Nftables (I)

Posada en marxa

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)

Una altra passa prèvia que cal fer per poder utilitzar "Nftables" directament és deshabilitar el tallafocs que vingui "de sèrie" a la distribució en qüestió. Això és perquè tenir dos tallafocs (el propi de la distribució i Nftables) funcionant a la vegada pot ocasionar moltes inconsistències entre un i l'altre i, per tant, un funcionament erràtic i impredecible. A Ubuntu el tallafocs que ve per defecte es diu "ufw" i a Fedora es diu "firewalld"; per tant, haurem de fer el següent: *sudo systemctl disable ufw && sudo systemctl stop ufw* (a Ubuntu) o *sudo systemctl disable firewalld && sudo systemctl stop firewalld* (a Fedora).

> NOTA: En realitat, tant "ufw" com "firewalld" no són més que "wrappers" de Nftables (és a dir, eines d'"alt nivell" per gestionar el mateix sistema de tallafocs Netfilter que gestionarem amb l'eina Nftables, però aquesta darrera ens permet fer-ho d'una forma molt més directa i detallada, cosa que "ufw" i "firewalld" no són capaces de fer. Tot i així, aquests "wrappers" continuen essent interessants perquè aporten funcionalitats extra que Nftables "a pèl" no proporciona (com per exemple la comunicació via canal D-Bus), així que els estudiarem també, en un posterior document.

Un cop instal.lat, cal comprovar si Nftables està funcionant: *systemctl status nftables* i si no ho està, executar *sudo systemctl start nftables* i també *sudo systemctl enable nftables*.

> NOTA: 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. 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. Si es vol esbrinar d'on "agafa" Nftables aquestes 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 d'exemple ubicats a la carpeta "/etc/nftables" però que estan tots comentats).

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***

> NOTA: Es pot obtenir la sortida de *nft list ruleset* en format JSON si s'escriu el paràmetre -j, així: *sudo nft -j list ruleset*

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 vulguem 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". Aquestes cadenes són "bàsiques" en el sentit que veuen tot el tràfic que travessa la tarja de xarxa associada, sense assumir res sobre els protocols L2 o L3.

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 de tipus "inet" (anomenada, per conveniència, *"filter"*) per encabir-hi les cadenes relacionades amb el filtratge de paquets d'entrada i sortida del sistema (les quals són, tal com ara veurem, cadenes de tipus "filter") i, si volem que la nostra màquina funcioni a més com a "router", una segona taula també de tipus "inet" (però ara anomenada, per conveniència, *"nat"*), per encabir-hi les cadenes relacionades amb la realització de NAT (les quals són, tal com ara veurem, 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)

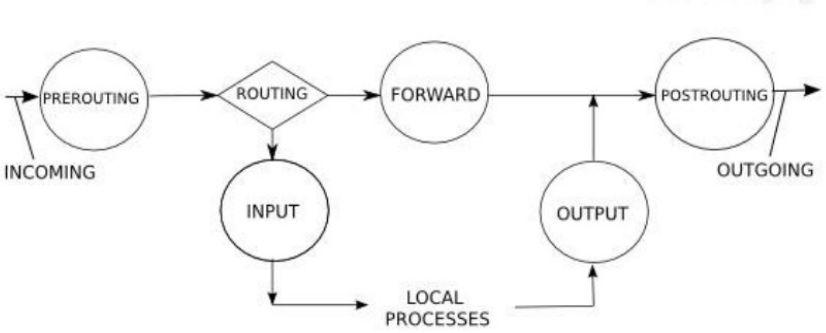
Hooks que només poden assignar-se a una cadena de tipus "nat"

*"**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). També s'utilitza per canviar el port original de destí dels paquets entrants per un altre port diferent (de la mateixa màquina).
*"**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)

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. 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): prioritat de defragmentació
NF_IP_PRI_RAW (-300): prioritat tradicional de la taula "raw" ubicada abans del traçat de connexions
NF_IP_PRI_SELINUX_FIRST (-225): operacions SELinux
NF_IP_PRI_CONNTRACK (-200): operacions de traçat de connexions
NF_IP_PRI_MANGLE (-150): operació "mangle"
NF_IP_PRI_NAT_DST **(-100): Destination NAT**
NF_IP_PRI_FILTER **(0): operacions de filtre** (la taula "filter")
NF_IP_PRI_SECURITY (50): operacions de seguretat com per exemple "secmark"
NF_IP_PRI_NAT_SRC **(100): Source NAT**

NF_IP_PRI_SELINUX_LAST (225): SELinux a la sortida del paquet
NF_IP_PRI_CONNTRACK_HELPER (300): traçat de connexions a la sortida del paquet

NOTA: Cada taula es pot veure com un tallafoc independent i separat. És a dir, si "taula1" accepta un paquet, encara pot ser eliminat a "taula2". Això vol dir que un paquet ha de ser acceptat per tots els "hooks" de cadena del mateix tipus de totes les taules per poder ser admès definitivament. Aquí és on entren en joc les prioritats del "hook": una cadena amb un valor de prioritat inferior a una cadena amb un valor de prioritat més alt sempre s'executarà; si les prioritats són iguals, el comportament no està definit.

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:

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 cadena ja està creada. En tot cas, ambdues comandes no funcionaran si no existeix prèviament la taula que allotjarà la cadena en qüestió
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 pot fer simplement fent *sudo nft add chain <familia> <taula> <cadena>*
NOTA: El valor de la directiva *priority* pot ser, en comptes d'un valor numèric, una paraula clau, com ara "filter"

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 [<familia>]*** i ***sudo nft flush ruleset [<familia>]***

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 comanda *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 (per conèixer més, es pot consultar la web https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes):

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: També es pot escriure *meta l4proto <xxx>*. 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**") indicada activada
tcp flags != <nom> : Flag indicada desactivada
tcp flags {<nom1>, <nom2>...} : Flags dins del conjunt indicat estan activades
tcp flags != {<nom1>, <nom2>...} : Flags fora del conjunt indicat estan activades (les de dins no ho estan)

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 té una combinació de "flags" TCP que no és admissible -com ara tots els "flags" a 1 ("Xmas attack"), tots a 0 ("Null attack"), SYN+RST (no admesa per cap transacció), SYN+FIN (tampoc admesa), etc-)

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: Hi ha altres opcions anomenades "**iif**" i "**oif**", però aquestes concorden no amb el nom d'una tarja de xarxa sinó amb l'índex numèric de la interfície dins del nucli. Una conseqüència d'això és que la regla no es pot afegir si la interfície no existeix. Una altra conseqüència és que si la interfície s'elimina i es torna a crear, la coincidència no es produirà ja que l'índex de les interfícies afegides al nucli augmenta monotònicament. Per tant, "iif/oif" és un filtre més ràpid que "iifname/oifname", però pot provocar problemes quan s'utilitzen interfícies dinàmiques. D'altra banda, "oifname" té un cost de rendiment perquè es fa una coincidència de cadenes en lloc d'una coincidència numèrica.

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

vlan id <n> : El paquet està etiquetat amb la VLAN amb l'identificador indicat
arp operation <tipus> : El paquet inspeccionat és ARP i de tipus petició o resposta segons si el tipus indicat és "**request**" o "**reply**", respectivament (poden haver més tipus i, de fet, l'opció *arp* pot tenir més modificadors a banda de *operation*)

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 ex:, *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 ex: *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...

meta length <n> : Longitud del paquet igual al valor indicat (en bytes)
meta length != <n> : Longitud del paquet diferent al valor indicat (en bytes)
meta length <n1>-<n2> : Longitud del paquet dins del rang indicat (en bytes, ambdós inclosos)
meta length != <n1>-<n2> : Longitud del paquet fora del rang indicat (en bytes, ambdós inclosos)

meta protocol <xxx> : Tipus de paquet. Es pot indicar: "**arp**", "**ip**", "**ip6**" o "**vlan**"

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", "port-unreachable", "host-prohibited", "net-prohibited**" o "**admin-prohibited**". Amb ICMPv6 els motius són uns altres però per sort 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**" o, d'altra banda, "**audit**"). 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>*

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>: Continua 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 num nº : Envia el paquet a l'espai d'usuari (concretament, a una determinada "cua" identificada pel número indicat, a escollir per nosaltres) i atura el seu processament. Ha d'haver executant-se una aplicació desenvolupada amb la llibreria "libnetfilter_queue" "escoltant" a la cua en qüestió 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 similar a l'anterior: 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 output ct state established,related accept
#Accepto peticions ping de fora
sudo nft add rule inet filter input iifname enp0s8 icmp type echo-request accept
sudo nft add rule inet filter output 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 de "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 ***sudo 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
```

A continuació es llisten diferents aspectes a tenir en compte en escriure aquest tipus d'scripts:

* 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).
* Els punts i coma serveixen per separar diferents comandes (normalment si s'escriuen en una sola l'inia, igual que als shells Bash). Remarcar que en els scripts Nft 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, com passava amb les comandes *nft* .
* 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).
* 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
```

L'altra sintaxi possible pels "scripts Nft" és la que mostra les comandes *nft list table nomtaula* o *nft list ruleset* (de fet, una manera de generar aquest tipus de és, si les regles en qüestió ja han sigut carregades, executar simplement *sudo nft list ruleset > base.nft*). Així, l'equivalent a l'script "base.nft" anterior amb aquesta altra sintaxi seria així:

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

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*
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.

En qualsevol cas, s'escriguin com s'escriguin les regles, aquestes són sempre temporals: s'apliquen al moment però només estaran actives mentre el servei Nftables està encès: si s'apaga (o s'apaga el sistema sencer) es perdran. Per a fer-les permanents, cal assegurar-se de què les regles es llegeixin en iniciar el servei, i això s'aconsegueix normalment indicant-les per escrit en el fitxer "/etc/nftables.conf" (a Ubuntu) o "/etc/sysconfig/nftables.conf" (a Fedora) o similar.

EXERCICIS:

Per realitzar aquests exercicis utilitzarem una màquina VirtualBox amb un sistema (Ubuntu o Fedora, Server o Workstation, és igual) que tingui la seva tarja de xarxa *enp0s3* en mode "adaptador pont".

1.-a) Instal.la en aquesta màquina virtual, si no ho està ja, un servidor SSH i posa'l en marxa (*sudo systemctl start sshd*). Tot seguit, crea-hi una taula de tipus *ip* anomenada *filter* que contingui tres cadenes, anomenades *input, forward* i *output* que estiguin associades respectivament als "hooks" homònims, i que tinguin totes tres la seva "policy" establerta a DROP. ¿Des de la màquina real s'hi pot connectar llavors via SSH a la màquina virtual? ¿Funcionen almenys els pings entre ambdues màquines? ¿Per què?

b) Afegeix la regla necessària dins de la cadena *input* (és a dir, mitjançant una comanda que comenci així: *sudo nft add rule ip filter input ...*) per a què es deixi connectar al servidor SSH, però només des de la màquina real

> PISTA: La regla caldrà que especifiqui, doncs, dues característiques dels paquets a inspeccionar: el port de destí del paquet (*tcp dport 22*) i la IP d'origen (*ip saddr x.x.x.x*)

bII) Comprova com, des de la màquina real, encara no pots fer SSH (el client es queda "penjat"). Això és perquè l'entrada al servidor l'hem permès però no pas la sortida (i, per tant, la resposta retornada). Així doncs, a més d'afegir la regla adient a la cadena INPUT, hem d'afegir una altra regla però a la cadena OUTPUT per deixar passar específicament tots els paquets que estiguin relacionats amb les conexions ja establertes des de fora (perquè si no, com veiem, el servidor no hi podrà contestar!!). Per tant, executa la comanda *sudo nft add rule ip filter output ct state established,related accept* i torna a intentar accedir via SSH des de la màquina real a la màquina virtual: ara hauria de poder aconseguir-ho sense problema.

bIII) Posa en marxa una segona màquina virtual amb la seva tarja també en mode "adaptador pont" i prova de connectar-te des d'allà al servidor SSH de la màquina virtual on estem implementant les regles del tallafocs. ¿Pots? ¿Per què? Apaga aquesta segona màquina virtual (ja no caldrà que la facis servir més)

c) Posa en marxa ara a la màquina virtual on està funcionant el servidor SSH un altre servidor, en aquest cas de tipus Netcat escoltant al port 5555/TCP (és a dir, executa la comanda *nc -l -p 5555*). Tot seguit (en un altre terminal perquè el servidor Netcat haurà bloquejat l'actual) afegeix la regla necessària dins de la cadena *input* (és a dir, mitjançant una comanda que comenci així: *sudo nft add rule ip filter input ...*) per a què es deixi connectar a aquest servidor Netcat des de qualsevol origen

> PISTA: La regla caldrà que especifiqui, doncs, només una característica dels paquets a inspeccionar: el port de destí del paquet (*tcp dport 5555*)

cII) Comprova com des de la màquina real pots connectar-te al servidor Netcat anterior sense problemes. I que, de fet, podràs fer-ho des de qualsevol altra màquina client (li pots demanar a un company que ho provi)

> NOTA: En aquest cas també hauria sigut necessari afegir una regla a la cadena OUTPUT per deixés passar tots els paquets relacionats amb les conexions ja establertes des de fora per a què el servidor pugui contestar, però en aquest cas no ha sigut necessari perquè això ja ho vam fer a l'apartat bII) : si et fixes, la regla allà indicada és molt genèrica: només indica connexions ja establertes però no restringeix per número de port o altres característiques, així que val per tots els servidors

2. a) Esborra les regles Nft establertes a l'exercici anterior a la màquina virtual de treball sense eliminar, però l'estructura de la taula i cadenes. Recorda que per eliminar regles concretes primer cal veure el nº de "handle" que hi té associat observant la sortida de la comanda *sudo nft -a list ruleset* i llavors esborrar-la amb la comanda *sudo nft delete rule ip filter input handle nº* (suposant que la regla en qüestió es trobi a la cadena "input" de la taula "filter") o similar.

> NOTA: Recorda que també pots esborrar totes les regles indiscriminadament de totes les cadenes (mantenint aquestes, però, buides) amb la comanda *sudo nft flush ruleset*

b) Crea-hi ara una nova regla (mitjançant una comanda que comenci així: *sudo nft add rule ip filter input ...*) que registri al Journal del sistema (amb l'acció LOG) tots els missatges ICMP de tipus "echo-request" rebuts de l'exterior i que, seguidament, els acceptin (amb l'acció ACCEPT).

> PISTA: La regla caldrà que especifiqui, doncs, el tipus de paquet a inspeccionar (*icmp type echo-request*)

bII) Comprova la regla anterior fent pings des d'un ordinador extern (com ara la màquina real) i observant el Journal en temps real amb la comanda *journalctl -k -f* ¿Funciona el "ping", però? ¿Per què?

c) Crea la regla necessària (mitjançant una comanda que comenci així: *sudo nft add rule ip filter output ...*) per permetre sortir de la màquina virtual als missatges ICMP de tipus "echo-reply" (que en principi seran resposta als missatges ICMP entrants anteriors).

> NOTA: Recorda que en el tràfic ICMP (o UDP!) no té sentit la regla "ct state established,related" perquè no existeix cap connexió ni estat que tenir en compte

cII) Comprova la regla anterior tornant a fer pings des d'un ordinador extern (com ara la màquina real) ¿Funciona el "ping", ara? ¿Què passa, però, si s'intenta fer un "ping" des de la màquina virtual cap algun destí exterior qualsevol? ¿Per què?

3.-a) Esborra les regles Nft establertes a l'exercici anterior a la màquina virtual de treball sense eliminar, però l'estructura de la taula i cadenes i afegeix ara la regla següent: *sudo nft add rule ip filter output ip protocol icmp ip daddr 8.8.8.8 counter* ¿Per a què serveix i quina comanda hauries de fer servir per comprovar-ho?

b) ¿Què pretén evitar aquesta regla: *sudo nft add rule inet filter input ip saddr 127.0.0.0/8 iifname != lo drop*?

4.-A partir de la configuració de Nftables mostrada a continuació, respon les següents preguntes:

a) ¿Per a què serveixen la primera regla de la cadena "input" i la primera de la cadena "output"?

aII) ¿Quina relació hi ha entre la segona regla de la cadena "input" i la segona de la cadena "output"?

aIII) ¿I entre la tercera regla de la cadena "input" i la tercera de la cadena "output"?

aIV) ¿I entre la quarta regla de la cadena "input" i la quarta de la cadena "output"?

aV) ¿Per a què serveix la cinquena regla de la cadena "input"? ¿I l'única regla de la cadena "forward"?

```
#!/usr/sbin/nft -f
flush table inet filter
table inet filter {
        chain input { type filter hook input priority 0; policy drop;
                iif lo accept
                ct state {established,related} accept
                tcp dport {80,443} counter log accept
                icmp type echo-request accept
                ip saddr {192.168.1.1,192.168.1.2} tcp dport 22 log accept
        }
```

```
        chain output { type filter hook output priority 0; policy drop;
                oif lo accept
                ct state new accept
                ct state {established,related} tcp sport {80,443} accept
                icmp type echo-reply accept
        }
        chain forward { type filter hook forward priority 0; policy accept;
                counter log
        }
}
```

NOTA: En comptes d'indicar els números de port, a les regles Nftables es pot escriure alternativament el nom amb el qual aquests números estan associats dins de l'arxiu "/etc/services". Així doncs, podríem escriure "ssh" en comptes de 22, "http" en comptes de 80 o "https" en comptes de 443.

b) Descriu amb les teves paraules les regles que es mostren a continuació de les cadenes "input" i "forward" (suposant que "lan0" és la tarja del tallafocs connectada a la xarxa local i "wan0" la tarja del tallafocs connectada a l'ISP que dóna accés a Internet) i les de la cadena "output" també:

```
#!/usr/sbin/nft -f
define dns_servers = { 1.1.1.1, 8.8.8.8 }
table ip filter {
    chain input {
        type filter hook input priority filter; policy drop;
        iifname "lan0" accept
        iifname "wan0" ip saddr $dns_servers udp sport 53 accept
    }
    chain output {
        type filter hook output priority filter; policy drop;
        oifname "lan0" accept
        oifname "wlan0" ip daddr $dns_servers udp dport 53 accept
    }
    chain forward {
        type filter hook forward priority filter; policy drop;
        iifname "lan0" oifname "wan0" accept
        iifname "wan0" oifname "lan0" ct state {related,established} accept
    }
}
```

La comanda Nmap permet realitzar (mitjançant la sintaxi *nmap -p nºport,... ip.ord* o *nmap -p nºport,... ip.xarxa/mask*) escanejos de ports (bé a un ordinador concret o bé a tots els d'una xarxa, respectivament) de diferents formes. Repassem alguns dels possibles paràmetres que admet, els quals canvien el mètode d'escaneig utilitzat (per defecte es fa servir *-sT*):

*Paràmetre *-sT* : Fa un 3-way handshake complet i just després envia un paquet RST+ACK.
*Paràmetre *-sS* : No completa el 3-way handshake perquè en comptes del darrer ACK envia un RST
*Paràmetre *-sF* : Envia un paquet FIN (sense cap context previ de desconnexió)
*Paràmetre *-sN* : Envia un paquet "deforme" perquè té tots els "flags" a 0 ("null attack").
*Paràmetre *-sX* : Envia un paquet "deforme" perquè té tots els "flags" a 1 ("Xmas attack").

Un sistema Nftables que estigui directament accessible des d'Internet ha d'implementar una sèrie de precaucions per a minimitzar tots aquests escanejos (i els possibles atacs conseqüents). En aquest sentit, sota l'opció *ct state invalid* es recopila la majoria de paquets "deformes" que es poden rebre per tal d'aplicar l'acció desitjada (que serà normalment "log" i/o "counter" i "drop").

5.-a) Assegura't de què el servidor SSH que vas fer servir en la màquina virtual utilitzada al primer exercici estigui de nou iniciat. Tot seguit, executa la comanda *nmap* (des de la mateixa màquina virtual o bé des de la real, com vulguis) utilitzant el mètode *-sT* (o *-sS*, tant se val) i observa el resultat que obtens. Finalment, afegeix en la màquina virtual una regla Nftables que apliqui una acció DROP a tots els paquets SYN (recorda l'opció *tcp flags*) dirigits al seu port 22. Torna a executar la comanda *nmap -sT* ... (o *nmap -sS* ...) i compara el nou resultat obtingut amb el primer.

aII) ¿Què passa si, amb la regla afegida a l'apartat anterior, intentes connectar amb el client *ssh* (des de la mateixa màquina virtual o bé des de la real, com vulguis)? ¿Per què? ¿Què hauries de fer llavors per permetre les connexions SSH només si provenen d'una adreça IP concreta (prova-ho indicant a la regla necessària la IP de la màquina real i tornant-te a intentar connectar; vigila l'ordre de les regles que tinguis definides si mantens encara la de l'apartat anterior).

aIII) Elimina les regles Nftables introduïdes als apartats anteriors. Tot seguit, connecta amb el client *ssh* (des de la mateixa màquina virtual o bé des de la real, com vulguis) al servidor SSH de la màquina virtual. En aquell moment, afegeix la mateixa regla aplicada a l'apartat a). ¿Li passa alguna cosa a la sessió SSH que tenies oberta? ¿Per què? Surt finalment d'ella amb *exit* i torna a intentar iniciar-ne una de nova. ¿Què passa?

b) Elimina la regla Nftables introduïda a l'apartat anterior. Executa la comanda *nmap* utilitzant ara el mètode *-sN* i observa el resultat que obtens. Seguidament afegeix en la màquina virtual de treball una regla Nftables que apliqui les accions DROP i LOG a tots els paquets "de tipus *ct state invalid*" dirigits al seu port 22. Torna a executar la comanda *nmap -sN ...* mentre tens funcionant en un altre terminal la comanda *journalctl -k -f* i, a més de comparar el nou resultat obtingut amb el primer, observa les línies d'informació que van apareixen en el registre.

c) Amb la mateixa regla activada de l'apartat anterior, ¿quin resultat obtens (i quines línies d'informació mostrades al registre veus) si executes la comanda *nmap -sX ...*? ¿Per què?

6.-a) Elimina totes les possibles regles Nftables que tinguis carregades a la màquina virtual de treball. Escriu-hi ara un arxiu de regles com el següent, aplica'l...

```
#!/usr/bin/nft -f
flush ruleset
table ip filter {
        chain input {
                type filter hook input priority filter ; policy drop;
                ct state {established,related} log prefix "DINS " accept
        }
        chain output {
                type filter hook output priority filter ; policy accept;
                log prefix "FORA "
        }
}
```

...i tot seguit observa la sortida de la comanda *journalctl -f -k*. ¿Quan veus la paraula "DINS " o "FORA " als registres mostrats? ¿Concretament, què signifiquen els camps següents de cada registre: IN=, OUT=, MAC=, SRC=, DST=, LEN=, PROTO=, DPT= i SYN (o FIN o ACK...)? ¿Què passaria si substitueixes ara la línia *log prefix "FORA "* de l'exemple anterior per aquesta altra: *tcp dport 443 log prefix "FORA "*?

b) Observa la sortida de la comanda *lsmod | grep nft_log* (o també *lsmod | nf_log_common*). ¿Què dedueixes del que veus? Executa ara *nft flush ruleset* i torna a executa la comanda anterior. ¿Què veus ara?