Firewall: iptables

CSC 154

Partial slides adapted from Breno de Medeiros, Blair Hicks, Trent Jaeger and iptables man page.

Firewall Options

- Commercial Firewall Devices (Cisco PIX)
- Routers (ACL Lists)
- Linux
- …

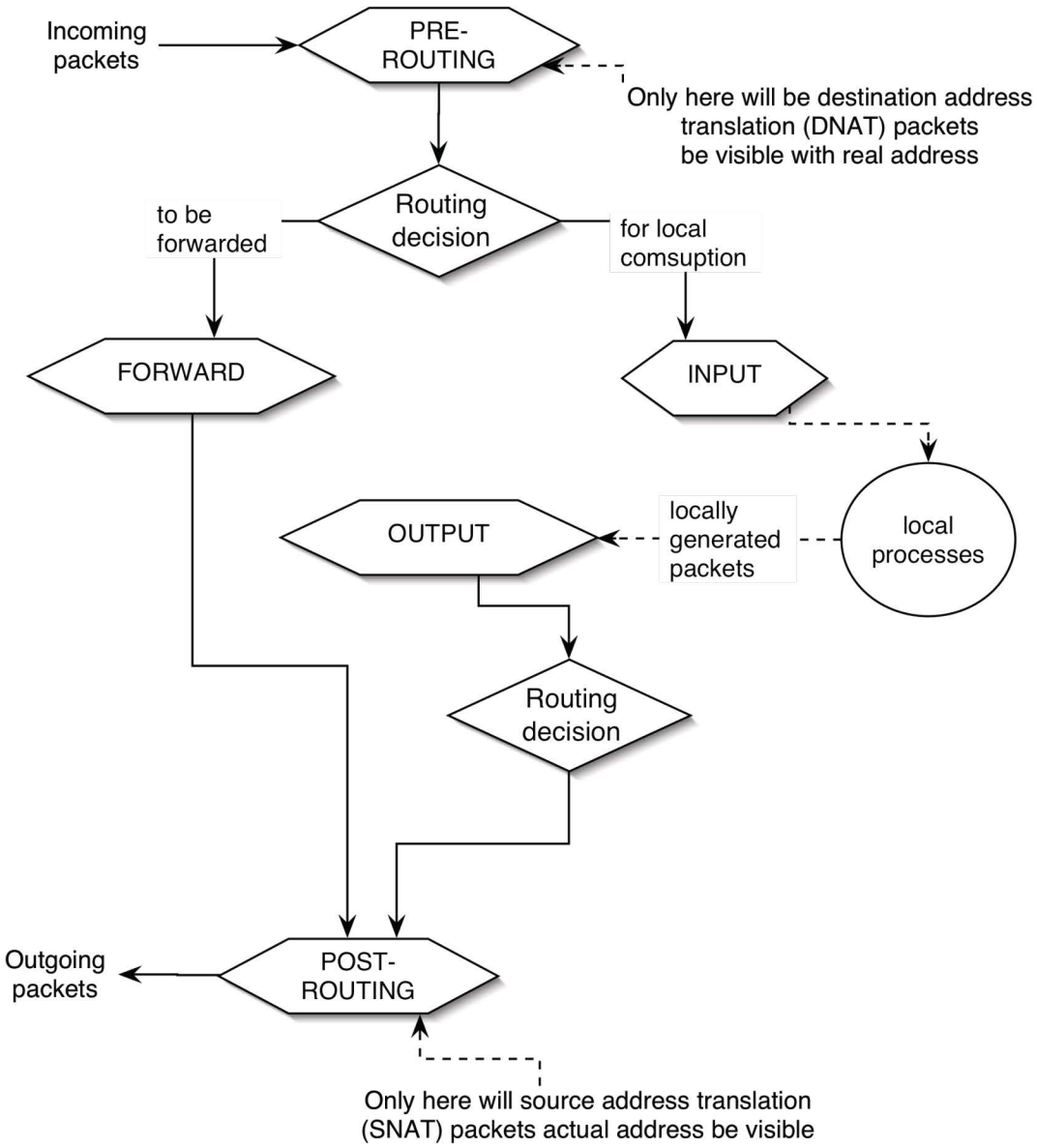
Commands to Configure Cisco Router

- cu -l /dev/ttyS0 -s 9600 //connect to cisco devices via serial port
- Router>enable //User mode->privilege mode
- Router#configure terminal //Privilege mode->global configuration mode
- Router(config)#hostname DMZ
- DMZ(config)#interface vlan 1 //Enter interface mode
- DMZ(config-if)# ip address 192.168.202.1 255.255.255.0
- DMZ(config-if)# no shutdown
- DMZ(config)#router rip //enter router configuration mode
- DMZ(config)#version 2
- DMZ(config)#network 192.168.101.0 //associate with the RIP routing process
- DMZ(config)#network 192.168.202.0
- DMZ(config)#ip route 0.0.0.0 0.0.0.0 192.168.101.14 //static routing
- DMZ(config)#access-list 101 permit tcp any host 192.168.101.5 eq 80
- DMZ(config)#inter ethernet 0
- DMZ(config-if)#ip access-group in 101 in //apply the access lists to Ethernet interfaces
- DMZ(config)#end
- DMZ# copy running-config startup-config // save the current running configuration to the //startup configuration file

Linux

- ipfwadm: Linux kernel 2.0.34
- ipchains: Linux kernel 2.2.*
- **iptables**: Linux kernel 2.4.* - later versions
  - Packet traversal in Linux

IPtables


Incoming packets
PRE-ROUTING
Only here will be destination address translation (DNAT) packets be visible with real address
Routing decision
to be forwarded
for local comsuption
FORWARD
INPUT
OUTPUT
locally generated packets
local processes
Routing decision
Outgoing packets
POST-ROUTING
Only here will source address translation (SNAT) packets actual address be visible

Netfilter Hooks

- **hook**s in Linux network protocol stack
  - PREROUTING: before routing
  - INPUT: inbound, for local consumption
  - FORWARD: inbound, to be forwarded
  - OUTPUT: outbound
  - POSTROUTING: after routing

iptables Concepts

- The iptables firewall looks in the firewall ***table*** to seek if the ***chain*** associated with the current hook ***matches*** a packet, and executes the ***target*** if it does.
  - Table: all the filewall rules
  - Chain: list of rules associated with the chain identifer, e.g. the hook name
  - Match: when a packet match all a rule's fields
  - Target: operation to execute when a packet matched

Syntax of IPtables commands

- iptables -A/D (INPUT/ OUTPUT/ FORWARD/ PREROUTING/ POSTROUTING) -s (source address) -p (protocol) - d (destination) (DROP/REJECT/LOG/ACCEPT/ User-defined chain)

Chain manipulation rules

- Create a new chain (-N)
- Delete an empty chain (-X)
- Change the policy for a built-in chain. (-P)
- List the rules in a chain (-L)
- Flush the rules out of a chain (-F)
- Zero the packet and byte counters on all rules in a chain (-Z)

Example: `# iptables -L -Z FORWARD`

Rule manipulation within a chain

- Append a new rule to a chain (-A)
- Insert a new rule at some position in a chain (-I)
- Replace a rule at some position in a chain (-R)
- Delete a rule at some position in a chain, or the first that matches (-D)

Examples:

- `iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP`
- `iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP`

Targets

- Define what to do with the packet at this time
  - Built-in targets
    - ACCEPT/DROP: allow/disallow the packet through
  - Extension target (come back later after extension introduced)
    - LOG: log any packet that matches
    - REJECT: drops and returns error packet
    - RETURN: enables packet to return to previous chain
    - QUEUE: pass the packet to the userspace
      - modprobe iptable_filter
      - modprobe ip_queue
      - iptables -A OUTPUT -p icmp -j QUEUE
  - User-defined chains (come back later after extension introduced)

How to specify

- Specifying Source and Destination IP Addresses
  - -s/-d
- Specifying Inversion
  - -s **!** localhost
- Specifying a protocol
  - -p TCP/-p ! TCP
- Specifying an Interface
  - -i/--in-interface
  - -o/--out-interface
- Specifying fragments
  - -f/--fragment

iptables Examples (1)

- iptables -L -v -n

- iptables -A INPUT -s 200.200.200.2 -j ACCEPT
- iptables -A INPUT -s 200.200.200.1 -j DROP
- iptables -A INPUT -s 200.200.200.1 -p tcp -j DROP
- iptables -A INPUT -s 200.200.200.1 -p tcp --dport telnet -j DROP
- iptables -A INPUT -p tcp --dport telnet -i ppp0 -j DROP
- iptables -A OUTPUT -f -d 192.168.1.1 -j DROP

iptables Examples (2)

- iptables -I INPUT -i eth1 -p tcp -s 192.168.56.1 --sport 1024:65535 -d 192.168.56.2 --dport 22 -j ACCEPT

- iptables -I OUTPUT -o eth1 -p tcp ! --syn -s 192.168.56.2 --sport 22 -d 192.168.56.1 --dport 1024:65535 -j ACCEPT

Test out (1)

- PING on localhost
  - *ping -c 1 127.0.0.1*
- Add iptables rule to block
  - *iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP*
- Try ping again
- Delete the rule
  - *iptables -D INPUT 1*
  - *iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP*
  - *iptables -F INPUT*

Test out (2)

- TCP via localhost loopback
  - *nc –l –p 3456 (server: listening to port 3456)*
  - *nc –p 2345 localhost 3456 (client: sent messages from port 2345 to 3456)*
- Add iptables rule to block
  - *iptables –A INPUT –s 127.0.0.1 –p tcp –j DROP*
- Try above TCP communication again
- Delete the rule
  - *iptables -D INPUT 1*
  - *iptables -D INPUT -s 127.0.0.1 -p tcp -j DROP*
  - *iptables -F INPUT*

iptables Extension

- Kernel Extension through kernel modules
  - /lib/modules/2.4.0-test10/kernel/net/ipv4/netfilter
  - CONFIG_KMOD configured in kernel compilation (/boot)

- Extension to the iptables program - new matches
  - /usr/local/lib/iptables/, /lib/iptables, /usr/lib/iptables
  - TCP/UDP/ICMP *automatically* offer new tests
    - -p TCP/UDP/ICMP
  - Use -m to invoke extension
    - -m mac (MAC address)
    - -m limit (rate of matches, suppressing log messages)
    - -m state (State)

    - iptables -A FORWARD -i ppp0 -m state ! --state NEW -j DROP

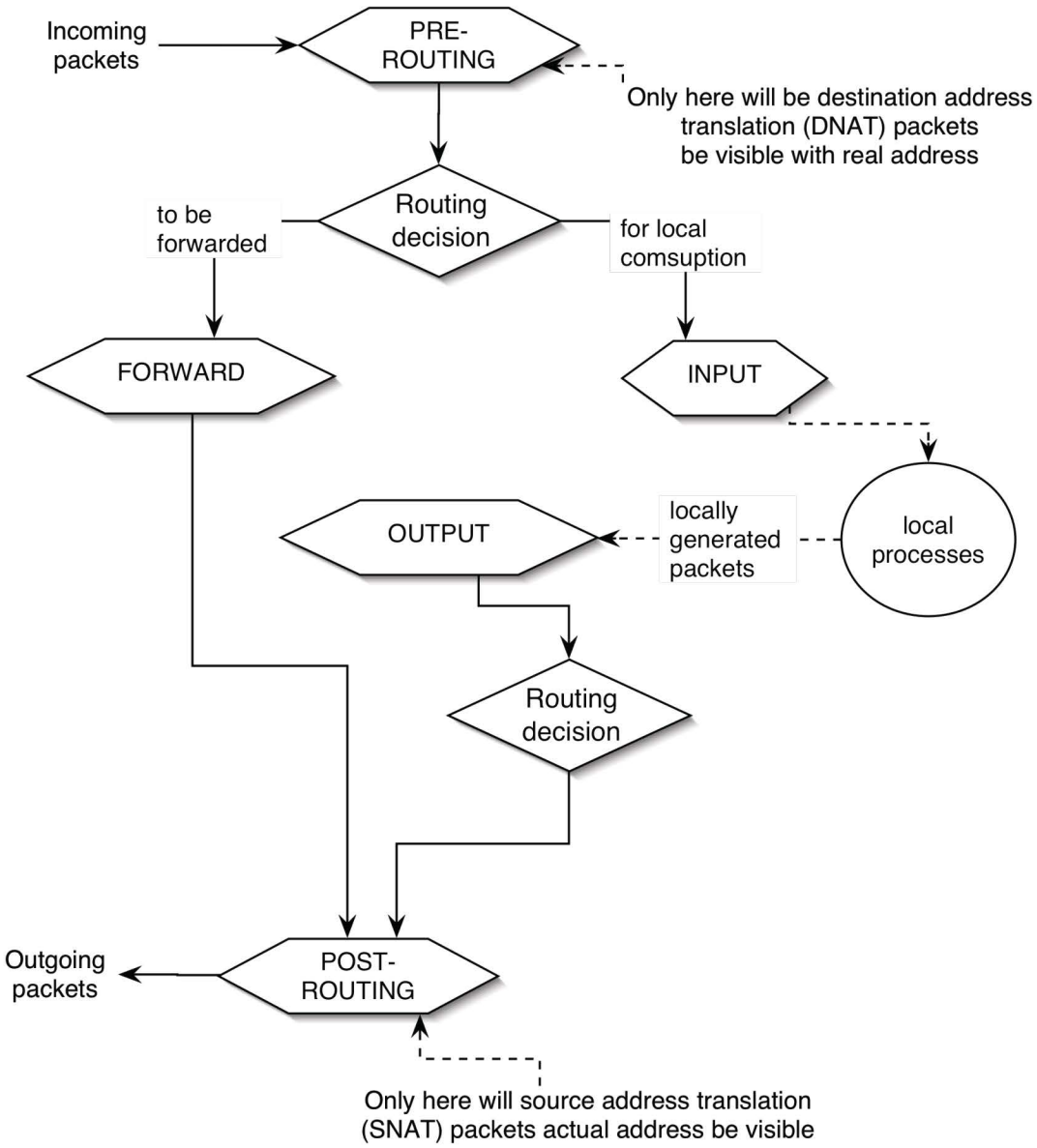
iptables – new targets

- Define what to do with the packet at this time
  - Built-in targets
    - ACCEPT/DROP: allow/disallow the packet through
  - Extension target (come back later after extension introduced)
    - LOG: log any packet that matches
    - REJECT: drops and returns error packet
    - RETURN: enables packet to return to previous chain
    - QUEUE: pass the packet to the userspace
      - modprobe iptable_filter
      - modprobe ip_queue
      - iptables -A OUTPUT -p icmp -j QUEUE
  - User-defined chains

Address Translation

- Source address translation (SNAT)
  - Used to multiplex a single IP address to provide internet connectivity to multiple boxes (clients); called “masquerading”
- Destination address translation (DNAT)
  - Allows to use several servers with a single IP address (for load balancing) or because of having a single IP for multiple servers on different ports “port forwarding”

IPtables


Incoming packets
PRE-ROUTING
Only here will be destination address translation (DNAT) packets be visible with real address
Routing decision
to be forwarded
for local comsuption
FORWARD
INPUT
OUTPUT
locally generated packets
local processes
Routing decision
Outgoing packets
POST-ROUTING
Only here will source address translation (SNAT) packets actual address be visible

NAT Overview

- Source NAT
  - The source address of the initial packet is modified.
  - Performed on the POSTROUTING Chain.
  - Includes MASQUERADE functionality.

- Destination NAT
  - The destination address of the initial packet is modified.
  - Performed on the PREROUTING or OUTPUT chain.

NAT and filtering

- The way the NAT rules are arranged with respect to regular filtering rules allows you to ignore routing when performing filtering
  - The addresses visible to the rules will be the native addresses, not the translated addresses for public consumption