IPFIX

Monitoritzar el comportament de la nostra xarxa serveix per detectar problemes que degraden el seu rendiment (com ara colls d'ampolla, atacs DDoS, usos indeguts de l'ample de banda, etc). Per realitzar aquest monitoratge es poden emprar dues tècniques: l'anàlisi de paquets i l'anàlisi de fluxes (flows).

L'anàlisi de paquets consisteix en ubicar un esnifador de paquets com ara Wireshark en un punt clau de la xarxa i anar emmagatzemant en disc tot el tràfic que hi travessi per tal de poder realitzar l'anàlisi posterior (o fer ús de tecnologies com el "Port Mirrorring" no sempre disponibles als switchos/routers). No obstant, aquesta tècnica moltes vegades no és possible perquè no sempre es pot instal.lar l'esnifador en el punt que caldria i, sobre tot, perquè demana un equip PC per realitzar la tasca de recollida en necessitar-se força CPU i MOLT d'espai de disc (d'altra banda, si es fa el "Port Mirrorring" el consum de CPU i RAM -i de xarxa!- per part dels switchos/routers també és prohibitiu). Per tant, l'anàlisi de paquets es reserva només per quan interessa realitzar un anàlisi en profunditat de tota la capçalera dels paquets o del seu payload.

L'anàlisi de fluxes, en canvi, només es preocupa de recollir "metadades" sobre el tràfic de xarxa i usar-les per generar estadístiques generals (quantitat de tràfic per protocols, nodes que generen més tràfic, ús de l'ample de banda, etc). Moltes vegades aquestes dades estadístiques són les úniques que voldrem perquè amb elles ja podem saber realment qui i com està fent ús de la nostra xarxa i no ens cal res més (és a dir, amb l'anàlisi de fluxes podem, tal com hem dit, monitoritzar l'ample de banda utilitzat segons protocols, detectar atacs DoS i realitzar anàlisis forenses, detectar fallades de la xarxa i colls d'ampolla que causen congestió i pèrdua de velocitat, implementar un sistema de facturació depenent de l'ús de la xarxa, etc, etc sense que sigui necessari realitzar recollides de paquets sencers).

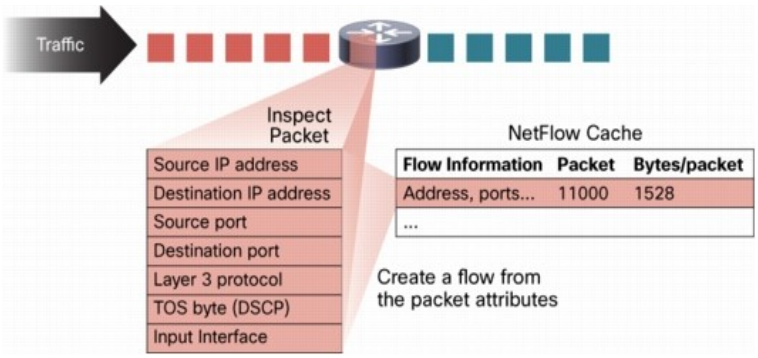
Un fluxe és un conjunt de paquets (observats durant un cert temps) que comparteixen el valor de certes característiques anomenades, en global, "tupla". Tradicionalment, aquest conjunt de característiques han sigut només les següents cinc (les quals moltes vegades ja són suficients, a més del timestamp i d'un comptador de bytes i paquets), però amb el protocol IPFIX es poden definir més (això ho explicarem de seguida):

Mateixa direcció IP d'origen
Mateixa direcció IP de destí
Mateix número de port d'origen
Mateix número de port de destí
Mateix protocol

La idea de tot plegat és que els switchos/routers incorporin una funcionalitat anomenada "flow exporter" mitjançant siguin capaços d'exportar periòdicament (a través d'alguna tarja de xarxa prèviament assignada) aquests fluxos detectats (en forma de paquets UDP o SCTP, generalment) a una determinada màquina-servidor central externa, la qual estarà executant un software específic (el "flow collector"), que els rebrà i emmagatzemarà en disc per tal de poder-ne fer els anàlisis i visualitzacions pertinents amb un altre software especialitzat (el "flow analyzer"). Fixeu-vos que la quantitat d'espai en disc es redueix dràsticament respecte l'anàlisi de paquets, a més de simplificar la infraestructura de la xarxa i no forçar tant l'ús de CPU, la RAM i la pròpia xarxa dels switchos/routers.

Més en concret, el procés de recollida de fluxos és el següent: cada paquet que travessa el switch/router és examinat tenint en compte -tradicionalment- els cinc aspectes de la tupla prèviament definida; el primer paquet únic crearà un fluxe (en forma d'entrada a la catxé del switch/router) i serà reenviat al seu destí com toqui. Els següents paquets que travessin el dispositiu amb els mateixos valors de la tupla inicial seran agregats a aquest fluxe i el comptador de bytes per aquest fluxe (és a dir, per aquesta entrada de catxé) en concret incrementarà. Si algun dels valors dels paquets analitzats no coincideix amb els de la tupla inicial, un altre fluxe serà creat a la catxé. So there can be hundreds of thousands flows recorded in the cache. Obviously, flows do not live in cache forever, instead they are exported from the cache to a flow collector on a regular basis. A flow is exported when it is inactive for a certain time (e. g. no new packets are received for the flow); by default, the inactive flow timer is set to 15 seconds. The flow is also exported when it is long lived (active) and lasts longer than the active timer; by default, the active timer is set to 30 minutes (for instance, a large file download that lasts longer than 30 minutes may be broken into

multiple flows...it is a role of the flow collector to combine these flows showing the total download). Multiple exporters send records periodically to one or more collectors using the UDP or SCTP protocol when the reliable transport is required. The role of a collector, as we have already said, is to gather, record, combine or aggregate the exported flows to produce the reports used for traffic and security analysis.



Hi ha molts protocols que permeten als switchos/routers recopilar i exportar fluxes però cadascun els hi dóna un format diferent als paquets UDP/SCTP resultants i cal mirar amb quin d'aquests protocols és compatible el nostre dispositiu. En podem destacar quatre (encara que n'hi ha força més però de caire privatiu):

*__Netflow__: Privatiu de Cisco. Actualment s'usen dues versions, la v5 (només compatible amb IPv4) i la v9 (més moderna i flexible). És 100% precís però només pot recopilar dades del nivell 3 (L3) cap amunt en la pila OSI.

NOTA: The default netflow implementation was a representation of every IP packet detected (a '1 to 1' relation). Specially in high-traffic environments (for example hosting companies or ISPs) this can become to resource intense, both for storage and processing power. That is when sampled netflow was designed. Sampled netflow is where every one packet out of n packets is processed. The sampling rate is "n". The sampling method can be different. Some implementations select every n packet, others select one random packet in an interval of n packets and some implementations even use other selection methods.

*__sFlow__: Només funciona en switchos/routers que incorporen un determinat chip privatiu. No és 100% precís perquè pren mostres dels paquets cada cert temps (no és en temps real) però pot recopilar dades de qualsevol nivell de la pila OSI (L2-L7) i consumeix menys recursos de CPU/RAM/Xarxa del switch/router en qüestió de Netflow.

*__SNMP__: El protocol més antic de tots, ha sigut reemplaçat pels competidors. Només hi ha un aspecte en el què encara és interessant i én en la recopilació de dades no de xarxa sinó del funcionament intern del switch/router, com ara l'ús de memòria, la càrrega de CPU, la temperatura, etc. No obstant, gràcies a les plantilles de l'IPFIX aquest aspecte en un futur tampoc serà rellevant.

*__IPFIX__: Estàndar oficial de la IETF. Evolució del protocol Netflow (de fet, moltes vegades se l'anomena "Netflow v10"). A l'igual que Netflow v9, aporta la possibilitat de definir plantilles per especificar les dades que es volen introduïr dins dels paquets UDP/SCTP que contenen la informació dels fluxos a exportar. Això fa que es puguin obtenir dades més enllà de la "tupla de cinc valors" clàssica. Però, a diferència de Netflow v9, aquestes plantilles admeten tipus de valors de llargària variable, amb la qual cosa es poden obtenir pràcticament qualsevol tipus de dades, no només els que estiguin predefinits al propi estàndar (que inclouen aspectes per sota de L3 de la pila OSI -MACs, etc- fins el nivell L7 -noms de domini, URLs http , etc- sinó també els que el fabricant pugui definir per ell mateix (com ara el jitter, pèrdua de paquets, RTTs, retransmissions, flags TCP, valors next-hop, etc). Això fa que IPFIX pugui ser una alternativa a SNMP en segons quins models per poder monitoritzar també aspectes internts dels dispositius (com ara la temperatura, ús de CPU, RAM, etc).

NOTA: Templates are sent separately from data flows, and tell the collector what data fields will be in flow packets, how big the fields are, and what order they arrive in. It's impossible to decode a data flow without a template, so if a flow packet arrives before a corresponding template is received, the standard dictates that it should be dropped. Collectors often send out templates at startup, and then at pre-defined intervals. Some manufacturers allow you to change that interval, others don't.

NOTA: Available standard fields to put in the definition of our IPFIX tuple to export are listed here: https://www.iana.org/assignments/ipfix/ipfix.xhtml

It's important to note that Netflow, sFlow, SNMP and IPFIX are NOT Deep Packet Inspection (DPI) - they don't examine the data payloads within packets. IDS, IPS, and Application-layer Firewalls provide the functionality to pick apart the payloads inside a packet and make decisions based on rules and signatures

Exporters

Cada model de switch/router ofereix la seva pròpia metodologia i comandes per tal d'implementar la funcionalitat de "flow exporter". Podeu trobar una llista força exhaustiva de models compatibles amb Netflow/sFlow/IPFIX i dels passos que cal seguir per configurar-los adientment a https://www.plixer.com/support/netflow-ipfix-sflow-configuration-guide

No obstant, a classe no disposem de cap dispositiu "real", així que, per realitzar les pràctiques haurem d'implementar algun "flow exporter" basat en Linux (és a dir, utilitzar algun software que, a partir del trànsit de xarxa capturat en alguna de les seves tarjes concretes, generi els "flows" pertinents en catxé i els acabi reenviant cada cert temps (en forma de paquets UDP/SCTP) a un destí remot -el "flow collector"-, del qual en parlarem més endavant). Afortunadament, existeixen diverses opcions:

*__Pmacct__ (https://github.com/pmacct/pmacct)

NOTA: VyOS fa servir Pmacct per la seva funcionalitat de "flow collector", però ho fa de forma transparent perquè per accedir a dita funcionalitat hem de fer servir les comandes pròpies de VyOS i no pas les pròpies de Pmacct. Concretament, per activar la generació de flows Netflow v9 (els de tipus IPFIX encara no es poden activar) a partir del trànsit detectat que travessa la tarja eth0 i reenviar-los a un "flow collector" executant-se a la màquina remota 1.2.3.4, caldria executar les següents comandes (les indicades començant per # són opcionals):

configure
set system flow-accounting interface eth0
set system flow-accounting netflow version 9
(per defecte si no s'indica res VyOS utilitza la versió 5)
set system flow-accounting netflow server 1.2.3.4 port 9995
(el nº de port 9995 -UDP- és l'estàndard pels collectors Netflow v9; en el cas de ser de tipus IPFIX seria el port nº 4739)
#set system flow-accounting netflow engine-id 100
(identificador de l'exporter; dada que s'enviarà al collector)
#set system flow-accounting netflow sampling-rate 100
(1 means logging every packet; 100 means one packet accounted for every 100)
#set system flow-accounting netflow timeout expiry-interval 60
(indica els segons a esperar sense activitat per tal de considerar un flow llest per enviar al collector)
#set system flow-accounting netflow timeout max-active-life 604800
(indica els segons màxims d'activitat en un flow abans d'enviar-lo al collector igualment -partint així el flow en trossos-)
commit

A partir d'aquí es poden fer servir comandes com ara *show system flow-accounting* o, més específicament, *show flow-accounting interface eth1 [port nº] [top nº]* . També es pot executar *clear flow-accounting counters*

*__YAF__ (https://tools.netsa.cert.org/yaf/docs.html)

*__Nprobe__ (https://www.ntop.org/products/netflow/nprobe , no és lliure)

*__IPT-Netflow__ (https://github.com/aabc/ipt-netflow)

NOTA: A diferència dels altres projectes, que treballen en l'espai d'usuari, "IPT-netflow" és un mòdul del kernel que, després de compilar-lo i carregar-lo, permet utilitzar una cadena Iptables anomenada NETFLOW per on s'hauran de reenviar els paquets que decidim. Una variant compatible amb Nftables és https://github.com/junjunk/ipt-netflow

*__Nflow-generator__ (https://github.com/nerdalert/nflow-generator)

NOTA: Aquest programa en realitat "s'inventa" tràfic de xarxa sota uns determinats patrons i, a partir d'aquest tràfic simular genera flows Netflow v5. Útil per laboratoris de pràctiques ràpids

EXERCICIS:

1.-a) Arrenca una màquina virtual que tingui com a disc dur el disponible al "cloud" del centre amb tota la suite Elastic ja preinstalada. A més, ha de tenir dues tarjes de xarxa en mode "adaptador pont", 5GB de RAM (o més) i el tallafocs desactivat. Instal.la-hi el paquet "pmacct".

> NOTA: Aquest paquet conté bàsicament el dimoni "pmacctd", el qual captura en temps real els paquets que travessen la interfície indicada (utilitzant per això la mateixa llibreria pcap que fa servir Wireshark) i genera les estadístiques pertinents (segons la "tupla" que s'hagi indicat) per tal de mostrar-les directament a pantalla o bé guardar-les a memòria o en una base de dades PostgreSQL/MySQL/SQLite/MongoDB o bé exportar-les en forma de paquets Netflow/IPFIX/sFlow (segons el "plugin" que s'hagi fet servir)

> NOTA: Aquest paquet també conté la comanda client "pmacct", que permet obtenir les estadístiques actuals de memòria. I també el dimoni "nfacctd", que funciona com a "collector" ja que s'encarrega de rebre paquets Netflox/IPFIX i, igualment, pot generar les estadístiques pertinents i mostrar-les a pantalla, guardar-les a memòria o a una base de dades PostgreSQL/MySQL/SQLite/MongoDB o bé replicar-les a altres collectors.

b) Edita l'arxiu "/etc/pmacct/pmacctd.conf" per a què tingui aquest contingut:

! Les línies següents han de tenir els valors indicats quan el tipus de servei Systemd és de tipus "forking" (que és com és)
daemonize: true
pidfile: /var/run/pmacctd.pid
syslog: daemon
! La línia següent indica la tarja de xarxa de la qual es monitoritzarà tot el tràfic
interface: enp0s8
! La línia següent indica un filtre de captura de paquets (la sintaxi és idèntica a la dels de Wireshark)
! pcap_filter: net 127.0.0.0/8
! La línia següent defineix la "tupla"; hem d indica els valors que volem que s'incloguin
! Per saber la llista entera de valors possibles, consultar https://github.com/pmacct/pmacct/blob/master/CONFIG-KEYS
aggregate: src_host, dst_host, src_port, dst_port, proto
! La línia següent indica que s'estarà fent servir pmacctd com a Netflow exporter
plugins: nfprobe
! La línia següent indica el Netflow collector. Aquí has d'escriure la IP de la teva màquina real.
nfprobe_receiver: 192.168.15.X:4739
! La línia següent indica que es farà servir IPFIX
nfprobe_version: 10
! La IP de la tarja de xarxa que es farà servir per enviar els paquets IPFIX a l'exterior. En principi aquesta dada no caldria
! facilitar-la perquè Pmacctd utilitza per defecte la tarja que sigui de la mateixa xarxa que el "nfprobe_receiver" però en aquesta
! pràctica dóna la casualitat que hi ha dues. Si volem no col.lapsar enp0s8, aquí hem de posar la IP de la tarja enp0s3
nfprobe_source_ip: 192.168.15.Y
! Reduïm una mica els timeouts per defecte per no haver d'esperar tant en fer les proves
nfprobe_timeouts: tcp=3:maxlife=6

c) Posa en marxa Pmacct amb *sudo systemctl start pmacctd* (per curiositat recorda que pots veure què fa exactament aquesta comanda executant *systemctl cat pmacctd*)

d) Posa en marxa a la teva màquina real un servidor Netcat que "simuli" ser un "IPFIX collector" (més que res per a què Pmacctd no obtingui errors de tipus ICMP "Port unreachable" en trobar-se el port 4739 tancat). És a dir, executa-hi la comanda *nc -u -l -p 4739*

e) Simula ara un atac DDoS demanant a algun company de la classe que executi la comanda següent (instal.lable amb el paquet "nmap"; tal com ja vam veure al primer trimestre) des de la seva màquina virtual (per a què no hagis d'obrir-ne una altra): *nping -c 100000 --rate 5000 --data-length 1200 --tcp --flags SYN --win 64 -p 22 -S random 192.168.15.Z* (on la IP que s'ha d'indicar és la pertanyent a la tarja enp0s8 de la teva màquina virtual, la monitoritzada). ¿Per a què serveixen cadascun dels paràmetres indicats?

f) ¿Què hi veus a la finestra oberta del Netcat? Obre el Wireshark de la teva màquina real i indica el filtre de pantalla "udp.dstport == 4739". ¿Què veus?

Collectors:

De "flow collectors" (moltes vegades integrats amb "flow analyzers") en tenim uns quants de disponibles a sistemes Linux. Per exemple:

*__Nfdump + Nfsen__ (https://github.com/phaag/nfdump) : Nfdump és en realitat un conjunt d'utilitats de terminal entre les quals es troben el servei "nfcapd" (que és l'encarregat de recollir els fluxos Netflow/IPFIX i de guardar-los en disc de forma rotatòria), i la comanda "nfdump" pròpiament dita (que permet cercar resultats concrets dins de la informació guardada per "nfcapd" fent ús de filtres de tipus BPF (com els de captura de Wireshark). També són interessants les comandes "nfexpire" i "nfreplay". Nfsen, per la seva banda, és la interfície web de "nfdump" (necessita un servidor web compatible amb PHP per poder funcionar).

*__Silk__ (https://tools.netsa.cert.org/silk/docs.html) : Suite completa amb múltiples comandes que permeten recopilar fluxes Netflox/IPFIX, guardar-los en disc de forma rotatòria i manipular-los i analitzar-los de diferents formes.

*__Flowanalyzer__ (https://gitlab.com/thart/flowanalyzer) : Especialment dissenyat per guardar els flows en ElasticSearch

*__Ntop__ (https://www.ntop.org/products/traffic-analysis/ntop) : Solució integral collector+analyzer però no és lliure

*__Argus__ (https://qosient.com/argus) : Una altra solució integral collector+analyzer

De totes formes, nosaltres utilitzarem un altre software que ja coneixem: LogStash. Gràcies a què incorpora un "codec" compatible amb Netflow/IPFIX, el podrem fer servir com un "collector" perfectament funcional i amb tots els avantatges de tenir-lo ja integrat amb la infraestructura Elastic.

EXERCICIS:

2.-a) Modifica la configuració de Pmacctd per a què ara el "flow collector" sigui la IP de la tarja enp0s3 de la teva màquina virtual (se suposa que es on estarà escoltant LogStash) i atura el servei Pmacctd.

b) Crea el fitxer "/etc/logstash/conf.d/ipfix.conf" amb el següent contingut...:

```
input {
   udp {
      host => "192.168.15.Y"  (la IP de la tarja enp0s3)
      port => 4739
      codec => netflow {  versions => [10]   }
   }
}
filter {}
output {
   elasticsearch { hosts => ["localhost:9200"] }
}
```

...i inicia el servidor ElasticSearch, el servidor LogStash i el servidor Pmacct (per aquest ordre). Repeteix "l'atac DoS" realitzat a l'apartat e) de l'exercici anterior i seguidament observa amb la comanda *curl http://localhost:9200/_cat/indices* si apareix a ElasticSearch un nou índex Logstash anomenat com el dia d'avui (corresponent a les dades IPFIX començades a rebre). Un cop comprovat això, atura de nou els servidors Pmacct i Logstash i executa la comanda *curl -X DELETE http://localhost:9200/logstash-**

c) Les dades rebudes de Pmacct es podrien "netejar" una miqueta. Per exemple, IPFIX informa del protocol del "flow" mitjançant el seu codi numèric estàndar (1=ICMP, 6=TCP, 17=UDP, etc) però seria molt més fàcil que aparegués directament el seu nom. Això es pot aconseguir amb el filtre "translate" de Logstash (https://www.elastic.co/guide/en/logstash/current/plugins-filters-translate.html). Així doncs, podem escriure el següent dins de la secció *filter { }* del seu fitxer de configuració:

```
translate {
   field => "[netflow][protocolIdentifier]"
   destination => "[netflow][protocolIdentifierString]"
   override => "true"
   dictionary => [ "6", "TCP", "17", "UDP", "1", "ICMP" ]
}
```

cII) Logstash can do a reverse DNS lookup for us on the source / destination IP addresses. To achieve this, we need to first copy the values of sourceIPv4Address and destinationIPv4Address to new fields to contain the name. This is so that if the reverse DNS fails, the field that should contain the name still contains the address. So inside *filter { ...}* section you should write this (it can be below the translate filter you wrote before) using the mutate filter (https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html) ...

```
mutate {
   add_field => ["sourceName", "%{[netflow][sourceIPv4Address]}"]
   add_field => ["destName", "%{[netflow][destinationIPv4Address]}"]
}
```

...and then, you can simply ask logstash to do a reverse lookup with dns filter (https://www.elastic.co/guide/en/logstash/current/plugins-filters-dns.html) :

```
dns {
   reverse => ["sourceName","destName"]
   action => "replace"
}
```

d) Inicia de nou el servidor Pmacct, LogStash i ara també el servidor Kibana. Repeteix "l'atac DoS" realitzat a l'apartat e) de l'exercici anterior un parell o tres de vegades. Accedeix seguidament des del navegador de la teva màquina real a Kibana (a través de qualsevol de les dues tarjes per on estigui escoltant) i "importa-hi" el nou índex LogStash que s'haurà d'haver generat per tal d'observar les dades dels fluxes "IPFIX" directament des de la pestanya "Discover". Concretament, en aquesta pestanya sel.lecciona només els següents camps per veure els seus valors: "netflow.sourceIPv4Address" , "netflow.destinationIPv4Address", "netflow.destinationTransportPort", "netflow.protocolIdentifierString", "netflow.octetDeltaCount", "netflow.packetDeltaCount" i "netflow.version"

e) Vés a la pestanya "Visualize" i genera els següents gràfics:

*"Pie chart" comptant els fluxos segons el seu protocol (UDP, TCP, etc)
*"Pie chart" comptant els fluxos segons el seu port de destí
*"Pie chart" comptant els fluxos segons la seva IP d'origen

NOTA: Recordeu que tots aquests "pie charts" es fan igual: l'apartat "Metrics" cal deixar-ho com està ("Slice size->Count") i a l'apartat "Bucket" cal sel.leccionar "Split slices->Aggregation->Terms" i llavors sel.leccionar el camp pel qual es dividirà el pastís.

eI) ¿Quina informació obtens si fas un "Pie chart" on a l'apartat "Metrics" sel.lecciones "Slice size->Sum->Field=netflow.octetDeltaCount" i a l'apartat "Bucket" sel.lecciones la IP d'origen com a terme divisor?