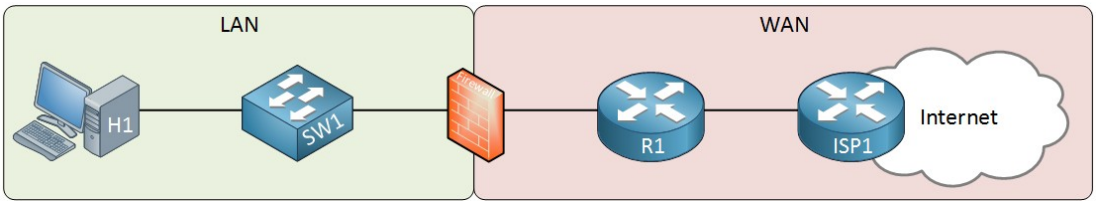
Nftables

Concepte de tallafocs:

The firewall is the barrier between a trusted and untrusted network, often used between your LAN and WAN. It's typically placed in the forwarding path so that all packets have to be checked by the firewall, where we can drop or permit them. Here's an example:



Firewalls use access-lists to check for the source and/or destination IP address and/or port numbers. Most routers however, don't spend much time at filtering…when they receive a packet, they check if it matches an entry in the access-list and if so, they permit or drop the packet. That's it. No matter if they receive a single packet or thousands, each packet is treated individually and we don't keep track of packets we have seen before or not. This is called stateless filtering.

Most firewalls, on the other hand, use stateful filtering. They keep track of all incoming and outgoing connections. For example: a web server is sitting behind a firewall, it's a busy server that accepts an average of 20 new TCP connections per second from different IP addresses. The firewall keeps track of all connections, once it sees a source IP address that is requesting more than 10 new TCP connections per second, it will drop all traffic from this source IP address, preventing a DoS (Denial of Service).

Some firewalls support some form of (deep) packet inspection, too. Simple access-lists only check source/destination addresses and ports, that's layer 3 and 4 of the OSI model. Packet inspection means we can inspect up to layer 7 of the OSI model. This means we can look at application data and even the payload. For example: instead of blocking all IP addresses that belong to lolcats.com, you can create a filter that looks for the URI in HTTP requests and block those instead so ou won't have to worry about IP addresses of web servers that might change in the future; or your firewall can check the payload to block any packets that contains known worms or viruses.

Introducció a Nftables

Netfilter (https://www.netfilter.org) és el tallafocs integrat dins el nucli Linux. Fins ara l'eina de terminal que s'havia fet servir per gestionar-lo era "Iptables" (i abans "Ipchains"), però actualment s'ha reemplaçat per "Nftables". És possible, no obstant, que a la distribució que facis servir encara no estigui instal.lat per defecte. Per tant, el primer que hauràs de fer per utilitzar Nftables serà executar, per si de cas, *sudo apt install nftables* (a Ubuntu) o *sudo dnf install nftables* (a Fedora)

> NOTA: Altres paquets importants (que s'instal.len com a dependències) són "libmnl" (proporciona les interfícies necessàries per a que es puguin comunicar el kernel i l'espai d'usuari mitjançant Netlink) i "libnftnl" (proporciona la API de baix nivell per transformar els missatges de xarxa als objectes; depén de l'anterior)

Un cop instal.lat, cal comprovar si Nftables està funcionant: *systemctl status nftables* i si no ho està, executar *sudo systemctl start nftables* . De forma alternativa, es pot confirmar que efectivament Nftables estigui funcionant observant si el mòdul del kernel "nf_tables" està carregat (ho hauria d'haver carregat la comanda anterior) escrivint en un terminal *lsmod | grep nf_tables* i observant si aquest apareix a la pantalla.

> NOTA: Altres mòduls de la família Nftables són "nf_tables_ipv6", "nf_tables_bridge", "nf_tables_arp" i "nf_tables_inet". Aquests mòduls proporcionen la corresponent taula i el suport de filtre de cadenes per cada família donada.

És molt probable que en posar en marxa Nftables es carreguin per defecte certes regles que els mantenidors de la nostra distribució Linux hagin considerat important que funcionin ja d'entrada. No obstant, per estudiar millor el funcionament de Nftables a nosaltres ens interessarà eliminar-les totes per començar "des de cero" sense interferències i control.lant així tots els detalls. Concretament:

*Per saber quines són totes les regles carregades actualment: *sudo nft list ruleset*
*Per esborrar (descarregar) totes les regles carregades actualment: *sudo nft flush ruleset*

Si es vol esbrinar d'on "agafa" Nftables les regles per defecte que els mantenidors de la nostra distribució Linux han decidit, es pot observar el valor de la línia "ExecStart" que mostra la comanda *systemctl cat nftables* Aquesta línia indica la comanda nftables concreta que s'executa en iniciar el servei i, en general, serà de la forma *nft -f /ruta/fitxer/regles*. Així doncs, podrem esbrinar que a Ubuntu el fitxer de regles per defecte utilitzat és "/etc/nftables.conf" (encara que com alternativa es podrien fer servir altres arxius de configuració d'exemple a escollir, ubicats tots a la carpeta "/usr/share/doc/nftables/examples/syntax") i a Fedora és "/etc/sysconfig/nftables.conf" (el qual apunta, al seu torn, a varis fitxers de configuració -a descomentar segons les nostres necessitats- ubicats tots a la carpeta "/etc/nftables").

Taules

Nftables està estructurat internament en forma de "taules", "cadenes" i "regles". Una "taula" és simplement un contenidor de "cadenes", les quals són al seu torn un contenidor de "regles". Aquesta estructura pretén ordenar/classificar les regles per gestionar-les d'una forma més còmoda i eficient. El primer que haurem de fer, doncs, per poder treballar amb Nftables és afegir com a mínim una taula. Un cop fet això, dins d'aquesta taula podrem afegir les "cadenes" que necessitem i dins d'aquestes "cadenes" agregarem finalment les "regles" que volguem definir.

Cada taula Nftables ha de ser d'un tipus ("familia") determinat, d'entre sis possibles:

*__ip__ : les cadenes que conté s'encarreguen de gestionar paquets IPv4
*__ip6__ : les cadenes que conté s'encarreguen de gestionar paquets IPv6
*__inet__ : les cadenes que conté s'encarreguen de gestionar paquets IPv4 i IPv6
*__arp__ : les cadenes que conté s'encarreguen de gestionar paquets ARP
*__bridge__ : les cadenes que conté s'encarreguen de gestionar paquets que travessen un "bridge"
*__netdev__ : les cadenes que conté s'encarreguen de gestionar paquets de nivell 2 que entren al sistema abans fins i tot del procés "prerouting"

Les comandes més comunes per gestionar les taules són:

sudo nft add table <família> <taula>	: crea una taula del tipus ("família") indicat
sudo nft flush table <família> <taula>	: buida tot el contingut de la taula dita (pas previ a eliminar-la)
sudo nft delete table <família> <taula>	: elimina una taula del tipus ("família") indicat
sudo nft list tables [<família>]	: mostra les taules existents (totes o les de la família indicada)
sudo nft list -nn -a table <familia> <taula>	: mostra tot el contingut (cadenes i regles) de la taula indicada El paràmetre -a serveix per mostrar al costat de cada regla el seu número identificador ("handle") -útil per quan volguem reemplaçar-la o esborrar-la -. El paràmetre -nn serveix per no resoldre noms ni d'adreces IP ni de ports

És un conveni força habitual per motius històrics (encara que no sigui obligatori de seguir) crear com a mínim una taula anomenada *"filter"* de tipus "inet" per encabir-hi les cadenes relacionades amb el filtratge de paquets d'entrada i sortida del sistema (és a dir, les cadenes de tipus "filter") i, si volem que la nostra màquina funcioni a més com a "router", una segona taula anomenada *"nat"*, també de tipus "inet", per encabir-hi les cadenes relacionades amb la realització de NAT (és a dir, les cadenes de tipus "nat").

Cadenes

Una cadena és un contenidor de regles. Cada cadena que creem ha de tenir un tipus, un "hook", una prioritat i, si la cadena és de tipus "filter", també una "policy".

1.- Els tipus d'una cadena poden ser:

*"__filte__r" : dissenyada per filtrar paquets. Pot existir dins de taules de totes les famílies menys "netdev"
*"__nat__" : dissenyada per realitzar NAT. Pot existir només dins de taules de les famílies "ip" i "ip6"
*"__route__" : dissenyada per marcar paquets. " " " " " " "

2.- Un "hook" representa un punt determinat en el camí que segueix el paquet al llarg del seu processament per part del kernel. Una cadena pot ser registrar-se en un "hook" concret per aconseguir que els paquets que apareguin en aquest "hook" travessin la cadena en qüestió (i per tant, siguin sotmesos a les regles allà

presents). Normalment en un "hook" hi ha registrada una sola cadena, però poden haver-hi més. D'altra banda, si una cadena no es registra en cap "hook", no serà travessada per cap paquet però pot ser usada per organitzar altres cadenes. Els "hooks" on "clavar" una cadena poden ser els següents:

Hooks que només poden assignar-se a una cadena de tipus "filter"

*"**input**" : representa el punt per on els paquets entren al sistema (perquè van dirigits a ell)
*"**forward**": representa el punt per on els paquets travessen el sistema (és a dir, entren i surten) perquè només estan de pas, ja que venen d'un altre sistema i van dirigits a un altre sistema diferent. Aquest hook no es pot assignar a una taula "arp"
*"**output**" : representa el punt per on els paquets surten del sistema (perquè són originats en ell)

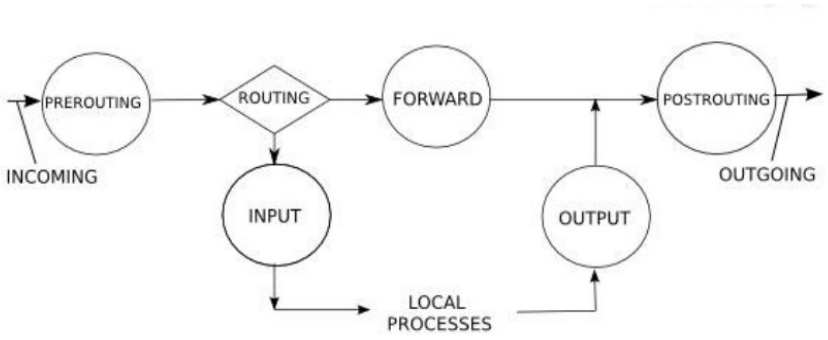
Hooks que només poden assignar-se a una cadena de tipus "nat"

*"**postrouting**" : representa el punt on els paquets, abans de sortir del sistema però un cop ja s'ha pres la decisió del seu enrutament, s'inspeccionen per últim cop. Aquest "hook" se sol fer servir per realitzar un tipus de NAT anomenat SNAT (Source NAT") en el qual es canvia l'adreça IP original d'origen dels paquets sortints (que sol ser la IP privada del node particular de la LAN interna que els ha generat) per una altra IP (que sol ser la IP pública -externa- del sistema)
*"**prerouting**" : representa el punt on els paquets, abans d'entrar al sistema, s'inspeccionen per prendre una decisió sobre la ruta que han de seguir. Aquest "hook" se sol fer servir per realitzar un tipus de NAT anomenat DNAT ("Destination NAT") en el qual es canvia l'adreça IP original de destí dels paquets entrants (que sol ser la IP pública -externa- del sistema) per una altra IP (que sol ser una IP privada d'algun node particular de la LAN interna)

Altres hooks

*"**ingress**" : hook que només pot assignar-se a una cadena pertanyent a una taula de tipus "netdev". Representa el punt on els paquets just són detectats per la tarja de xarxa (a nivell 2), abans de realitzar el prerouting . Pot servir com alternativa a l'eina *tc*

El següent esquema exposa gràficament l'ordre de processament dels "hooks" (faltaria el de tipus "ingress", que és previ a "prerouting"):



3.- Hi ha certes prioritats predefinides (llistades tot seguit) que estan assignades a determinats tipus de cadenes (i a altres processos relacionats amb la gestió de paquets al sistema). En general, la prioritat que s'associarà a una cadena sol ser simplement la predefinida corresponent al tipus de cadena que és però si s'indica una prioritat específica, aquesta cadena (i les regles que inclogui a dins, per tant) se situarà "artificialment" amb més o menys prioritat que les cadenes que tinguin prioritat predefinida. Això és important perquè en el cas de què hi hagi regles que es contradiguin entre sí, s'aplicarà sempre la que tingui una prioritat major (independentment de l'ordre en el que aquestes regles estiguin escrites). Només en el cas de regles amb la mateixa prioritat serà important l'ordre en el que aquestes regles s'hagin definit entre sí (veure més avall). Les prioritats predefinides (de menor a major) són:

NF_IP_PRI_CONNTRACK_DEFRAG (-400): priority of defragmentation
NF_IP_PRI_RAW (-300): traditional priority of the raw table placed before connection tracking operation
NF_IP_PRI_SELINUX_FIRST (-225): SELinux operations
NF_IP_PRI_CONNTRACK (-200): Connection tracking operations
NF_IP_PRI_MANGLE (-150): mangle operation

NF_IP_PRI_NAT_DST **(-100): destination NAT**
NF_IP_PRI_FILTER **(0): filtering operation,** the filter table
NF_IP_PRI_SECURITY (50): Place of security table where secmark can be set for example
NF_IP_PRI_NAT_SRC **(100): source NAT**
NF_IP_PRI_SELINUX_LAST (225): SELinux at packet exit
NF_IP_PRI_CONNTRACK_HELPER (300): connection tracking at exit

4.- Finalment, la "policy" d'una cadena (de tipus "filter") indica la política que s'aplicarà a un paquet que no concorda amb cap de les regles que es defineixin dins de la cadena en qüestió i pot valer: "**accept**","**drop**","**queue**","**continue**" o "**return**" (el seu significat és equivalent a les accions homònimes que estudiarem tot seguit).

NOTA: El més segur per controlar tràfic d'entrada és aplicar la "policy" "drop" i llavors només obrir els ports estrictament necessaris un a un amb regles específiques. Pel tràfic de sortida se sol aplicar la "policy" "accept".

Les comandes més comunes per gestionar les cadenes són les següents (per més informació es pot consultar la web https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes):

sudo nft add chain <familia><taula><cadena> { type <tipus> hook <hook> priority <n> \; policy <policy> \; }
sudo nft create chain <familia><taula><cadena> { type <tipus> hook <hook> priority <n> \; policy <policy> \; }
sudo nft flush chain <familia> <taula> <cadena> : buida la cadena de regles (pas previa a eliminar-la)
sudo nft delete chain <familia> <taula> <cadena> : elimina la cadena de regles
sudo nft list chains : mostra les cadenes carregades (sense mostrar regles)
sudo nft list chain <familia> <taula> <cadena> : mostra les regles de la cadena indicada

NOTA: La comanda *nft create ...* és igual que *nft add ...* però dóna error si la regla ja està creada

NOTA: Tal com ja s'ha comentat, es poden crear cadenes que no vegin cap paquet (en no estar unides a cap "hook") però que poden ser útils per organitzar un conjunt de regles en un arbre de cadenes utilitzant salts a l'acció de la cadena. Això es fa així amb la comanda *sudo nft add chain <taula> <cadena>*

Regles

Un cop el paquet ja està “encuat” al "hook" adient, el que Nftables farà és inspeccionar les diferents regles, una per una i en ordre, que hi hagi definides a la/es cadena/es definida/es per aquest "hook" i comparar, per cadascuna d'elles, si el valor de certes característiques del paquet inspeccionat (n'hi ha moltes possibles per mirar: ip d'origen, ip de destí, port d'origen, port de destí, etc) coincideixen amb els establerts a la regla en qüestió. Si és així, s'aplicarà sobre el paquet inspeccionat una determinada acció (acceptar-lo, rebutjar-lo, etc) i, eventualment, es finalitzarà el processament d'aquest paquet ; si no, se seguirà comprovant aquestes característiques amb la següent regla definida dins de la cadena, fins arribar al final. En aquest cas en el que les característiques del paquet processat no concordin amb cap de les regles definides, se li aplicarà la "policy" per defecte.

Les comandes més comunes per gestionar les regles són:

sudo nft add rule <familia> <taula> <cadena> [handle nº] ...
sudo nft insert rule <familia> <taula> <cadena> [handle nº] ...
sudo nft replace rule <familia> <taula> <cadena> handle nº
sudo nft delete rule <familia> <taula> <cadena> handle nº
I les ja conegudes *sudo nft -nn -a list ruleset* i *sudo nft flush ruleset*

NOTA: La comanda *nft add ...* afegeix la regla al final de la cadena (és a dir, després de totes les regles que ja hi existeixin) i la comanda *nft insert ...* l'afegeix en primera posició (és a dir, abans de totes elles) . El paràmetre *handle* en aquestes dues comandes serveix per alterar aquesta ubicació per defecte de la regla dins de la cadena: concretament a la comanda *nft add ...* l'afegeix darrera del handle indicat i la comand*a nft insert ...* l'afegeix abans d'aquest.

NOTA: El paràmetre *handle* en les comandes *nft replace ...* i *nft delete ...* serveix per no haver d'indicar tota la regla explícitament a l'hora de reemplaçar-la (o eliminar-la) sinó identificar-la simplement amb la seva posició dins la cadena

Característiques a inspeccionar dels paquets

Els punts suspensius a les comandes *nft add/insert* indiquen que a partir d'aquí s'hauran d'indicar les característiques dels paquets a inspeccionar i l'acció corresponent a executar. Respecte les característiques a inspeccionar, algunes de les més habituals són:

ether saddr <dirMAC> : MAC d'origen igual al valor indicat
ether daddr <dirMAC> : MAC de destí igual al valor indicat
ether type <tipus> : Tipus de paquet. Poden ser tres: "arp", "ip" o "vlan"

ip saddr <dirIP> : IP d'origen igual al valor indicat. Pot ser tant de host com de xarxa (IP/màscara)
ip saddr != <dirIP> : IP d'origen diferent del valor indicat. Pot ser tant una de host com de xarxa (")
ip saddr <dirIP1>-<dirIP2> : IPs possibles d'origen dins del rang indicat (ambdós inclosos)
ip saddr != <dirIP1>-<dirIP2> : IPs possibles d'origen fora del rang indicat (ambdós inclosos)
ip saddr {<dirIP1>, <dirIP2,...} : IPs possibles d'origen dins del conjunt indicat
ip saddr != {<dirIP1>, <dirIP2,...} : IPs possibles d'origen fora del conjunt indicat

ip daddr <dirIP> : IP de destí igual al valor indicat. Pot ser tant de host com de xarxa (IP/màscara)
ip daddr != <dirIP> : IP de destí diferent del valor indicat. Pot ser tant una de host com de xarxa (")
ip daddr <dirIP1>-<dirIP2> : IPs possibles de destí dins del rang indicat (ambdós inclosos)
ip daddr != <dirIP1>-<dirIP2> : IPs possibles de destí fora del rang indicat (ambdós inclosos)
ip daddr {<dirIP1>, <dirIP2,...} : IPs possibles de destí dins del conjunt indicat
ip daddr != {<dirIP1>, <dirIP2,...} : IPs possibles de destí fora del conjunt indicat

ip ttl <n> : Valor de la capçalera TTL igual al valor indicat
ip protocol <xxx> : Protocol de nivell de transport del paquet. Es pot indicar: "tcp", "udp", "icmp" i més
NOTA: En ip6 seria *nexthdr <xxx>*

udp sport <n> : Port d'origen UDP igual al valor indicat
udp sport != <n> : Port d'origen UDP diferent al valor indicat
udp sport <n1>-<n2> : Ports d'origen UDP dins del rang indicat (ambdós inclosos)
udp sport != <n1>-<n2> : Ports d'origen UDP fora del rang indicat (ambdós inclosos)
udp sport {<n1>, <n2>...}: Ports d'origen UDP dins del conjunt indicat
udp sport != {<n1>, <n2>...}: Ports d'origen UDP fora del conjunt indicat

udp dport <n> : Port de destí UDP igual al valor indicat
udp dport != <n> : Port de destí UDP diferent al valor indicat
udp dport <n1>-<n2> : Ports de destí UDP dins del rang indicat (ambdós inclosos)
udp dport != <n1>-<n2> : Ports de destí UDP fora del rang indicat (ambdós inclosos)
udp dport {<n1>, <n2>...}: Ports de destí UDP dins del conjunt indicat
udp dport != {<n1>, <n2>...}: Ports de destí UDP fora del conjunt indicat

tcp sport <n> : Port d'origen TCP igual al valor indicat
tcp *sport != <n>* : Port d'origen TCP diferent al valor indicat
tcp sport <n1>-<n2> : Ports d'origen TCP dins del rang indicat (ambdós inclosos)
tcp sport != <n1>-<n2> : Ports d'origen TCP fora del rang indicat (ambdós inclosos)
tcp sport {<n1>, <n2>...}: Ports d'origen TCP dins del conjunt indicat
tcp sport != {<n1>, <n2>...}: Ports d'origen TCP fora del conjunt indicat

tcp dport <n> : Port de destí TCP igual al valor indicat
tcp dport != <n> : Port de destí TCP diferent al valor indicat
tcp dport <n1>-<n2> : Ports de destí TCP dins del rang indicat (ambdós inclosos)
tcp dport != <n1>-<n2> : Ports de destí TCP fora del rang indicat (ambdós inclosos)
tcp dport {<n1>, <n2>...}: Ports de destí TCP dins del conjunt indicat
tcp dport != {<n1>, <n2>...}: Ports de destí TCP fora del conjunt indicat

tcp flags <nom> : Flag ("syn","ack","fin","rst","psh","urg","ecn","cwr") igual al valor indicat
tcp flags != <nom> : Flag diferent del valor indicat
tcp flags {<nom1>, <nom2>...} : Flags dins del conjunt indicat
tcp flags != {<nom1>, <nom2>...} : Flags fora del conjunt indicat

tcp sequence <n> : Valor del camp "Seq" de la capçalera TCP igual al valor indicat
tcp ackseq <n> : Valor del camp "Ack" de la capçalera TCP igual al valor indicat
tcp window <n> : Valor del camp "Window" de la capçalera TCP igual al valor indicat

icmp type <tipus> : Tipus de paquet ICMP igual al valor indicat. Alguns dels valors possibles són: *echo-request, echo-reply, destination-unreachable, redirect, time-exceeded*, entre altres. També es pot escriure *icmp type {<tipus1>,<tipus2>...}* per indicar un conjunt de tipus
icmp code <n> : Codi del paquet ICMP igual al valor indicat. També es pot escriure *icmp code != <n>*, *icmp code <n1>-<n2>*, *icmp code != <n1>-<n2>* o *icmp code {<n1>,<n2>...}* amb el significat habitual

ct state <estat1>,<estat2>,... : L'estat de la connexió a la qual pertany el paquet és igual a l'indicat. Els valors possibles són: *new* (el paquet forma part d'un inici de nova connexió), *established* (el paquet pertany a una connexió ja establerta), *related* (el paquet es correspon a un inici de nova connexió però que està relacionada amb alguna altra connexió prèviament existent; això és habitual en transferències FTP o errors ICMP) i *invalid* (el paquet no està associat a cap connexió coneguda).

iifname <nomTarja> : El paquet inspeccionat entra a la tarja de xarxa indicada
oifname <nomTarja> : El paquet inspeccionat surt de la tarja de xarxa indicada
NOTA: Note that there's another characteristic named "iif" (and "oif") but these ones are in reality a match on the integer which is the index of the interface inside of the kernel. Userspace is converting the given name to the interface index when the nft rule is evaluated (before being sent to kernel). A consequence of this is that the rule can not be added if the interface does not exist. An other consequence, is that if the interface is removed and created again, the match will not occur as the index of added interfaces in kernel is monotonically increasing. Thus, "oif" is a filter faster than "oifname" but it can lead to some issues when dynamic interfaces are used. On the other hand, "oifname" has a performance cost because a string match is done instead of an integer match.
iiftype <tipusTarja> : El paquet inspeccionat entra a una tarja del tipus indicat. Alguns dels valors possibles són: *ether* o *loopback* entre altres.
oiftype <tipusTarja> : El paquet inspeccionat surt d'una tarja del tipus indicat

limit rate <n/t> : Es detecta una ràtio per sota de nº paquets/temps,
on temps es pot indicar amb les paraules "second", "minute", "hour", "day" o "week"
Per exemple, *limit rate 400/second*
limit rate <n b/t> : Es detecta una ràtio per sota de nº bytes/temps,
on el valor "b" s'ha d'indicar amb les paraules "bytes", "kbytes", "mbytes" o "gbytes"
Per exemple, *limit rate 400 kbytes/second*
NOTA: Recordeu que la MTU d'un paquet Ethernet estàndar és de 1500 bytes
limit rate over <n/t> : Es detecta una ràtio per sobre de nº paquets / temps
limit rate over <n b/t> : Es detecta una ràtio per sobre de nº bytes / temps
NOTA: Es pot afegir l'opció *burst <n> packets* o *burst <n> bytes* (o *kbytes*, *mbytes*, etc) per indicar que es podrà superar per aquesta quantitat el límit de la ràtio indicada
NOTA: Les expressions *limit* ... són útils per protegir-se d'escaneigs de passwords/ports, DoS...

NOTA: A més de l'operador de desigualtat (!=), el qual també es pot escriure amb la notació "ne", també es poden fer servir els següents operadors (segons el significat del camp a inspeccionar: lt(\<) , gt(\>) , le(\<=) , ge(\>=)

Accions

Les accions que es poden realitzar amb cada paquet són:

Associades a regles pertanyents a cadenes de tipus "filter"

accept : El paquet s'envia amb èxit a l'aplicació destí, i s'atura el seu processament

drop : El paquet és rebutjat (sense notificar-ho al remitent), i s'atura el seu processament

reject : Igual que *drop*, però envia un missatge ICMP/ICMPv6 "port unreachable" al remitent informant del rebuig. Es pot indicar un altre tipus de missatge ICMP si s'escriu *reject with icmp type xxx* on "xxx" pot ser "host unreachable", "net-unreachable", "prot-unreachable", "host-prohibited", "net-prohibited" o "admin-prohibited". Amb ICMPv6 els motius són uns altres però afortunadament es pot usar el wrapper "icmpx" que mapeja al motiu que pertoqui segons sigui IP o Ipv6.

log : S'envia una notificació relativa a la detecció del paquet inspeccionat al registre del sistema en forma de missatge de l'estil "IN=... OUT=..." (si el registre és de tipus Systemd, aquest registre es pot gestionar mitjançant la comanda *journalctl -k)* . El processament del paquet continuarà sense alteracions en espera de trobar una altra acció com *accept*, *drop* o *reject*. Es pot indicar un determinat nivell de "priority" si s'escriu *log level xxx* (on "xxx" pot ser la paraula *debug, info, notice,warning, err, crit, alert, emerg*). També es pot indicar un determinat prefixe en els missatges a guardar al registre si s'escriu *log prefix \"Hola\"* Òbviament es poden combinar aquestes dues possibilitats: *log level emerg prefix \"Pepe\"*

counter: Indica que s'utilitzarà un comptador de paquets i bytes per la regla en qüestió. Per veure els valors en un moment donat dels comptadors definits a les regles d'una taula concreta, es pot executar la comanda *sudo nft -a list table <taula>* Més informació a https://wiki.nftables.org/wiki-nftables/index.php/Stateful_objects

dup to x.x.x.x : Duplica el paquet en qüestió i envia la còpia a la IP/Ipv6 indicada.

Les accions *log, counter* i *dup* es poden escriure a una mateixa regla abans de les accions *accept, drop* o *reject*

Associades a regles pertanyents a cadenes de tipus "nat"

snat x.x.x.x : Només útil a cadenes associades al hook "postrouting": canvia la IP d'origen (i opcionalment també el port d'origen si s'usa la notació *snat x.x.x.x:nº*) del paquet en qüestió per la IP (i port) indicats

masquerade: Només útil a cadenes associades al hook "postrouting": igual que *snat* però canvia automàticament la IP de l'origen per la IP de la tarjeta de sortida sense que calgui especificar-la "a mà"

dnat x.x.x.x : Només útil a cadenes associades al hook "prerouting": canvia la IP de destí (i opcionalment també el port de destí si s'usa la notació *dnat x.x.x.x:nº*) del paquet en qüestió per la IP (i port) indicats

redirect to <n> : Només útil a cadenes associades al hook "prerouting": només canvia el port de destí del paquet en qüestió pel número de port indicat

Altres

jump <nomCadena>: Continúa el processament del paquet a la cadena indicada. Un cop el paquet retorni d'aquesta cadena (si retorna), se seguirà el processament a la regla següent. Una alternativa similar és l'acció *goto <nomCadena>* , en la qual els paquets passen a la cadena indicada però no retornen mai a la cadena "base"; en aquest cas la política per defecte aplicada al paquet serà la política per defecte de la cadena "base" on es començà a processar el paquet.

return : Retorna el processament de l'actual cadena a la cadena des d'on es va cridar mitjançant *jump*. Si no es va cridar des de cap (perquè és una cadena "base") equival a *accept*

continue : No es realitza cap processament del paquet i es continua avaluant la següent regla

queue : Envia el paquet a l'espai d'usuari i s'atura el seu processament. Ha d'haver executant-se una aplicació desenvolupada amb la llibreria "libnetfilter_queue" per tal de què pugui recollir correctament aquest paquet. Un exemple d'aplicació d'aquest tipus podria ser un NIDS, com Suricata

Exemples de regles

Les regles que diuen quins paquets (de quines cadenes i amb quines característiques concretes) han de ser rebutjats i quins no, s'han d'escriue en ordre. És molt important això perquè en el moment que un paquet coincideix amb la descripció que fa d'ell una regla, s'executa l'acció i ja no es continua investigant cap més regla que hi hagi per sota Si un paquet no coincideix amb cap regla existent, després d'haver passat per totes, se li aplicarà la "policy" corresponent a la cadena on estigui encuat.

NOTA: A partir de l'explicat al paràgraf anterior, és evident que quan més específica sigui una regla, menys paquets coincidiran amb la descripció i per tant a menys paquets afectarà. És per això que el més freqüent és indicar primer les regles que solen tenir més coincidències i deixar per després les regles més genèriques.

NOTA: Estem suposant en qualsevol cas que les regles a tractar es troben sempre en cadenes amb la mateixa prioritat

Als exemples següents farem la suposició de que tenim una taula anomenada "filter" que inclou tres cadenes de tipus "filter" anomenades "input", "output" i "forward", associades als hooks homònims. Aquests noms són, de fet, els més comuns en les regles predefinides dins de les distribucions més importants. Per tant, el primer que haurem de fer és executar les següents comandes, i a partir d'elles, provar els diferents exemples que mostrarem a continuació:

sudo nft add table inet filter
sudo nft add chain inet filter input { type filter hook input priority 0 \; policy accept \; }
sudo nft add chain inet filter output { type filter hook output priority 0 \; policy accept \; }
sudo nft add chain inet filter forward { type filter hook forward priority 0 \; policy accept \; }

També suposarem als exemples que el sistema on està funcionant Nftables té dues tarjes: "enp0s3" i "enp0s8". La primera tindrà la IP 192.168.1.1 i és per on escoltaran els diferents servidors que tinguem configurats (SSH, HTTP, DHCP, etc) a la nostra LAN 192.168.1.0/24; la segona tindrà la IP 10.0.1.1 i servirà per connectar el sistema amb "l'exterior".

A continuació mostrem exemples de regles individuals, com ara:

*Bloquejar tot el tràfic que entri per la tarja de xarxa enp0s3:
sudo nft add rule inet filter input iifname enp0s3 drop

*Bloquejar (només) el tràfic que entri per la tarja de xarxa enp0s3 i que a més provingui d'una IP concreta (també es podria bloquejar a la xarxa sencera especificant 192.168.1.0/24):
sudo nft add rule inet filter input iifname enp0s3 ip saddr 192.168.1.234 drop

*Bloquejar (només) tot el tràfic que vagi dirigit a dues IPs concretes, establint comptadors:
sudo nft add rule inet filter output oifname enp0s8 ip daddr {8.8.8.8, 8.8.4.4} counter drop

*Bloquejar (només) tot el tràfic ICMP que entri per qualsevol tarja:
sudo nft add rule inet filter input ip protocol icmp drop

*Bloquejar (només) el tràfic ICMP que entri per qualsevol tarja que sigui de tipus "echo-request":
sudo nft add rule inet filter input icmp type echo-request drop

*Bloquejar (només) el tràfic que entri per la tarja de xarxa enp0s3 (provinent de qualsevol lloc) i que a més vagi dirigit a certs ports concrets del nostre sistema (en aquest cas, els ports 22 i 80 TCP):
sudo nft add rule inet filter input iifname enp0s3 tcp dport {22, 80} drop

*Bloquejar (només) el tràfic que entri per la tarja de xarxa enp0s3 (provinent de qualsevol lloc) i que a més vagi dirigit a un port determinat del nostre sistema (en aquest cas, el port 22 TCP) i que, a més, guardi un registre sobre aquest fet:
sudo nft add rule inet filter input iifname enp0s3 tcp dport 22 log prefix \"Accés al port 22 detectat\" drop

*Bloquejar (només) el tràfic que entri per la tarja de xarxa enp0s3 que a més provingui d'una IP concreta i que a més vagi dirigit a un port determinat del nostre sistema (en aquest cas, el port 22 TCP):
sudo nft add rule inet filter input iifname enp0s3 ip saddr 192.168.1.234 tcp dport 22 drop

*Acceptar tots els paquets que provinguin de l'exterior però només pertanyents a connexions prèviament obertes des del nostre sistema (aquesta regla és útil -si la política per defecte és *drop*- per evitar inicis de connexió generats des de l'exterior ja que només accepta els paquets d'entrada corresponents a les respostes de les connexions obertes des del sistema local):
sudo nft add rule inet filter input ct state established,related accept
Un altre exemple: si una màquina tingués TOT el tràfic entrant i sortint tancat, per aconseguir que un servidor SSH instal.lat en aquella màquina pugui comunicar-se normalment amb els client, hauríem d'escriure les següents dues regles:
sudo nft add rule inet filter input iifname enp0s8 tcp dport 22 ct state new,established accept
sudo nft add rule inet filter output oifname enp0s8 tcp sport 22 ct state established accept

Cal tenir en compte que les regles es composen d'expressions que s'avaluen d'esquerra a dreta. Això és important perquè si el valor de la primera expressió coincideix amb la característica en qüestió del paquet analitzat en aquell moment, es passarà a avaluar la següent expressió i així successivament fins arribar a la darrera expressió al final de la regla. Però si el valor d'una expressió ja no coincideix, no es continua analitzant res de la regla actual i es passarà a analitzar la següent. Així doncs, no és el mateix

sudo nft add rule inet filter input limit rate 10 kbytes/minute tcp dport 80 log

que

sudo nft add rule inet filter input tcp dport 80 limit rate 10 kbytes/minute log

En el primer cas només es registren tots els paquets que, pertanyents a un tràfic amb una ràtio de 10 kbytes/minut, vagin dirigits al port 80. En el segon cas es registren només els paquets dirigits al port 80 que mantinguin una ràtio de 10 kbytes/minut. És a dir: en el primer cas s'està considerant qualsevol tipus de tràfic (ssh, ftp, etc) en el límit de ràtio i seguidament només un subconjunt d'aquest, el tràfic que és http, és registrat mentre que en el segon cas només el tràfic http es conta per calcular la ràtio.

Ja hem comentat que és molt important fixar-se en l'ordre de les regles. La primera regla que coincideixi amb les característiques del paquet processat és la que s'aplicarà (a mateixa prioritat de la cadena) i ja no es continuarà mirant altres regles. A continuació es presenta un exemple de tallafocs en un sistema funcionant com servidor HTTP (i SSH només per un client determinat):

```
#Preparació
sudo nft flush ruleset
sudo nft add table inet filter
sudo nft add chain inet filter input      { type filter hook input     priority 0 \; policy drop \; }
sudo nft add chain inet filter output     { type filter hook output    priority 0 \; policy accept \; }
sudo nft add chain inet filter forward { type filter hook forward priority 0 \; policy drop \; }
#Regles
sudo nft add rule inet filter input iifname loopback accept
sudo nft add rule inet filter input iifname enp0s8 tcp dport 80 ct state new accept
sudo nft add rule inet filter input iifname enp0s8 ip saddr 192.168.1.234 tcp dport 22 ct state new accept
sudo nft add rule inet filter input ct state established,related accept
#Accepto peticions ping de fora i també respostes ping de fora
sudo nft add rule inet filter input iifname enp0s8 icmp type echo-request accept
sudo nft add rule inet filter input iifname enp0s8 icmp type echo-reply accept
```

En general, el que s'hauria de procurar aconseguir amb les regles que s'escriguin és:

*Permetre tot el tràfic d'entrada i sortida per la interfície Loopback
*Tenir com a "policy" general que tots els paquets puguin sortir del sistema però que no puguin entrar
*En aquest sentit, permetre els paquets entrants només quan formin part de connexions ja establertes o de connexions noves que estiguin relacionades amb prèviament existents o quan pertanyin a una connexió nova només si van dirigits a un port adient

Scripts Nftables

En comptes d'haver d'escriure les regles una per una al terminal, una opció molt més còmoda (i útil) és escriure-les dins d'un script i executar-lo en un determinat moment (via la comanda *nft -f /ruta/script*) o bé fer que aquest script es llegeixi automàticament en iniciar-se el servei nftables (via la comanda *nft -f /ruta/script* present a la línia ExecStart= del fitxer de configuració d'aquest servei -que sol ser "/usr/lib/systemd/system/nftables.service", tal com ja hem vist). No obstant, els scripts que podem indicar amb el paràmetre *-f* de *nft* no són scripts Bash sinó "scripts Nft", els quals tenen característiques una mica diferents.

Els "scripts Nft" poden tenir dues sintaxis diferents, que són completament equivalents. A continuació es presenta una de les dues, la qual es pot veure que és molt semblant a la sintaxi emprada a les comandes de terminal:

```
#!/usr/sbin/nft -f
flush ruleset
add table inet filter
add chain inet filter input { type filter hook input priority 0; policy drop;}
add rule inet filter input iifname lo accept
```

Es poden afegir comentaris d'una línia començant-les amb el símbol "#" (igual que als shells Bash). Es pot dividir una mateixa comanda en vàries línies amb el símbol "\" al final de la línia partida (igual que als shells Bash). Remarcar també que no cal escapar els punts i comes perquè ara estan sent interpretats directament per Nft i no hi ha possibilitat de confusió amb l'intèrpret Bash.

També es poden definir variables amb la paraula *define* (i després usar el seu valor precedint el seu nom amb el símbol "$", igual que als shells Bash) i també es poden incloure, amb la paraula *include*, scripts Nft dins d'altres fitxers Nft indicant la seva ruta absoluta (o bé només el seu nom si l'script Nft a incloure es troba dins de la carpeta "/etc/nft" o de qualsevol altra carpeta indicada amb el paràmetre *-I* de la comanda *nft*). Per exemple, suposant que l'script anterior ho haguéssim guardat en un fitxer anomenat "base.nft", podríem tenir un altre script Nft que fes ús d'ell i que afegís més regles a la taula "filter" així:

```
#!/usr/sbin/nft -f
include "base.nft"
define variable=8.8.8.8
define conjunt={ 80, 433, 22 }
add rule inet filter input ip saddr $variable tcp dport $conjunt counter
```

NOTA: It is also possible to use named sets (they will be created as an object belonging to a specific table). To declare one set containing, for instance, ipv4 addresses, you should do the following: *nft add set inet filter pepe { type ipv4_address\;}*
To add elements to the set: *nft add element inet filter pepe { 192.168.1.4, 192.168.1.5 }*
Listing the set is done via: *nft list set inet filter pepe*
The set can then be used in rule: *nft add rule inet filter input ip saddr @pepe drop*
Later when a "new bad boy" is detected, you can evolve the set content: *nft add element inet filter pepe{ 192.168.1.6 }*
It is possible to remove element from an existing set: *nft delete element inet filter pepe { 192.168.1.5 }*
and to delete a set: *nft delete set inet filter pepe*
Named sets can have several characteristics like "type", "timeout", "flags", "elements" or "size", among others. "type"'s value can be: "ipv4_addr" (IPv4 address), "ipv6_addr" (IPv6 address), "ether_addr" (Ethernet address), "inet_proto" (inet protocol type) or "inet_service" (internet service like read tcp port for example); "timeout"'s value is a time string like "1d2h3m4s" which defines how long an element stays in the set; "flags"'s value can be: "constant" (set content may not change while bound), "interval" (set contains intervals) or "timeout" (elements can be added with a timeout); "elements" initializes the set with member elements, "size" limits the maximum number of elements of the set.

L'altra sintaxi és la que mostra les comandes *nft list table nomtaula* o *nft list ruleset* . És a dir, l'equivalent a l'script "base.nft" anterior amb aquesta altra sintaxi seria:

```
flush ruleset
table inet filter {
        chain input {
                type filter hook input priority 0; policy drop;
                iifname lo accept
        }

}
```

De fet, una manera d'obtenir un fitxer de regles d'aquestes característiques és, si aquestes regles ja estan carregades, simplement executant la comanda *sudo nft list ruleset > base.nft*

NOTA: Si no es vol executar un script Nft amb la comanda *nft -f nomScript* és possible assignar-li permisos d'execució per tal de poder-lo executar directament així: *./nomScript* . És justament per això que existeix la línia inicial *#!/usr/sbin/nft -f*

NOTA: Per més informació, consulteu https://wiki.nftables.org/wiki-nftables/index.php/Scripting

Existeix una altra possibilitat de la comanda nft que és utilitzar el seu paràmetre *-i* ; d'aquesta manera s'entra en un "shell" propi des d'on es poden escriure les mateixes ordres que s'escriurien en un script però de forma interactiva.

NAT

Per a què els ordinadors puguin establir comunicacions entre ells han d'estar identificats i això es fa amb l'adreçament: a cada ordinador s'hi assigna una adreça IP que l'identifica de manera única en la xarxa, No obstant, a causa de la limitació de l'espai d'adreces del protocol IPv4 no és possible que a Internet tots els ordinadors tinguin una única adreça. Per solucionar aquest problema, el que es fa és disposar a voluntat d'una quantitat (generalment gran) d'adreces privades per als diferents ordinadors de la nostra xarxa local (assignades per nosaltres ja sigui estàtica o dinàmicament) però utilitzar només una sola adreça pública (assignada per l'ISP que tinguem contractat) per comunicar tots aquests ordinadors amb Internet. Per a què això funcioni cal tenir a la nostra xarxa local un sistema que, quan un ordinador de la nostra xarxa vulgui comunicar-se amb Internet, tradueixi la seva adreça privada concreta a la única adreça pública existent. Aquest sistema es diu NAT ("Network Adress Translation") i l'implementen els encaminadors, que són els dispositius que fan de "frontera" entre la xarxa local i la pública.

> NOTA:Cal tenir en compte que NAT és una sol.lució de compromís perquè presenta una sèrie de problemes. Per exemple, el retard, ja que en efectuar la traducció de cada adreça en els encapçalaments dels paquets es produeix un alentiment. A més per cada paquet s'ha de decidir si s'efectua la traducció o no, que també alenteix el procés. Tampoc es pot fer un seguiment dels paquets, ja que poden passar per diversos encaminadors executant el NAT i això fa que siguin més difícils de seguir, múltiples salts de NAT i pot provocar que algunes aplicacions no funcionin perquè s'amaguen les adreces de destinació i d'origen. Tots aquests problemes se sol.lucionen amb l'IPv6.

Recordem els rangs d'IPs privades que es poden fer servir (tota IP que no estigui dins d'aquests rangs, és pública i, per tant, cau fora de la nostra administració): de classe A tenim la xarxa sencera 10.0.0.0/8 ; de classe B tenim el rang de la IP 172.16.0.0/16 fins a la 172.31.0.0/16 i de classe C tenim el rang de la IP 192.168.0.0/24 fins a la 192.168.254.0/24

Teoria: SNAT

El cas descrit anteriorment, és a dir, el canvi en l'adreça d'origen dels paquets de l'adreça privada d'un node de la LAN per la pública del sistema enrutador-tallafocs quan el trànsit és originat des de la xarxa local cap a l'exterior és el tipus de NAT més habitual amb diferència i és el que se sol entendre quan es parla de NAT genèricament però més específicament hauríem de parlar de **SNAT**, de "Source NAT"). SNAT és un mecanisme que permet compartir una direcció IP pública per molts equips que tenen cadascú una IP privada diferent. D'aquesta manera, es permet l'accés a "Internet" a tots aquests equips minimitzant el problema de l'escassetat de direccions IPv4 (públiques).

En el SNAT el destí del paquet "veurà" que l'origen del paquet és el "router" i, per tant, serà a ell a qui li reenviarà la resposta (més en concret, a un port determinat del "router"). Quan aquesta resposta arribi al "router", aquest consultarà una taula (la "taula NAT") que tindrà en memòria on s'associa la IP d'origen real (i també el port d'origen) amb la seva IP (i el port que ha utilitzat el router per connectar amb l'exterior) i reenviarà convenientment aquesta resposta a l'origen real.

A la taula NAT hi ha quatre dades: IP origen, port origen, IP router, port router. D'aquesta manera, el router és capaç de reenviar una resposta rebuda per un determinat port seu a un determinat port de l'origen. Per tant, és perfectament possible que cada origen pugui tenir diferents connexions en paral.lel (cadascuna creades en un port diferent) perquè el router saps quina és quina en tot moment. Normalment, el port que fa servir el router per accedir a l'exterior és el mateix que el port utilitzat per l'origen, però si hi hagués més d'un origen utilitzant el mateix port, el router llavors automàticament en farà servir un altre (i,per tant, també canviarà al paquet el port d'origen).

Resumint,

1.-Un equip d'una LAN amb IP privada (suposarem 192.168.3.14) vol accedir a una pàgina web (port 80 TCP) com www.lerele.net. Realitza la consulta DNS i obté que l'equip que allotja la pàgina té la direcció 76.74.254.126

2.-Consulta la seva taula d'encaminament i com no l'equip destí no està a la seva xarxa, envia la petició de pàgina a la seva porta d'enllaç definida (que suposarem que té la IP 192.168.3.254)

3.-El gateway, en comprovar que la direcció IP de destí no és la seva, l'enviarà a la seva pròpia porta d'enllaç, que ja serà una direcció IP pública. Abans de què el paquet surti per la seva tarja de xarxa externa, però, el gateway canvia la direcció IP d'origen (192.168.3.14) per la seva direcció IP pública (que suposarem que és 80.58.1.14) i es guarda la petició en una taula en memòria (anotant també el port d'origen, que suposarem que és el 50158 TCP).

4.-El paquet viatja per Internet saltant de router a router fins que arriba al seu destí

5.-L'equip 76.74.254.126 rep la petició des de la direcció 80.58.1.14 i la respon. Per tant, el paquet de resposta tindrà com direcció IP d'origen 76.74.254.126, direcció IP de destí 80.58.1.14, port d'origen 80 TCP i port de destí 50158 TCP

6.-Aquesta resposta arriba a la tarja externa del gateway de la nostra LAN, que consulta la seva taula en memòria i comprova (gràcies al port origen allà registrat) que correspon amb una petició realitzada des de l'equip 192.168.3.14, així que modifica la direcció IP de destí del paquet per aquesta IP i l'envia directament.

Hi ha un tipus de "Source NAT" específic anomenat "**IP masquerading**", el qual és el que passa quan la direcció IP pública que sustitueix a la IP d'origen és dinàmica (el cas més habitual en connexions a Internet domèstiques). En el "Source NAT" clàssic la IP pública és estàtica (uns circumstància molt rara avui dia)

Teoria: DNAT

Hi ha un altre tipus de NAT on en l'adreça de destinació si el trànsit és originat des de l'exterior cap a un servidor que tinguem funcionant a la xarxa local; aquest tipus de NAT -no tan habitual- se sol anomenar **DNAT**, de "Destination NAT" o també "NAT invers". El que fa aquest DNAT és canviar la IP de destí del paquet (que sol ser la IP pública de la tarja externa del nostre gateway) per una IP privada d'alguna màquina de la nostra LAN. Cal tenir en compte que aquest tipus de NAT es realitza només quan el gateway detecta que aquest paquet no es correspon amb cap connexió iniciada des de la nostra LAN (és a dir, ha de ser un equip extern qui inicii la connexió) i té prèviament definida la regla DNAT adequada (si no la té, el paquet es descarta). Aquest tipus de NAT, el qual se sol anomenar també "Port forwarding", s'utilitza quan volem que algun servidor funcionant en una màquina de la nostra LAN sigui accessible des de l'exterior. Els passos concrets són aquests:

1.-Un equip qualsevol d'Internet amb direcció IP pública 150.212.23.6 vol connectar-se per SSH (port 22 TCP) a l'equip "miservidor.midominio.com". Realitza una consulta DNS i obté com a resposta que aquest equip té la direcció IP 85.136.14.7

2.-Estableix la connexió (suposem que el port d'origen que utilitza és 23014 TCP) amb l'equip 85.136.14.7, que resulta ser un dispositiu NAT que no té cap servei SSH escoltant al port 22 TCP però que té una regla DNAT que fa que tot el que li arribi a aquest port ho reenviarà a un equip de la seva xarxa local (suposarem que és el 10.0.0.2). Per tant, canvia la direcció IP de destí (85.136.14.7) per la 10.0.0.2 i registra aquesta associació a una taula en memòria

3.-A l'equip 10.0.0.2 li arriba una sol.licitud al port 22 TCP i la resposta del servidor SSH tindrà les següents característiques: IP d'origen 10.0.0.2, port d'origen 22 TCP, IP de destí 150.212.23.6 i port de destí 23014 TCP

4.-El dispositiu NAT canvia la direcció IP d'origen per la seva direcció IP pública (85.136.14.7) i el paquet de resposta arriba al seu destí.

Hi ha un tipus de "Destination NAT" específic anoemant "PAT" ("Port Address Translation") o simplement "**redirect**", el qual és un mecanisme que només canvia els ports (normalment el de destí) però no les IPs. El PAT és útil quan un servidor ubicat al mateix sistema que Nftables no escolta al seu port estàndard però no volem fer que els client hagin d'especificar manualment el port no estàndard al que han d'anar quan connectin a aquest servidor.

Teoria: la taula NAT

Ja s'implementi SNAT o DNAT, en l'encaminador es manté una taula amb les traduccions que va fent de manera que pugui portar el control de quins paquets corresponen a cada connexió individual encara que totes elles comparteixin la mateixa IP pública.

Si a més de produir canvis en les adreces IP el router també canvia els números de ports (en el cas del SNAT, el port d'origen i en el cas del DNAT el port de destí -en el que popularment s'anomena "port-forwarding" o "redirecció de ports"-) estarem parlant d'un NAT de fa servir la tècnica **PAT** ("Port Address Translation") per realitar la seva funcionalitat. En aquest tipus de S/DNAT (el més habitual amb diferència) l'encaminador fa servir de manera dinàmica números elevats de ports per identificar de manera única la connexió d'un ordinador local a l'exterior. La idea és que s'ha de discriminar a quin ordinador de la xarxa local corresponen les dades que arriben o provenen d'una única adreça pública, ja que si només es tradueixen adreces no es pot diferenciar (perquè tenint una sola adreça pública totes les adreces privades s'acaben traduint a aquesta).

El mecanisme PAT concret implementat en un NAT de tipus SNAT és el següent: l'encaminador registra a la seva taula NAT la IP privada de l'ordinador de la nostra LAN que vol "sortir a Internet" juntament amb el número de port client que hagi obert per iniciar la connexió i vincula aquesta parella de valors, respectivament, amb la seva IP pública i un número de port propi, que pot ser el mateix que l'utilitzat per l'ordinador origen (o no, si ja està "agafat"). D'aquesta manera, aquest port propi permet a l'encaminador saber a quina connexió en concret de les que pugui tenir establertes correspon el trànsit de la xarxa pública. La figura següent mostra un exemple de taula SNAT d'un encaminador fent servir PAT:

RED PRIVADA		INTERNET		Prot.
IP interna	Puerto interno	IP externa	Puerto externo	
10.0.0.13	1050	83.77.100.34	1050	TCP
10.0.0.15	1050	83.77.100.34	12311	TCP

En el cas dels NATs de tipus DNAT, el funcionament del PAT és similar però en aquest cas l'associació no es realitza dinàmicament cada cop (i mentre) que s'estableix la connexió sinò que s'estableix de forma estàtica a la configuració del router, de tal manera que aquest sempre sàpiga que les peticions provinents de l'exterior dirigides a la IP pública del router (no pot ser una altra) i a un determinat port públic del router les haurà de reenviar a una determinada IP privada i, en aquesta, a un determinat port. La majoria de routers comercials ofereixen un apartat específic dins del seu panell de control web per indicar aquesta associació permanent, normalment anomenat "Port Forwarding" o similar.

Pas previ: activar l'"Ip-Forwarding"

El més habitual és utilitzar la taula NAT en un sistema que tingui dues tarjes, per tal de funcionar com a "router", on es rep el tràfic per una tarja i es redirigeix a l'altra, que és per on surt. No obstant, per a què aquesta funcionalitat es pugui fer servir, primer hem d'activar el "IP forwarding" entre ambdues tarjes per tal de què el tràfic arribat per una d'elles pugui "passar" internament a l'altra i viceversa.

Per activar l'"IP forwarding" només cal escriure el valor "1" dins de l'arxiu "/proc/sys/net/ipv4/ip_forward" (així, per exemple: *sudo echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward*). Una altra opció equivalent, en comptes d'editar manualment aquest arxiu seria utilitzar la comanda *sysctl*, així: *sudo sysctl -w net.ipv4.ip_forward=1*

NOTA: Recordem que sempre podrem mirar el valor actual de l'opció indicada executant *sysctl net.ipv4.ip_forward*

No obstant, l'activació feta de qualsevol de les maneres anteriors és temporal (és a dir, el valor "1" només es manté mentre la màquina estigui encesa). Si volem que l'IP forwarding" sigui permanent, haurem de modificar l'arxiu anomenat "99-sysctl.conf" que hi ha dins la carpeta "/etc/sysctl.d" de manera que hi aparegui la següent línia: *net.ipv4.ip_forward=1* (i, en el cas de fer servir Ipv6, la línia *net.ipv6.conf.all.forwarding=1*).

NOTA: L'"IP forward" es pot implementar, alternativament, simplement afegint la línia *IPForward=yes* a l'arxiu "*.network" corresponent a qualsevol de les tarjes de xarxa gestionades per systemd-networkd (amb una sola que s'indiqui ja funciona per totes) En realitat, el que fa aquesta línia és executar la comanda *echo 1 > /proc/sys/net/ipv4/ip_forward* automàticament cada cop que la tarja en qüestió s'activi

NAT a la pràctica

Per activar la funcionalitat NAT a Nftables primer hem de crear una taula del tipus específic "nat" i seguidament crear al seu interior alguna cadena o bé de tipus "postrouting" (si es vol implementar SNAT) o bé de tipus "prerouting" (si es vol implementar DNAT) o totes dues, (si es volen implementar els dos tipus de NAT). És a dir (la prioritat 100 és la per defecte en aquest tipus de cadenes):

```
sudo nft add table ip nat
sudo nft add chain ip nat prerouting { type nat hook prerouting priority 100 \; }
sudo nft add chain ip nat postrouting { type nat hook postrouting priority 100 \; }
```

A partir d'aquí:

a) Per fer "IP masquerading" en un sistema que faci de "router", la regla que hem d'escriure és (suposant que la seva tarja interna estigui connectada a la LAN 192.168.1.0/24 i enp0s8 estigui connectada a "Internet"):

sudo nft add rule ip nat postrouting ip saddr 192.168.1.0/24 oifname enp0s8 masquerade

NOTA: L'"IP masquerading" es pot implementar, alternativament, simplement afegint la línia *IPMasquerade=yes* a l'arxiu .network corresponent a la tarja enp0s8 (aquesta línia ja implica *IPForward=yes*). En realitat, el que fa aquesta línia es executar internament la comanda *nftables* anterior

b) Per fer SNAT en un sistema que faci de "router" dirigit a un determinat destí (amb IP 1.2.3.4, per exemple), la regla que hem d'escriure és (suposant que la seva tarja interna estigui connectada a la LAN 192.168.1.0/24 i enp0s8 estigui connectada -o tingui definida una ruta- a aquest destí):

sudo nft add rule ip nat postrouting ip saddr 192.168.1.0/24 oifname enp0s8 snat 1.2.3.4

c) D'altra banda, si tenim a la nostra LAN, per exemple, un servidor HTTP escoltant al port 1234 en comptes del port per defecte (80), podem fer que totes les peticions provinents de l'exterior al nostre port 80 es redireccionin automàticament al port 1234 sense que els clients (en aquest cas, els navegadors) tinguin constància. És a dir, fer un redireccionament PAT amb la següent regla:

sudo nft add rule ip nat prerouting iifname enp0s8 tcp dport 80 redirect to 1234

d) Seguint amb el mateix exemple anterior, si la màquina on tenim el "router-tallafocs" funcionant (que és la que rep les peticions dels navegadors externs d'Internet) no fos la mateixa que la que corre el servidor HTTP (és a dir, si el tallafocs ofereix una protecció al davant del servidor web, el qual funciona al "darrera" -en una màquina amb IP 192.168.1.4, per exemple-), es podria fer igualment una redirecció de màquina (+port). És a dir, fer un DNAT, així:

sudo nft add rule ip nat prerouting iifname enp0s8 tcp dport 80 dnat 192.168.1.4:80

NOTA: Una altra funcionalitat interessant del DNAT, combinada amb l'acció de duplicar paquets del Nftables, és redirigir una còpia de determinats paquets a un destí alternatiu, així: *sudo nft add rule ip nat prerouting dup to 172.20.0.2*