
- Informally speaking, connecting to the Internet a LAN using un-routable in-house LAN addresses

- NAT in a routed firewall:
  - Can filter requests from hosts on WAN side to hosts on LAN side
  - Allows host requests from the LAN side to reach the WAN side
  - Does not expose LAN hosts to external port scans



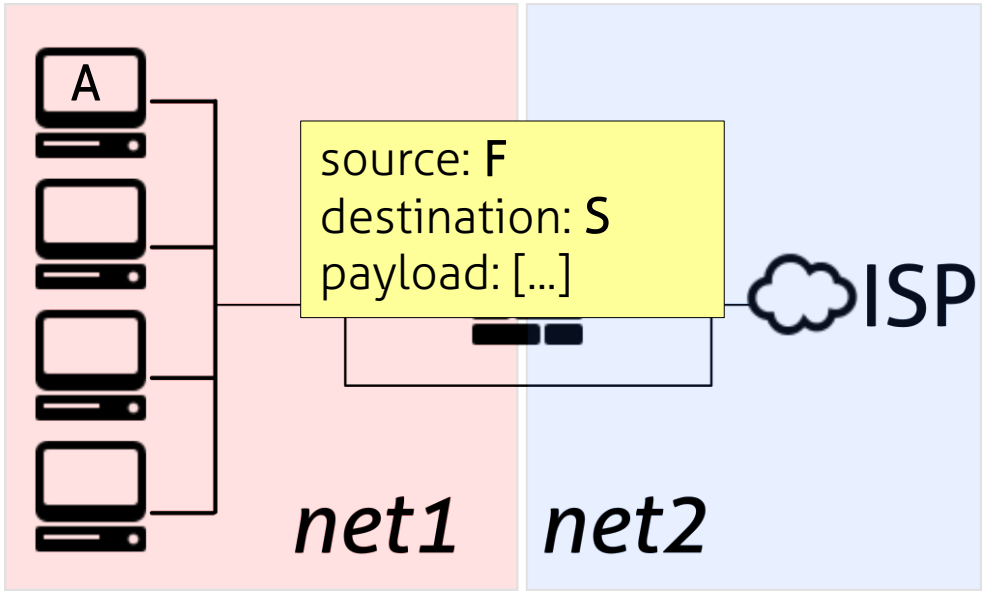
# NAT goals

- A private network uses just one IP address provided by ISP to connect to the Internet
- Private networks use private IP addresses provided by IETF
- Can change address of devices in private network without notifying outside world
- Can change ISP without changing the address of devices in private network
- Devices inside private network are not explicitly addressable by external network, or visible by outside world (a security plus)

# Source NAT (SNAT)

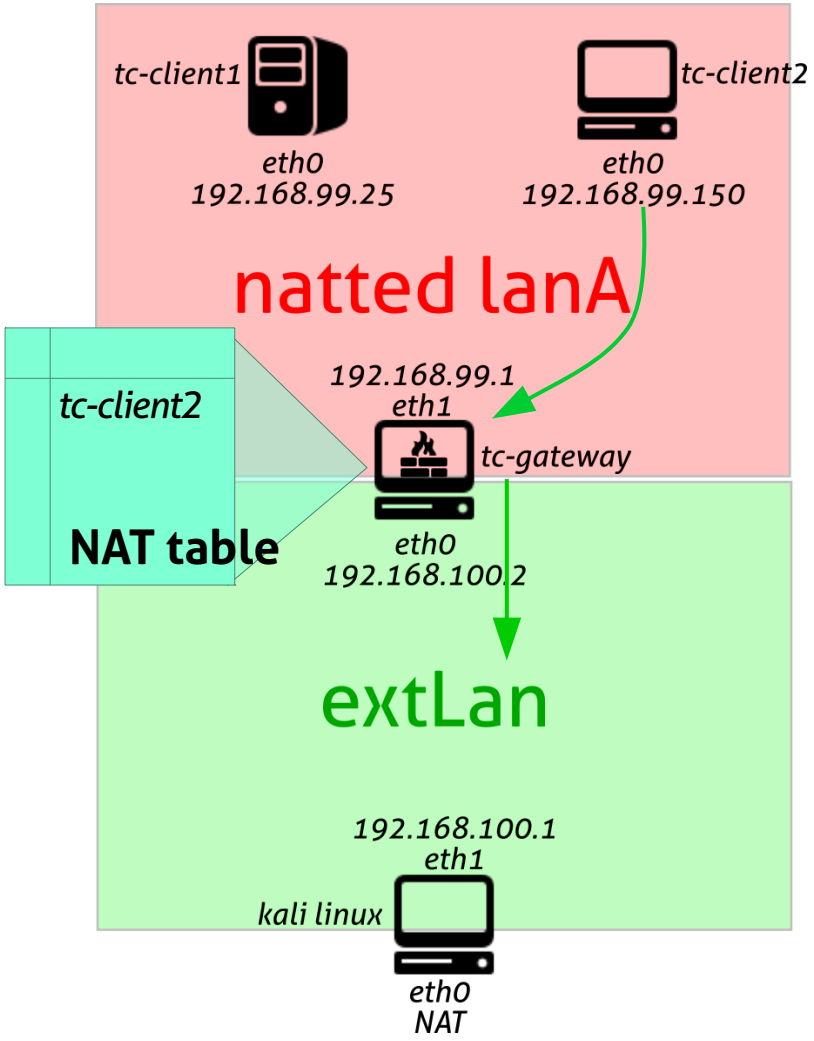
- Translate **outgoing** IP source packet headers from the internal host addresses to the WAN IP address
- The session is **masqueraded** as coming from the NATting device

source: **A**
destination: **S**
payload: [...]



# Source NAT (SNAT)

- The firewall/router:
  - Forwards replies from the client service request by the external server to the client
  - Enables the client-server session or connection to continue on another port as requested by the external server, forwarding any responses by the server to the client (the RELATED connections)
- The NAT table is where associations between requests and internal IP addresses are kept



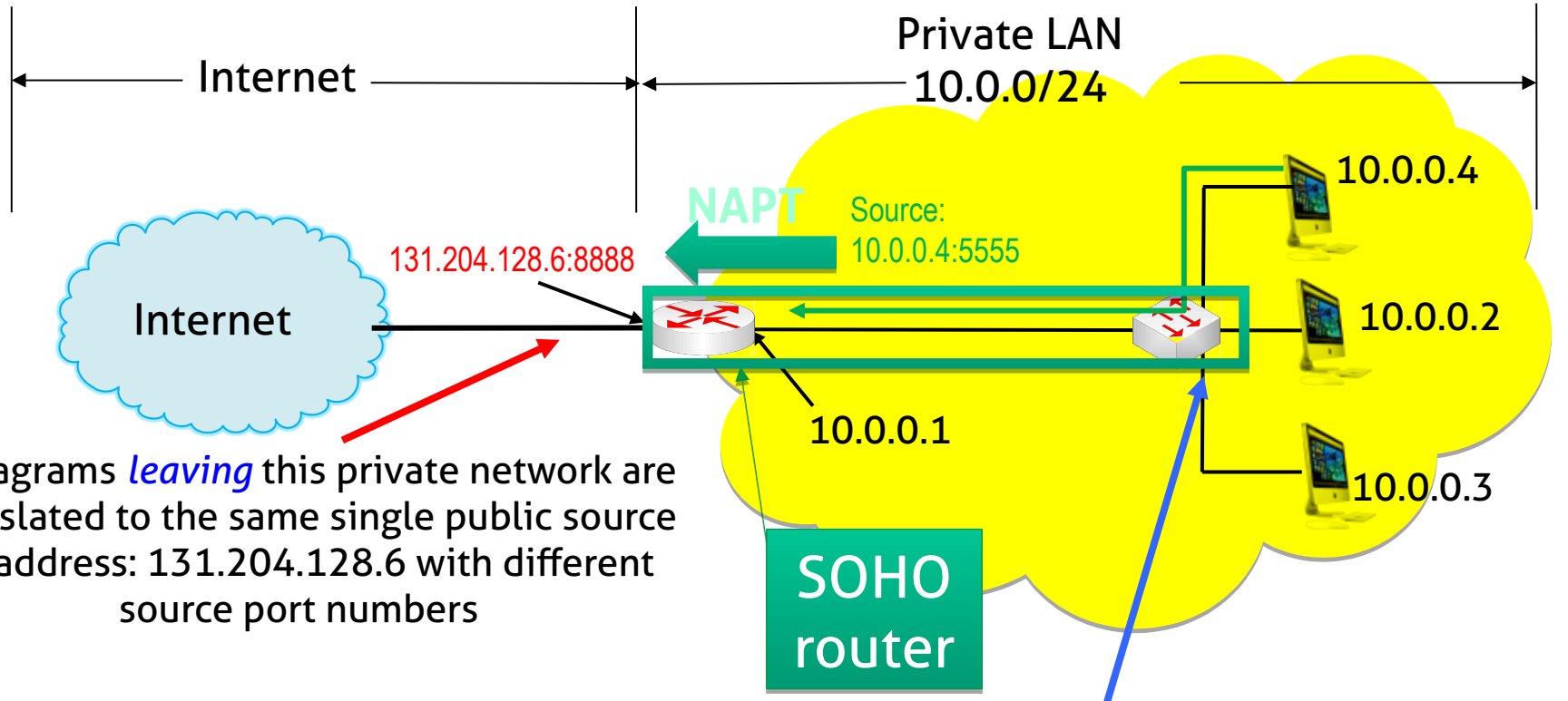
# Types of NAT: Basic and NAPT

- Basic NAT: a block of external/public IP addresses are set aside for translating the addresses of hosts within a private domain as they originate sessions to the external domain
  - For packets outbound from the private network, the source IP address and related fields such as IP, TCP, UDP and ICMP header checksums are translated
  - For inbound packets, the destination IP address and the checksums as listed above are translated
- However, multiple external/public IP addresses are difficult to obtain due to the shortage of IPv4 addresses
- Network Address Port Translation (NAPT)
- The NAPT also translates transport identifiers, e.g., TCP and UDP port numbers as well as ICMP query identifiers

# Network Address Port Translation

- Multiplex a number of private hosts into a single external/public IP address
- The IP address binding extends to transport level identifiers such as TCP/UDP ports
  - For most of the small office and home (SOHO) routers
  - The private network usually relies on a single IP address, supplied by the ISP to connect to the Internet
  - SOHO can change ISPs without changing the private IP addresses of the devices within the network
- The terms NAT and NAPT are used interchangeably in the literature
  - However the RFCs, such as RFC 3022, use the term NAPT when port numbers are involved in translation
  - Cisco refers to NAPT as PAT, i.e. Port Address Translation

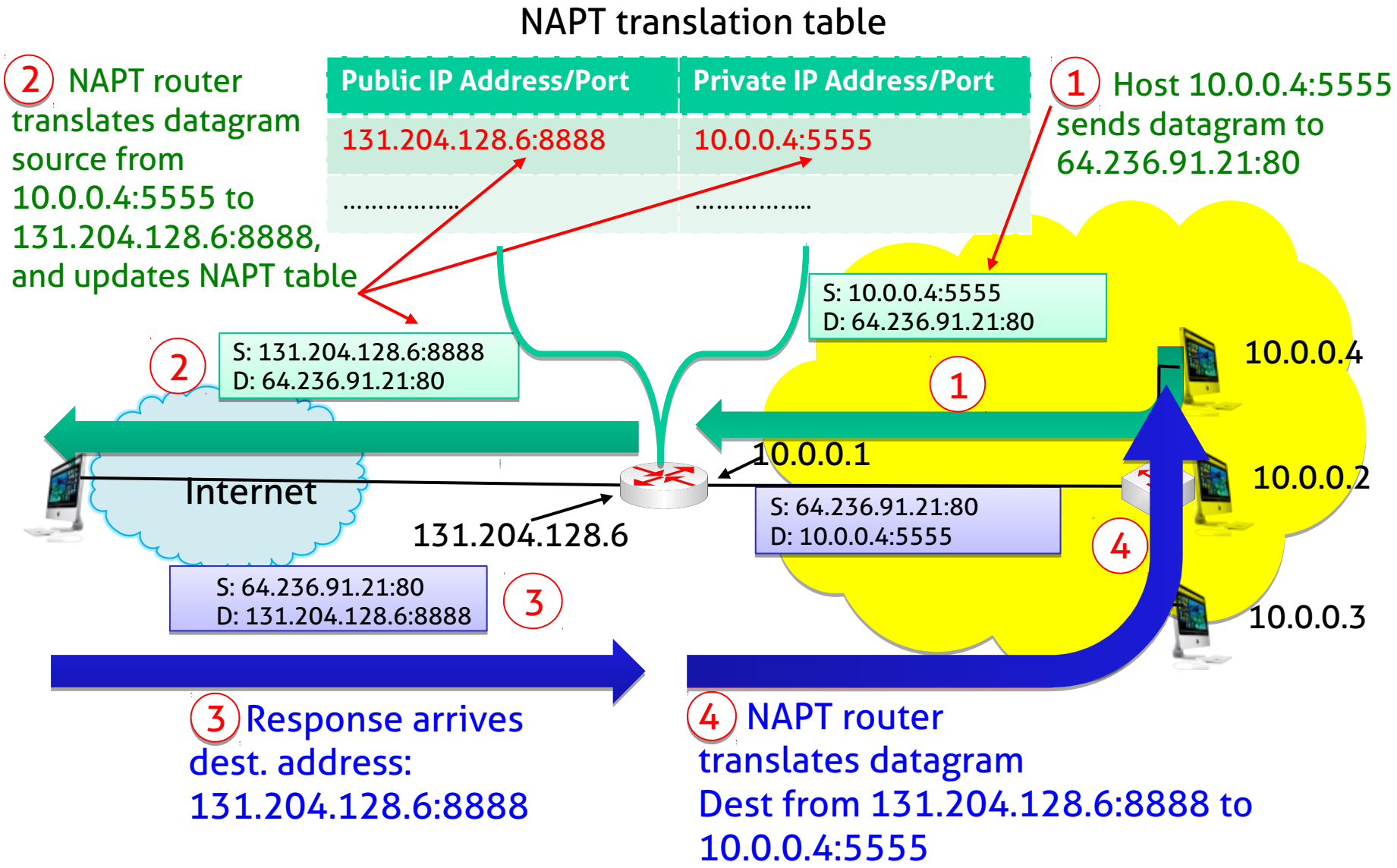
# NAPT: leaving packets




Datagrams *leaving* this private network are translated to the same single public source IP address: 131.204.128.6 with different source port numbers

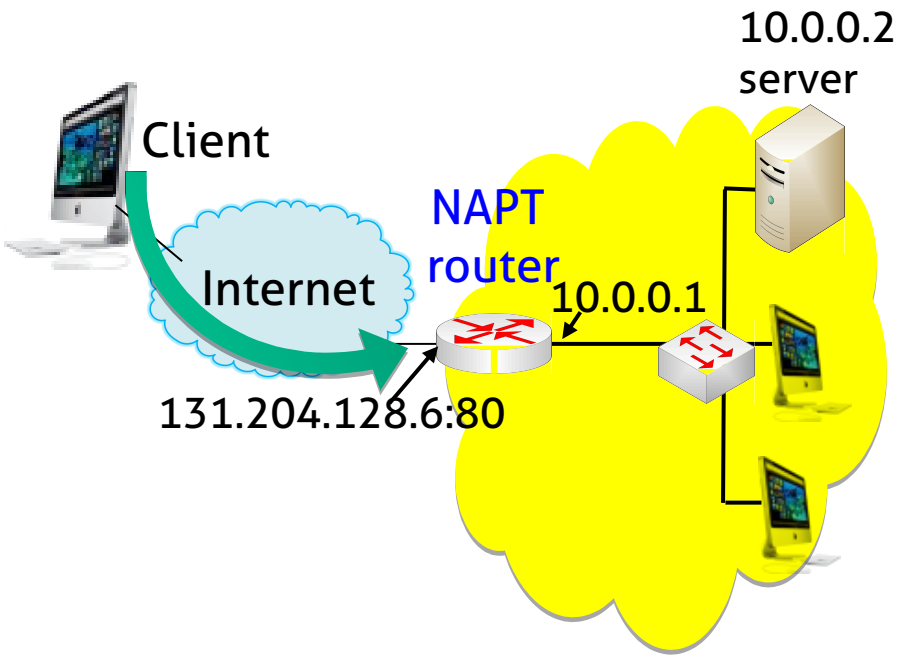
No NAPT for datagrams within this private network that have 10.0.0.0/24 IP address as a destination

# NAPT: responses



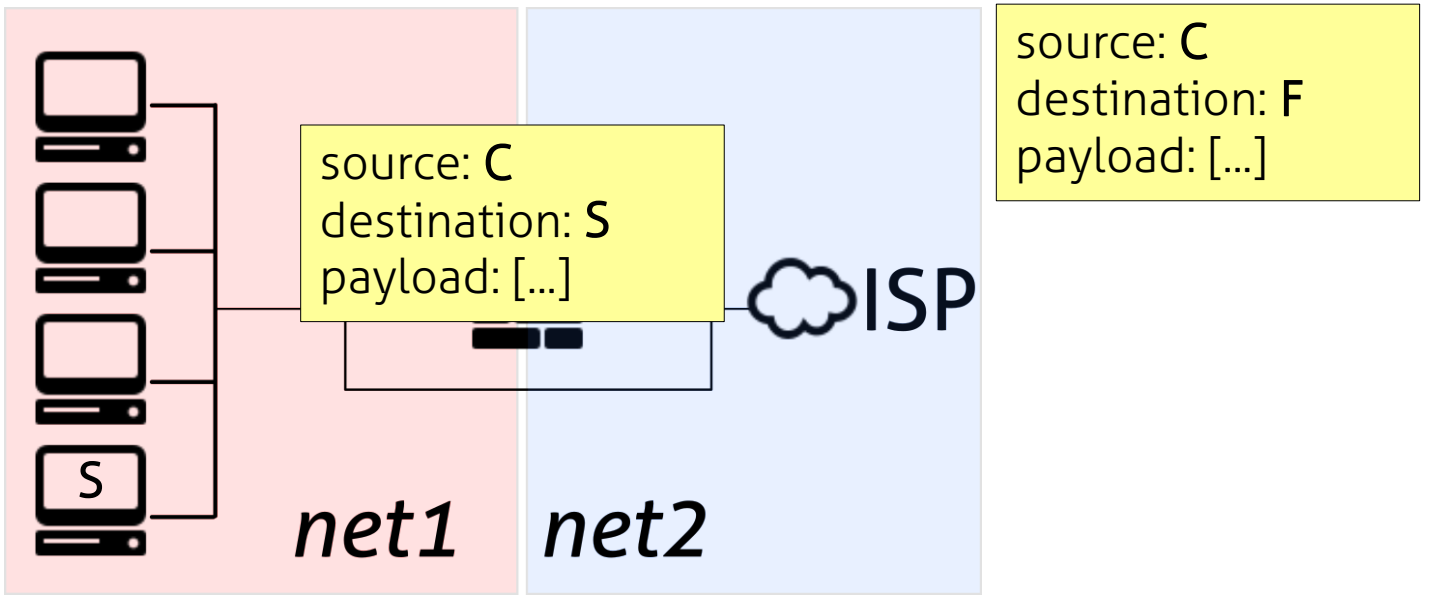
# NAPT for Incoming Requests

- NAPT router blocks all incoming ports by default
- Many applications have had problems with NAPT in the past in their handling of incoming requests
- Four major methods
  - Application Level Gateways (ALGs)
  - Static port forwarding
  - Universal Plug and Play (UPnP) Internet Gateway Device (IGD) protocol
  - Traversal Using Relays around NAT (TURN)



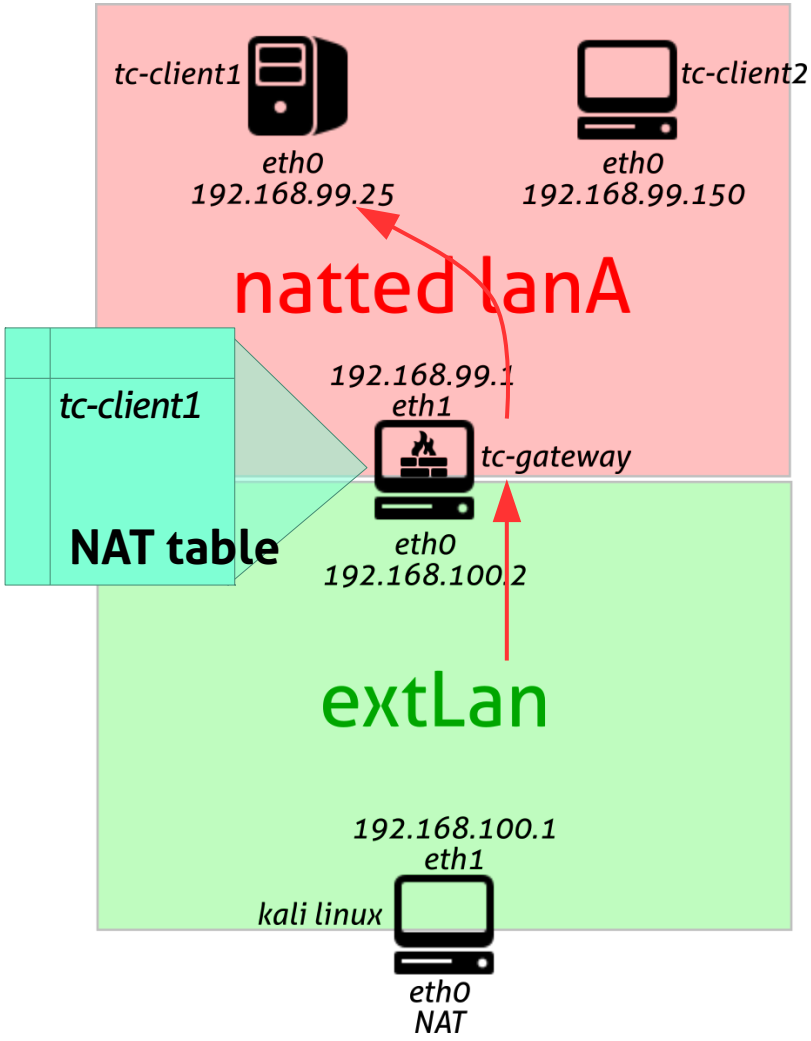
# Destination NAT (DNAT)

- Enables servers located **inside** the firewall/router LAN to be accessed by clients located outside.
- The service appears to be hosted by the firewall/router



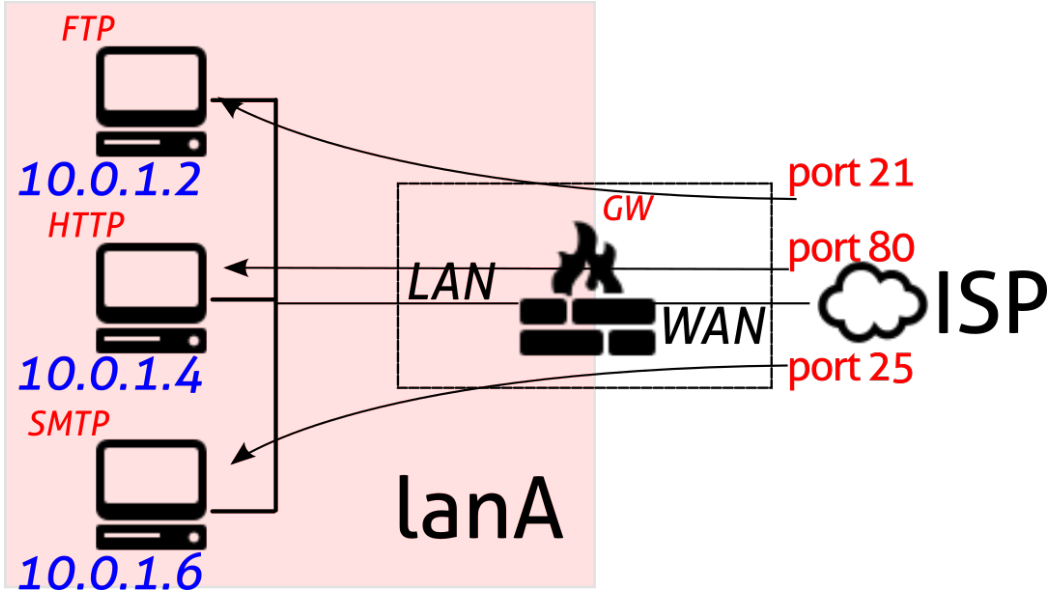
# Destination NAT (DNAT)

- The firewall/router uses the NAT table to:
  - Translate incoming packets from the firewall/router WAN IP address to the internal address of the server
  - Forward the replies to the client service request from the internal server to the external client
  - Enables the client-server session or connection to continue on another port as requested by the internal server, forwarding any responses by the client to the server (the RELATED connections)



# More on DNAT

- It is also called port forwarding or Virtual Server
- According to the port accessed from the external interface, the packets can be forwarded towards different internal hosts.
- Ex:
  - SMTP (port 25) to host A
  - HTTP (port 80) to host B
  - FTP (port 21) to host C



# NAT port forwarding

Change port numbers on DNAT sessions, to enable an internal server  to provide a particular service

- Can make use of different or differently-configured server programs to respond to internal LAN requests and to external WAN requests
- Example:
  - a host might be configured to provide outgoing SMTP service for the LAN on port 25 and incoming SMTP service on port 2525
  - the firewall will translate the port numbering for DNAT'ed incoming SMTP requests from 25 to 2525 and will also translate outgoing responses on this port intelligently


# NAT pros and cons

# NAT one-size-fits-all

- Artifacts of NAT devices values:
  - The need to establish state before anything gets through from outside to inside solves one set of problems
  - The expiration of state to stop receiving any packets when finished with a flow solves a set of problems
  - The ability for nodes to appear to be attached at the edge of the network solves a set of problems.
  - The ability to have addresses that are not publicly routed solves yet another set (mostly changes where the state is and scale requirements for the first one).

# Perceived benefits of NAT and impact on IPv4 (RFC 4864)


- Simple Gateway between Internet and Private Network
- Simple Security Due to Stateful Filter Implementation
- User/Application Tracking
- Privacy and Topology Hiding
- Independent Control of Addressing in a Private Network
- Global Address Pool Conservation
- Multihoming and Renumbering with NAT

# RFC 4864, IPv4 vs IPv6

| Goal | IPv4 | IPv6 |
|---|---|---|
| Simple Gateway as default router and address pool manager | DHCP - single address upstream DHCP - limited number of individual devices downstream | DHCP-PD - arbitrary length customer prefix upstream SLAAC via RA downstream |
| Simple Security | Filtering side effect due to lack of translation state | Explicit Context Based Access Control (Reflexive ACL) |
| Local Usage Tracking | NAT state table | Address uniqueness |
| End-System Privacy | NAT transforms device ID bits in the address | Temporary use privacy addresses |
| Topology Hiding | NAT transforms subnet bits in the address | Untraceable addresses using IGP host routes /or MIPv6 tunnels |
| Addressing Autonomy | RFC 1918 | RFC 3177 & 4193 |
| Global Address Pool Conservation | RFC 1918 << 2^48 application end points topology restricted | 17*10^18 subnets 3.4*10^38 addresses full port list / addr unrestricted topology |
| Renumbering and Multihoming | Address translation at border | Preferred lifetime per prefix & multiple addresses per interface |

# Applications not working with NAT (RFC 3027)

- Applications that have realm-specific (public or private) IP address information in payload
- Bundled session applications
- Peer-to-peer applications
- IP fragmentation with NAPT enroute
- Applications requiring retention of address mapping
- Applications requiring more public addresses than available
- Encrypted protocols like IPsec, IKE, Kerberos

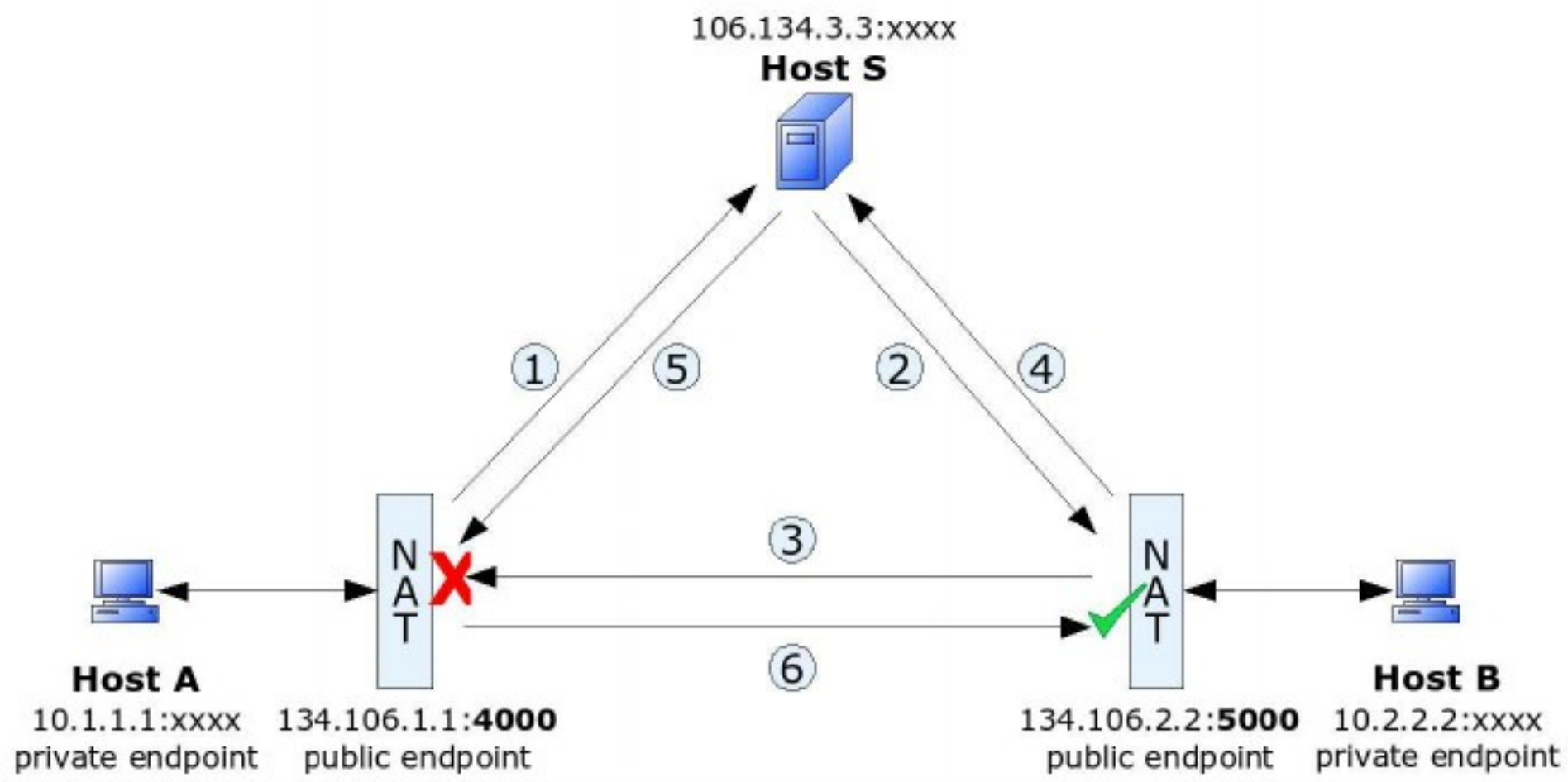
# Possible mitigation

- Trivial: write applications that do not have IP info in the payload
- Use NAT in conjunction with Application Layer Gateways
  - Gateways specific for making NAT working with a given application
    - Ex: FTP, SIP, SMTP
  - Look for specific application traffic and perform IP/port translation also in the payload
- Use Interactive Connectivity Establishment (ICE)
  - Using STUN servers and TURN servers
- Unlikely to find a solution for encrypted application traffic

# Hole punching

- A method for establishing bidirectional connections between Internet hosts, both in private networks using NAT
  - Working depends on the NAT implementation details
- Main idea:
  - find the public IP address of the other peer and initiate a connection to create a NAT state, so that replies can be correctly passed
- STUN: Session Traversal Utilities for NAT is used for this purpose, discovery and help two peers to communicate

# Example of third party for NAT traversal



# Example: UDP hole punching

1)Host A submits the request for communication with Host B to Host S

2)Host S submits the public endpoint (port 4000) of Host A to Host B

3)Host B sends a packet via it's public-private endpoint (port 5000) to the public endpoint of Host A (4000).

1) The NAT system of Host A rejects the packet, but a NAT session on B's side is now established ("a hole is punched"), ready to translate any packets from A's public endpoint (port 4000) to B's public endpoint (port 5000) into B's corresponding private endpoint

4)Host B notifies Host S and awaits an incoming connection from Host A

5)Host S submits the public endpoint of Host B (port 5000) to Host A

6)Host A uses its public-private endpoint (port 4000) to establish a connection with Host B's public endpoint (port 5000)

This technique appears to work with 82 percent UDP and 64 percent TCP NAT systems

B. Ford. *Peer-to-peer communication across network address translators,2005. USENIX Annual Technical Conference*


# iptables
# Tables and chains

# Iptables fundamantals

- The rules are grouped in “tables”
- Each table has different “chains” of rules and different possible “targets” (packet fates)
- Each packet is subject to each rule of a chain
- Packet fates depend on the **first matching rule**
- To see chains and rules of the filter table

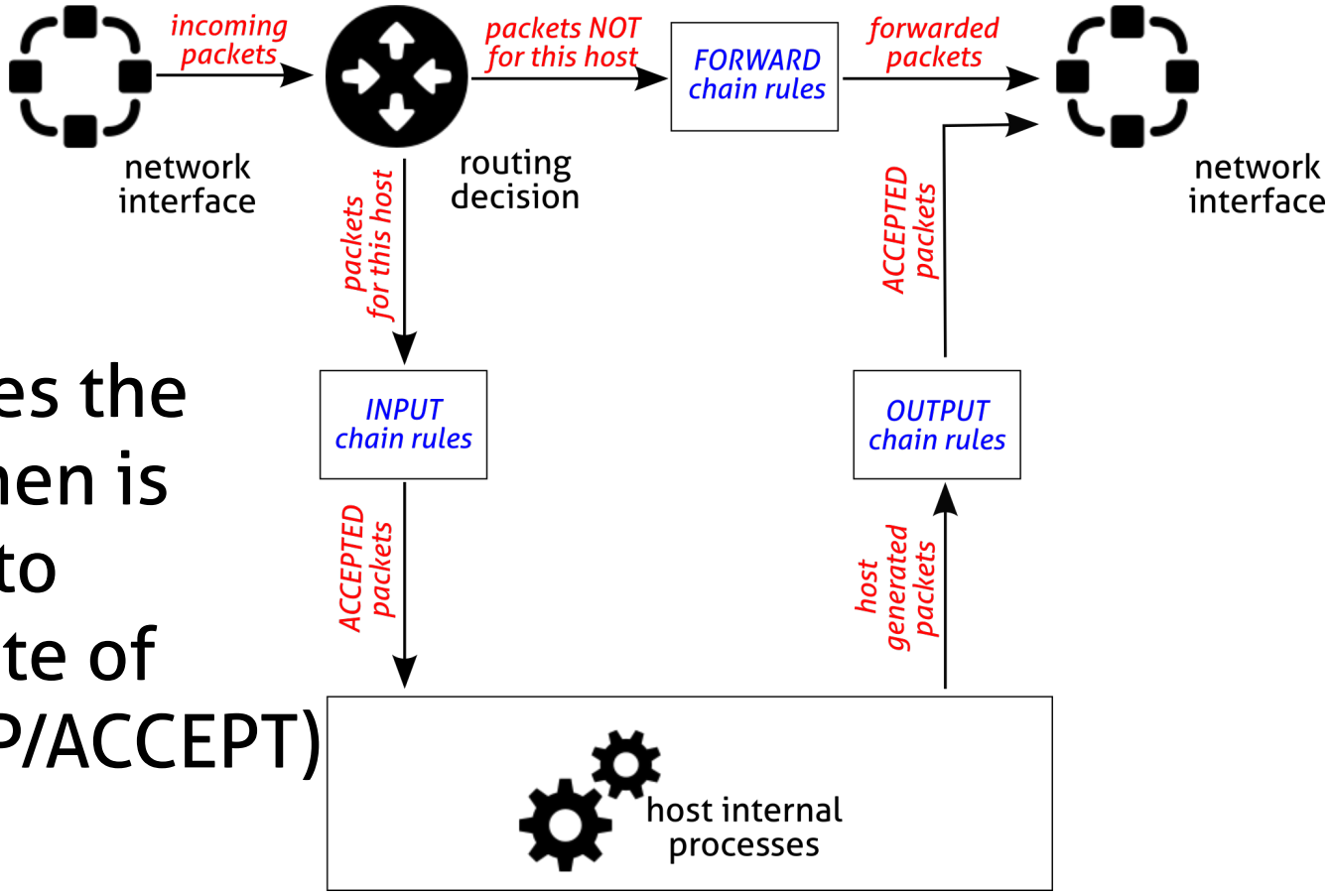
```
iptables -L
```

or for extended output

```
iptables -L -n -v --line-numbers
```

# Filter table

- Three built-in rule chains:
  - INPUT
  - OUTPUT
  - FORWARD
- If a packet reaches the end of a chain, then is the chain policy to determine the fate of the packet (DROP/ACCEPT)



# Useful iptables command switches

| iptables switches | Description |
|---|---|
| -t table | Specifies the table (filter if absent) |
| -j target | Jump to the target (it can be another chain) |
| -A chain | Append a rule to the specified chain |
| -F | Flush a chain |
| -P policy | Change the default policy |
| -p protocol | Match the protocol type |
| -s ip-address | Match the source IP address |
| -d ip-address | Match the destination IP address |
| -p tcp --sport port | Match the tcp source port (also works for udp) |
| -p tcp --dport port | Match the tcp destination port (also works for udp) |
| -i interface-name | Match input interface (from which the packet enters) |
| -o interface-name | Match output interface (on which the packet exits) |

# Other useful iptables command switches

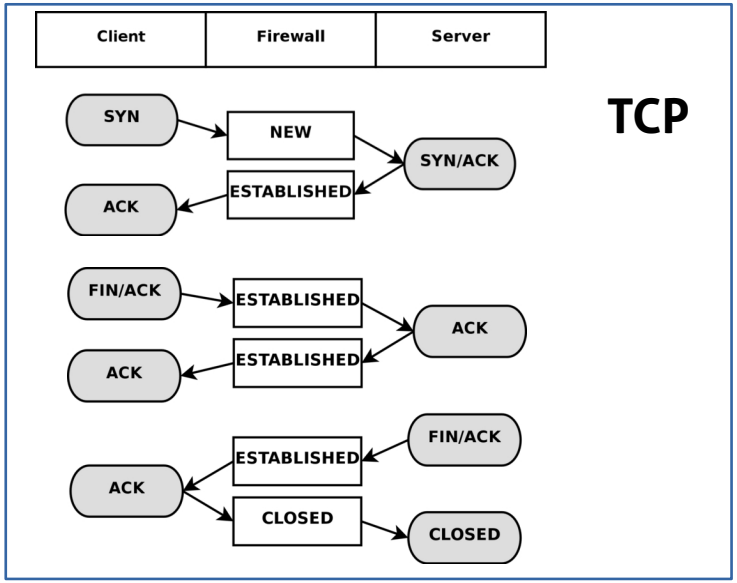
| iptables switches | Description |
|---|---|
| -p tcp --sport port | Match the tcp source port |
| -p tcp --dport port | Match the tcp destination port |
| -p udp --sport port | Match the udp source port |
| -p udp --dport port | Match the udp destination port |
| --icmp-type type | Match specific icmp packet types |
| -m *module* | Uses an extension module |
| -m state --state s | Enable connection tracking.<br>Match a packet which is in a specific state:<br>NEW: the packet is the start of a new connection<br>ESTABLISHED: the packet is part of an established connection<br>RELATED: the packet is the starting of a related connection (like FTP data)<br>INVALID: the packet could not be identified |
| -m multiport ... | Enable specification of several ports with one single rule |

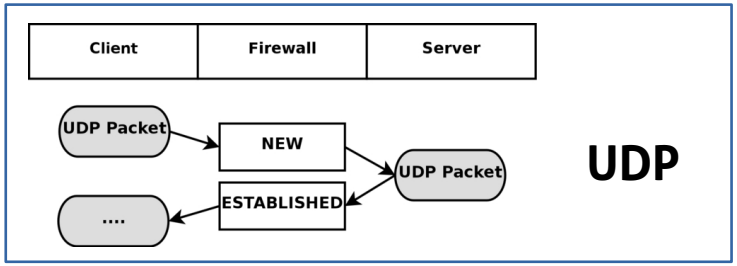
# Even more useful iptables command switches

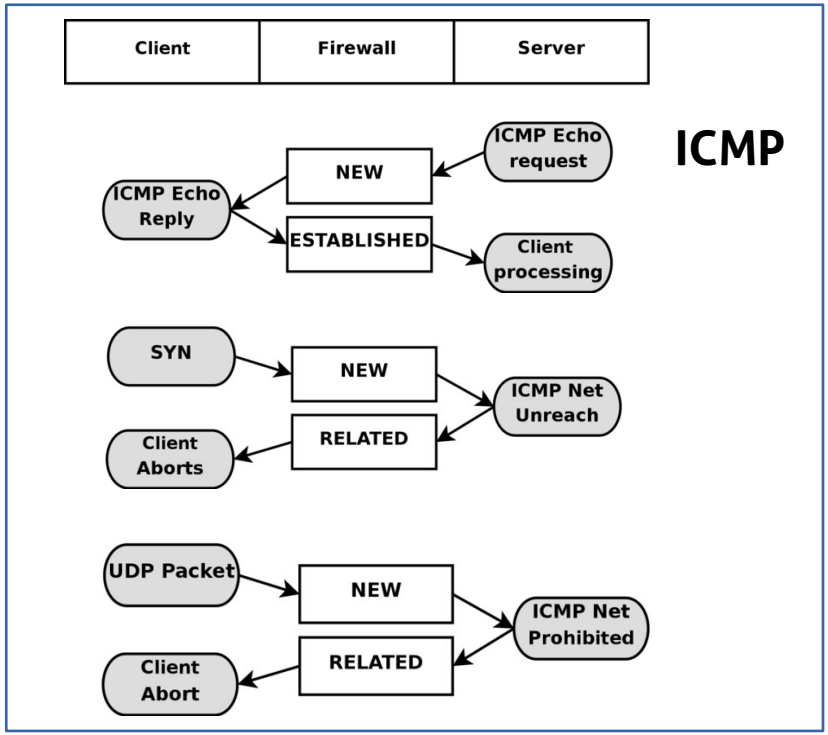
| iptables switches | Description |
|---|---|
| -f | Match fragments which are not the first of a packet |
| --syn | Match packets which start a TCP connection<br>It is equivalent to --tcp-flags SYN,RST,ACK SYN |
| -n | Used to avoid name substitution for IP hosts and ports |
| ! (==*not*) | Used to negate the match. Can be used with many flags.<br>Example: all the source addresses but a specific one<br>**-s ! ip_address** |
| -m conntrack -ctstate... | Enable connection tracking (superset of state) |
| -m mac --mac-source | Match packets with a specific source MAC address |
| -m limit | Limit the number of packets for a specific time period<br>--limit and --limit-burst<br>**iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT** |

# More on the conntrack module

- Clever use of logic to recognize connections, even with connection-less protocols (UDP, ICMP...)




More on this:

https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html#STATEMACHINE

# iptables: four built-in tables

1. MANGLE: manipulate bits in TCP header
2. FILTER: packet filtering
3. NAT: network adress translation
4. RAW: exceptions to connection tracking
   - When present RAW table has the highest priority
   - Used only for specific reasons
   - Default: not loaded

# MANGLE table

- Used for IP header manipulation (like ToS, TTL,...)
  - Should not be used for FILTERING
  - Should not be used for NAT
- Five chains to alter:
  - PREROUTING: incoming packets before a routing decision
  - INPUT: packets coming into the machine itself
  - OUTPUT: locally generated outgoing packets
  - FORWARD: packets being routed through the machine
  - POSTROUTING: packets immediately after the routing decision

# FILTER table

- Used for filtering packets
- Three built-in rule chains:
  - INPUT: incoming packets intended for the filtering machine
  - OUTPUT: outgoing packets
  - FORWARD: for the packets that are not intended for this machine
    - The FORWARD chain only used when the machine is configured as a router (net.ipv4.ip_forward to 1 in the /etc/sysctl.conf file)

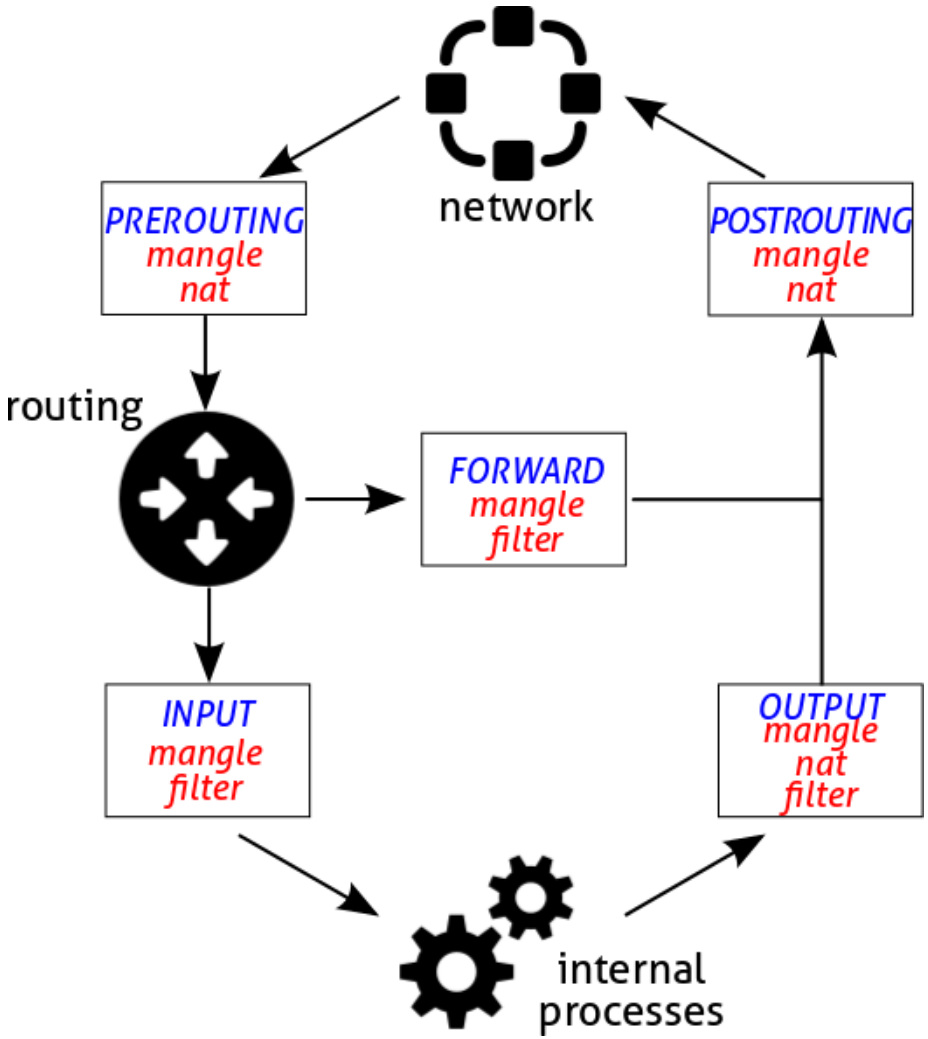
# NAT table

- Used for NAT (Network Address Translation): to translate the packet's source field or destination field
  - Only the first packet in a stream will hit this table (the rest of the packets will automatically have the same action)
- Special targets (*packet fates/actions*):
  - DNAT: destination nat
  - SNAT: source nat
  - MASQUERADE: dynamic nat (when fw interface address is dynamically assigned)
  - REDIRECT: redirects the packet to the machine itself

# NAT'ing targets

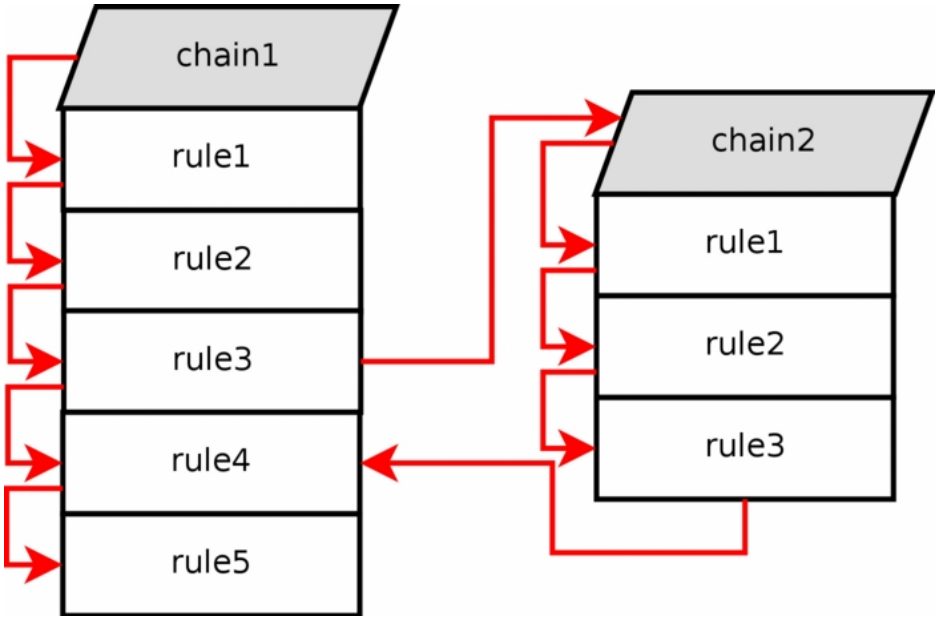
- DNAT: Destination address translation
  - Transform the destination IP of incoming packets
  - Used in PREROUTING chain
- SNAT: Source address translation
  - Transform the source IP of outgoing packets
    - Can be done one-to-one or many-to-one
  - Used in POSTROUTING chain
- MASQUERADE: like SNAT but the source IP is taken form the dynamically assigned address of the interface

# Chain and table priorities




- MANGLE>NAT>FILTER
- RAW>MANGLE
  - Not shown in the picture
  - Only used during PREROUTING and OUTPUT

# User defined chains




- It is possible to specify a jump rule to a different chain within the same table
- The new chain must be user specified
- If the end of the user specified chain is reached, the packet is sent back to the invoking chain

# iptables logging

- LOG as possible target
  - "non-terminating target", i.e. rule traversal continues at the next rule
  - to log dropped packets, use the same DROP rule, but with LOG target
- When this option is set for a rule, the Linux kernel will print some information on all matching packets (like most IP header fields) via the kernel log (where it can be read with dmesg or syslogd(8))

*--log-level level*: specifies the type of log (emerg, alert, crit, err, warning, notice, info, debug)

*--log-prefix prefix:* add further information to the front of all messages produced by the logging action

# Log example

- Log fowarded packets
  - `iptables -A FORWARD -p tcp -j LOG \ --log-level info --log-prefix "Forward INFO"`
- Log and drop invalid packets
  - `iptables -A INPUT -m conntrack --ctstate \ INVALID -j LOG --log-prefix "Invalid packet"`
  - `iptables -A INPUT -m conntrack --ctstate \ INVALID -j DROP`

# That's all for today

- **Questions?**
- See you tomorrow
- Resources:
  - "Building internet firewalls", Elizabeth D. Zwicky, Simon Cooper, D. Brent Chapman, O'Reilly 2nd ed.
    - https://docstore.mik.ua/orelly/networking_2ndEd/fire/index.htm
  - "Firewalls and Internet security: repelling the wily hacker", William R. Cheswick, Steven M. Bellovin, Aviel D. Rubin, Addison-Wesley 2nd ed.
  - www.frozentux.net/iptables-tutorial/iptables-tutorial.html
  - Local Network Protection for IPv6 (https://tools.ietf.org/html/rfc4864)