NIDS Suricata

Introducció

Un NIDS ("Network Intruder Detection System") és un sistema que intenta identificar els intents d'introduir-se en una xarxa o fer-ne un ús indegut. És a dir: un NIDS supervisa els paquets que passen per la xarxa i detecten patrons sospitosos. Els sistemes NIDS es poden utilitzar contra malware/virus/troians/cucs, etc, així com detectors d'atacs DoS i/o Spoofing / escanejadors de vulnerabilitats i/o ports, etc.

Quan el NIDS detecta una activitat de xarxa sospitosa, pot actuar de forma passiva (simplement alertant a l'usuari administrador mitjançant l'enviament d'un correu electrònic, per exemple), o bé, en configuracions més sofisticades, pot actuar de forma activa (ordenant al tallafocs que reaccioni dinàmicament a aquestes activitats). En aquest darrer cas es diu que s'està fent servir llavors un NIPS ("Network Intruder Prevention System").

En aquest sentit, un NIDS/NIPS és una eina complementària al tallafocs de la xarxa perquè en ser capaç de supervisar en profunditat el trànsit intern, entrant i sortint de la xarxa (inspeccionant tant el seu contingut en busca de "payloads" com monitoritzant el seu patró de comportament) pot identificar el trànsit sospitós o maliciós que hagi pogut passar per alt el tallafoc (com, per exemple, l'aparició a la LAN d'una nova màquina no coneguda).

> NOTA: Sovint, al fet d'inspeccionar l'interior dels paquets, tot i que tècnicament es pot fer de moltes formes diferents, genèricament se l'anomena **DPI** ("Deep Paquet Inspection"). Així que els NIDS/NIPS en realitat podríem dir que és software que aplica tècniques DPI per aconseguir, en el seu cas, un objectiu ben concret, que és el de detectar paquets maliciosos. Altres elements software o hardware com els tallafocs o els esnifadors de xarxa també poden implementar tècniques DPI per altres motivacions.

Molts NIDS treballen buscant coincidències de signatures (és a dir, seqüències de bytes) conegudes de programari maliciós dins del contingut dels paquets. Si és aquest el cas, els usuaris poden personalitzar conjunts de regles i adquirir signatures (de pagament o comunitàries) per detectar les darreres amenaces de programari maliciós. Per tant, en aquest aspecte, una solució NIDS només és tan bona com les regles disponibles que apliqui al trànsit supervisat: cal ser conscient d'això.

En resum, la tasca de NIDS és escoltar passivament el trànsit de la nostra xarxa (descodificant paquets i tornant-los a muntar en temps real si cal) i buscar signatures/patrons al flux de dades que puguin indicar una amenaça de seguretat per a la xarxa. Per aconseguir-ho, utilitza un llenguatge basat en regles que descriu el trànsit que hauria de eliminar o passar. Els usuaris poden escriure i modificar les seves pròpies regles o bé aplicar conjunts de regles proporcionats per la comunitat; en qualsevol cas, aquestes regles permeten identificar gairebé qualsevol tipus de trànsit que travessa la xarxa i realitzar alguna acció en conseqüència. Com ja hem dit, aquestes accions poden variar entre respostes "passives" (registrar-les al log del sistema, enviar un correu electrònic) a respostes "actives" (fer alguna cosa per evitar que es produeixi l'activitat maliciosa).

> NOTA: La recerca d'atacs no és l'únic cas d'ús d'IDS, també es poden utilitzar per trobar infraccions de la política de xarxa d'una empresa. Per exemple, un IDS podrà indicar que un empleat xatejava o mirava vídeos en lloc de treballar.

Tipus de NIDS

En realitat, l'explicació de l'apartat anterior sobre NIDS només està relacionada amb un tipus específic de NIDS: els NIDS "basats en signatures". En un NIDS "basat en signatures", com acabem de dir, hi ha regles o patrons coneguts de trànsit maliciós (contenint malware o generat per aquest, provocat per escanejos de diversos tipus, etc) que es busca; quan es troba una coincidència amb una signatura, genera una alerta. És el mètode de detecció més senzill perquè només es compara (mitjançant expressions regulars, normalment) el que s'està analitzant amb una llista determinada de signatures. Exemples de signatures poden ser: "una sol·licitud HTTP de '/etc/passwd' adreçada a un servidor web de Linux", "un correu electrònic amb un fitxer adjunt anomenat morw.exe, que és una forma coneguda de programari maliciós", etc. En tot cas, cal que el conjunt de signatures reconegudes sigui adient per limitar el nombre de falsos positius.

De totes formes, hi ha un altre tipus de NIDS: els NIDS "basats en anomalies". Amb aquest altre tipus de NIDS, el "payload" del trànsit és molt menys important que l'activitat que el va generar perquè un NIDS basat en anomalies el que fa és buscar activitat inusual que es desviï de les mitjanes estadístiques d'activitats anteriors (o d'activitats que no s'havien vist anteriorment). Dit d'una altra manera, els NIDS basats en l'anomalia comparen el que es podrien considerar perfils de comportament 'normals' amb els esdeveniments observats a la realitat per veure si pateixen desviacions importants. Un perfil pot ser, per exemple, el nombre d'intents fallits d'inici de sessió a un amfitrió o el nombre de correus electrònics enviats per un usuari, etc. L'anàlisi basada en anomalies pot detectar amenaces desconegudes anteriorment (atacs de dia zero), però normalment hi ha un període d'entrenament activat durant el qual es generen els perfils inicials; a partir d'aquí, un cop establerta la línia base, potser arriba el moment en què un servidor envia més activitat HTTP de l'habitual o s'ha vist un nou amfitrió a la LAN, etc: és llavors quan es dispara l'alerta.

Construir perfils precisos és una tasca complexa i no sempre és possible: quan un perfil és massa ajustat al patró d'entrenament, l'activitat legítima pot activar alertes que produeixin falsos positius (emmascarant l'amenaça real, per tant) i si les polítiques són massa fluixes, el programari maliciós podrà ser invisible i no ser detectat. Per tant, definir un comportament legítim és difícil i s'ha de fer per usuari, per ordinador i per entorn. A més, les línies base dels perfils s'han d'actualitzar dinàmicament perquè els perfils estàtics acaben sent inexactes ja que els sistemes i les xarxes canvien amb el pas del temps. Per tot això, la tècnica basada en anomalies poques vegades s'implementa.

En resum: els enfocaments basats en signatures són més ràpids, generen menys falsos positius i no requereixen temps per basar-se que els enfocaments basats en anomalies. Dit això, perquè tenen una naturalesa reactiva, els sistemes totalment dependents de NIDS basats en signatures estan completament exposats a nous atacs: com que la seva efectivitat depèn completament d'una base de dades de signatures d'intrusió preexistents, els NIDS basats en signatures solen ser paralitzats davant d'atacs sofisticats i nous. Per contra, tot i que un NIDS basat en anomalies pot ser difícil de configurar i "entrenar", pot ser bastant eficaç en definir el comportament base esperable d'un sistema, disseccionant-lo per cada pila de protocols.

Software

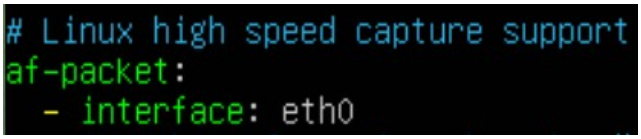
Snort (https://www.snort.org) és un NIDS open-source molt famós però les actualitzacions del seu conjunt de regles són de pagament basat en una subscripció mensual. D'altra banda, Suricata (https://suricata.io; https://github.com/OISF/suricata ; https://suricata.readthedocs.io), és un altre NIDS també open-source que té una política d'actualització de regles molt més oberta. A més, utiliza el mateix motor de regles que Snort, així que la sintaxi general de les seves regles és heredada directament de les de Snort. Suricata és el programa que estudiarem en aquest document.

Zeek (https://www.zeek.org) és el NIDS "basat en l'anàlisi" més famós. De fet, en realitat és un llenguatge específic de domini per a aplicacions de xarxa en què està escrit Zeek. Per tant, és difícil aprendre a comparar amb NID basats en regles. Com que Zeek és especialment eficaç en l'anàlisi del trànsit, s'utilitza sovint en casos forenses i d'ús relacionats també.

Suricata: instal·lació i posada en marxa

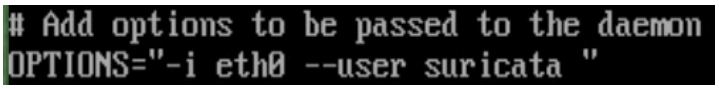
Els passos per instal·lar la darrera versió estable de Suricata i posar-la en marxa són els següents:

*Ubuntu: després d'executar *sudo apt install suricata*, cal editar l'arxiu "**/etc/suricata/suricata.yaml**" (que és el fitxer de configuració del servei Suricata, tal com de seguida estudiarem) i substituir el nom genèric "eth0" que apareix a la directiva "interface" sota la secció "af-packet" (tal com mostra la captura de pantalla) pel nom real que tingui al sistema la tarja de xarxa per on volem que escolti Suricata (per exemple, "enp0s3").



*Fedora: després d'executar *sudo dnf install suricata*, cal editar el mateix arxiu que s'ha descrit al paràgraf anterior ("**/etc/suricata/suricata.yaml**") en referència a sistemes Ubuntu i s'ha de fer de la mateixa manera. Però, a més a més, cal editar també l'arxiu "**/etc/sysconfig/suricata**" (el qual és llegit per l'arxiu "suricata.service", via la directiva *EnvironmentFile=*, quan iniciem el servei) i substituir el nom genèric "eth0" que apareix a com a valor de la línia *OPTIONS=* (tal com mostra la captura de pantalla) també pel nom real que tingui al sistema la tarja de xarxa per on volem que escolti Suricata (per exemple, "enp0s3").

NOTA: L'edició es podria ferdirectament executant aquesta comanda: *sudo sed -i "s|eth0|enp0s3|" /etc/sysconfig/suricata*



A partir d'aquí, caldrà comprovar si el servei *suricata* es troba encès (a Ubuntu intenta posar-se en marxa per defecte) amb *systemctl status suricata* i, si no, comprovar si és capaç d'encendre's amb *sudo systemctl start suricata* No obstant, encara no tindrem definides cap regla per detectar intrusos, així que no ens servirà de gaire tenir el servei Suricata encès si no fem agluna cosa més (seguiu llegint).

Suricata: fitxers de registre

Un cop tinguem el programa Suricata en marxa al nostre sistema, dins de la carpeta "**/var/log/suricata**" trobarem diversos arxius de log que tenen diferents funcionalitats...:

*"**suricata.log**" : emmagatzema els logs del propi programa (inicis, recàrregues, errors...)
*"**stats.log**" : emmagatzema diverses estadístiques sobre el tràfic recollit fins el moment

...però sobre tot ens interessaran aquests dos:

*"**fast.log**" : emmagatzema les alertes disparades per les regles, en format simplificat
*"**eve.json**" : emmagatzema les alertes disparades per les regles i altres events relacionats amb el tràfic de xarxa detectat (identificats segons el seu tipus: "fileinfo", "dns", "http", "stats", "anomaly", "tls", "ssh", "smb", etc -de fet, les alertes no són més que un tipus concret d'event, categoritzat per l'identificador "alert"-) en format JSON. El nom i estructura concreta dels camps JSON guardats es troba documentat a https://suricata.readthedocs.io/en/latest/output/eve/eve-json-format.html

NOTA: Sota la subdirectiva "*types:*" ubicada a la subsecció "*eve-log*" de la secció "*outputs:*" de l'arxiu de configuració "suricata.yaml" es pot triar justament quins tipus d'events la detecció dels quals es vol registrar dins de l'arxiu "eve.json"; aquests tipus són, com ja hem dit, entre d'altres: "alert", "anomaly", "files", "http", "dns", "tls", "smtp", "dhcp", "ssh", "nfs", etc).

NOTA: Tots els arxius de logs anteriors existeixen i funcionen perquè els valors de la configuració definits per defecte en una instal.lació estàndar de Suricata així ho fan possible, però aquests valors es podrien canviar per a què aquests arxius de log no s'anomenessin així, o deixessin de funcionar, o bé crear-ne d'altres tipus (relacionats amb alertes o bé amb simples events), etc.

Suricata: descàrrega i actualització de regles

Abans de posar en marxa definitivament el dimoni Suricata, però, un pas previ imprescindible és omplir per primera vegada l'arxiu de regles a partir de la seva descàrrega més actualitzada d'Internet. Suricata ve amb una comanda anomenada *suricata-update* (https://github.com/OISF/suricata-update) que serveix justament per això (concretament, aquesta comanda per defecte fa la descàrrega de la llista de regles disponible lliurement al lloc web https://rules.emergingthreats.net) i també per, cada cop que l'executem, actualitzar-hi les regles amb les més noves que hi hagin aparegut. Per fer-la servir només ens cal executar-la així: *sudo suricata-update* La comanda anterior omplirà de contingut l'arxiu de regles de Suricata, que per defecte és "**/var/lib/suricata/rules/suricata.rules**"

NOTA: La ruta de la carpeta on es troba el fitxer de regles es pot controlar mitjançant la directiva "*default-rule-path*" de l'arxiu de configuració "/etc/suricata/suricata.yaml", la qual per defecte té, efectivament, el valor "/var/lib/suricata/rules". D'altra banda, el propi nom del fitxer del regles també es pot controlar, en aquest cas mitjançant la directiva "*rule-files*", que apareix just a sota de "*default-rule-path*" i que pren com a valor un array de diferents noms de fitxers (tots ubicats dins de la carpeta indicada a "default-rule-path") que es reconeixeran com a fitxers de regles vàlids (essent el nom "suricata.rules" l'únic nom indicat allà per defecte).

Un cop fet aquest pas, ja podrem posar en marxa el dimoni *suricata* i començarà a detectar les alertes definides a les regles importades. No obstant, per tenir un control més exhaustiu del funcionament del nostre NIDS serà necessari que aprenguem com configurar-lo (a través de les directives adients de l'arxiu "/etc/suricata/suricata.yaml") però independentment també serà imprescindible conèixer com s'escriuen les regles, per tal de poder adaptar les que ja hi són a les nostres necessitats particulars (o bé fer-ne de noves i personalitzades si calgués). El següent apartat explica això darrer.

Suricata: definició de regles

Cada regla (la qual podrà generar eventualment una alerta que s'emmagatzemarà als arxius de log descrits en un apartat anterior) es pot escriure en forma de línia en un fitxer de regles amb el següent format:

acció protocol IPOrigen PortOrigen direcció IPDestí PortDestí (opcio1:valor1;opcio2;...)

on:

acció : normalment serà la paraula "***alert***" (per indicar que Suricata haurà de generar l'alerta si el paquet avaluat té les característiques que coincideixen amb la regla indicada) però també pot ser "***pass***" (per indicar que Suricata ha de deixar d'avaluar el paquet en qüestió si coincideix amb la regla i no seguir avaluant la resta de regles posteriors) o "***drop***"/"***reject***" (per indicar que Suricata, a més de generar l'alerta si el paquet avaluat coincideix amb la regla, ha d'ordenar al tallafocs subjacent que elimini aquest paquet -en un cas sense notificar-ho a l'origen i en l'altre cas notificant-ho via paquet TCP RST o, en el cas d'altres protocols, via missatge ICMP de tipus 3-; no obstant, per a què això funcioni s'ha de configurar convenientment el tallafocs pertinent)
protocol : pot ser la paraula "***tcp***", "***udp***", "***icmp***" o "***all***" (aquest darrer és equivalent a "*ip*" o "*any*"). També pot ser qualsevol de les següents paraules si s'ha activat el protocol de nivell 7 corresponent dins de l'arxiu "suricata.yaml" : "***http***", "***https2***", "***ftp***", "***tls***", "***smb***", "***dns***", "***ssh***", "***smtp***", "***imap***", "***nfs***", "***dhcp***", "***ntp***", "***krb5***", entre d'altres.
IPOrigen i *IPDestí* : poden ser IPs de host o IPs de xarxa (amb notació CIDR). Es poden indicar un conjunt d'IPs (de host i/o de xarxa) si s'escriuen entre claudàtors i separades per comes, així: *[1.1.1.1,1.2.3.4]* Es pot indicar també el símbol "!" davant de la IP (o del conjunt) per indicar un "excepte". Es poden fer servir les variables $HOME_NET o $EXTERNAL_NET com a valors d'IPs vàlids (veieu els exercicis)
PortOrigen i *PortDestí* : es pot indicar un número només o un conjunt de ports si s'escriuen entre claudàtors i separades per comes, així: *[80,443,23]* També es pot indicar un rang entre dos números, així: *nº:nº*. També es pot escriure *:nº* (equivalent a "tots els ports inferiors o igual a nº") o també *nº:* (equivalent a "tots els ports igual o superiors a nº"). Es pot indicar també el símbol "!" davant del nº de port per indicar "excepte".
direcció : normalment serà el símbol "->" però també pot ser "<>" per indicar que les IPs i ports especificats a la regla poden ser indistintament d'origen o de destí independentment del costat del símbol "<>" on s'hagin escrit

Les "opcions" indicades entre parèntesi poden tenir associades un determinat valor (amb la sintaxi *opcio1:valor1*) o bé poden tenir la forma d'una simple paraula (així: *opció2*). La majoria de les opcions serveixen per comparar el contingut del paquet a avaluar en el nivell 7 a la capa OSI amb la informació indicada a la regla (i disparar l'alerta si s'escaiés). Una llista classificada d'alguna de les opcions possibles, (entre moltes d'altres que es poden consultar a https://suricata.readthedocs.io/en/latest/rules/index.html) , és la següent:

* Metadadades de l'alerta

"**msg**" : Opció obligatòria. El seu valor serà el missatge personalitzat de l'alerta.

> NOTA: Els caràcters **;** \ " han de ser escapats si s'escriuen dins del missatge
> NOTA: Molts dels missatges de les alertes predeterminades tenen un conveni on la primera paraula -en majúscules- indica l'entitat/organització que ha definit aquesta alerta (per exemple, ET es correspon a https://rules.emergingthreats.net) i la segona paraula -en majúscules també- indica el tipus d'atac reconegut (per exemple, SCAN significaria un atac d'escaneig de ports)

"**sid**" : Opció obligatòria. El seu valor és un número identificador de la regla; per regles personalitzades aquest número ha de ser igual o major a 10000000 (tal com està documentat a https://doc.emergingthreats.net/bin/view/Main/SidAllocation)

> NOTA: També existeix l'opció "**gid**" , que serveix per indicar un número identificador de grup per agrupar diferents regles (segons criteri de l'administrador) sota un mateix conjunt que tingui el mateix "gid"

"**rev**" : Opció opcional. El seu valor és el nº de revisió -incremental- de l'alerta, per si aquesta s'ha anat modificant al llarg del temps; útil per portar un control dels cops que s'ha anat editant la regla.

"**classtype**" : Opció opcional El seu valor és el nom d'algun "classtype" dels que estiguin prèviament definits dins de l'arxiu "/etc/suricata/classification.config" (o a l'arxiu indicat a la directiva de configuració *classification-file*), com ara, per exemple, "icmp-event". El "classtype" serveix per classificar la regla de forma que als visors de logs aparegui el camp "Classification" i la prioritat associada, facilitant així la seva recerca.

> NOTA: L'arxiu "classification.config" està format per línies amb el següent format: *config classification: nomAEscollirDelClasstype,MissatgeQueSortiràAlLog,nºPrioritat* on el número de prioritat pot valer entre 1 (molt poc important) fins 255 (molt important).

"**reference**" : Opció opcional. El seu valor serveix per apuntar a altres recursos amb més informació sobre l'alerta en qüestió. Aquesta opció pot aparèixer varis cop en una mateixa alerta. El seu valor segueix el patró genèric "tipus,referència" i exemples concrets poden ser: *url,www.info.com* (per apuntar a una URL) o també *cve,CVE-2014-1234* (per apuntar a un informe de vulnerabilitat oficial en la base de dades CVE), entre d'altres. Tots els tipus de referència possibles estan definits a l'arxiu "/etc/suricata/reference.config" i tenen el següent format: *config reference: nom URL*

* Característiques a avaluar de la capçalera IP

"**ttl**" : El seu valor és el valor numèric del camp "ttl" de la capçalera IP del paquet que es vol contrastar. També es pot indicar si es vol buscar un valor major o menor del donat (per exemple, 30) amb la sintaxi *ttl:>30 o ttl:<30* , respectivament.

"**ip_proto**": El seu valor és el nom o bé l'equivalent numèric del protocol de transport del paquet a avaluar. Els valors més comuns són 1 (o "icmp"), 6 (o "tcp") i 17 (o "udp").

"**geoip**": Aquesta opció serveix per comprovar, gràcies a la llibreria GeoIP2 (la ruta de la qual ha d'estar indicada amb l'opció "*geoip-database*" a la configuració de Suricata), la geolocalització de la IP d'origen, de destí o ambdues del paquet. El seu valor segueix el patró genèric "direccio,PAIS" on la "direcció" pot ser "*src*", "*dst*" o "*any*" (per indicar si es vol comprovar la IP d'origen, de destí o qualsevol de les dues , respectivament) i "PAIS" pot ser un codi de país (com ara *ES*, *US*, *RU*, *CN*, etc) o més (si estan separats per comes) que seran els que s'utilitzaran per comparar a veure si concorden amb el país associat a la IP estudiada.

> NOTA: Cal tenir en compte, però, que l'opció "geoip" només funciona si Suricata s'ha compilat integrant la llibreria "GeoIP-devel". Afortunadament, la versió de Suricata instalada a Ubuntu o Fedora seguint els passos indicats en aquest document compleixen aquest requisit (això es pot comprovar observant les característiques mostrades a la sortida de la comanda *suricata --build-info*)

* Característiques a avaluar de la capçalera TCP

"**seq**" : El seu valor és el valor numèric del camp "seq" de la capçalera TCP del paquet que es vol contrastar. També es pot indicar si es vol buscar un valor major o menor del donat (per exemple, 30) amb la sintaxi *seq:>30* o *seq:<30*, respectivament.

"**ack**" : El seu valor és el valor numèric del camp "ack" de la capçalera TCP del paquet que es vol contrastar. També es pot indicar si es vol buscar un valor major o menor del donat (per exemple, 30) amb la sintaxi *ack:>30* o *ack:<30*, respectivament.

"**window**" : El seu valor és el valor numèric del camp "window" de la capçalera TCP del paquet que es vol contrastar. També es pot indicar si es vol buscar un valor major o menor del donat (per exemple, 30) amb la sintaxi *window:>30* o *window:<30*, respectivament.

"**flags**" : Serveix per identificar paquets TCP amb el flag indicat actiu ("S", "A", "R", "F", "P", "U"). Si es vol indicar la presència d'un determinat flag "o més", s'ha d'afegir el símbol "+" (així, "S+")

* Característiques a avaluar de la capçalera ICMP

"**itype**" : El seu valor és el tipus numèric del paquet ICMP que es vol contrastar. També es pot indicar si es vol buscar un valor major o menor del donat (per exemple, 3) amb la sintaxi *itype:>3* o *itype:<3*, respectivament. O bé indicar un rang (per exemple, entre 0 i 3) amb la sintaxi: *itype:0<>3*

"**icode**" : El seu valor és el codi numèric del paquet ICMP que es vol contrastar. També es pot indicar si es vol buscar un valor major o menor del donat (per exemple, 3) amb la sintaxi *icode:>3* o *icode:<3*, respectivament. O bé indicar un rang (per exemple, entre 0 i 3) amb la sintaxi: *icode:0<>3*

* Característiques a avaluar del "payload" d'un paquet UDP/TCP

"**content**" : El seu valor -de tipus cadena i, per tant, escrit entre cometes- indica el contingut que es comprovarà si apareix en qualsevol lloc dins del "payload" del paquet inspeccionat per, si és així, disparar l'alerta. Per defecte la recerca és de tipus case-sensitive (és a dir, es distingeix entre majúscules i minúscules). És possible escriure el símbol "!" per indicar excepcions. Aquesta opció pot aparèixer varis cop en una mateixa alerta.

> NOTA: Es pot introduir valors escrits en hexadecimal -separats per espai a cada byte- si el conjunt apareix entre "|"; això és útil per inspeccionar paquets maliciosos amb malware o similars i, d'altra banda, és obligatori per indicar els símbols **; : | "** perquè a Suricata tenen un significat especial -concretament, la seva representació en hexadecimal és, respectivament: |3B| , |3A| , |7C| i |22|-). Així doncs, per buscar la cadena "http://" dins del contingut d'un paquet HTTP el valor de l'opció "content" seria "http|3A|//"

"**nocase**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que la recerca efectuada per "content" és de tipus case-insensitive (és a dir, no es distingeix entre majúscules i minúscules)

"**pcre**" : Aquesta opció és similar a "content" però permet especificar expressions regulars com a valor a buscar dins del contingut del paquet. Molt útil per trobar números de tarjes de crèdit, direccions de e-mail, etc. No obstant, cal tenir en compte que el seu ús pot ralentitzar el funcionament de Suricata; és per això que se sol combinar l'opció "content" amb aquesta opció, de manera que primer es comprova sempre "content" i, només si no hi ha coincidència, es passa a comprovar "pcre". El valor d'aquesta opció té la forma "*/<regex>/mod*" on <regex> representa l'expressió regular definida (que pot incorporar qualsevol dels símbols ja coneguts: ^, $, . , ?, *, +, {}, etc) i "mod" representa una o més modificadors, essent el més important "i" (per fer que l'expressió regular sigui case-insensitive).

"**depth**": Aquesta opció s'escriu sempre després de l'opció "content". El seu valor és un número sencer que indica la quantitat de bytes del contingut del paquet que s'avaluaran (començant des del principi). Més enllà d'aquest número la regla ja no es tindrà en compte.

"**offset**" : El seu valor és un número sencer que indica quants bytes Suricata "saltarà" des del principi del contingut del paquet avaluat per començar a tenir en compte la regla. Per exemple, "offset:3;" comprova la regla a partir del quart byte del contingut del paquet en endavant. Aquesta opció s'escriu després de l'opció "content" i es pot combinar amb "depth" per indicar un tros concret de "payload".

"**startswith**" : Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que la regla d'activarà només si el valor de "content" es troba específicament al començament del contingut del paquet. Per exemple, "content:"GET|20|";startswith;" seria equivalent a "content:"GET|20|"; depth:4;offset:0;"

"**isdataat**" : El seu valor és un número sencer que indica la posició (en bytes) dins del contingut del paquet a avaluar (contant des del seu començament) on es mirarà si hi ha, efectivament, dades (o dit d'una altra manera, si la llargària del paquet és prou per a què a la posició donada hi aparegui alguna cosa). És possible escriure el símbol "!" per indicar excepcions. Opcionalment, després del seu valor i separat per una coma, es pot escriure la paraula "**relative**" (és a dir, així per exemple: *isdataat:50,relative;*); aquesta paraula indica que el valor no s'ha de comptar des del començament del contingut del paquet sinó a partir del final de la darrera ocurrència del contingut indicat a la directiva *content* que s'indiqui

> NOTA: Una opció similar és "**dsize**", la qual admet la sintaxi "dsize:>nº" per tal de buscar tamanys de paquets majors que l'indicat (molt útil per trobar tamanys anormals, símptoma comú d'atacs de tipus "buffer overflow")

"**distance**" i "**within**" : Opcions interessants si apareix més d'un cop l'opció "content" en una regla. Concretament, "distance" indica a partir de quin byte després d'un "content" anterior es va a buscar el "content" actual (sense cap límit fins arribar al final), mentre que "within" indica el nombre màxim de bytes on es buscarà aquest "content" actual a partir de l'inici marcat per "distance" (serveix, per tant, per posar un límit de recerca). Per exemple, la regla "... *content:"shellbackdoor", offset:23; depth:13; content:"exploit", distance:34; within:7;* ..." dispara l'alerta quan troba un paquet que conté la paraula "shellbackdoor" a partir del seu byte nº23 (i sempre en els 13 següents) i que conté la paraula "exploit" entre els 34 i 41 bytes més enllà de "shellbackdoor".

* Característiques a avaluar del "payload" d'un paquet HTTP

Les següents opcions es podran indicar només si el protocol associat a la regla és HTTP:

"**http_uri**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" forma part de la URI d'una petició HTTP. Si el valor de "content" inclou caràcters en hexadecimal, cal utilitzar llavors l'opció **http_raw_uri** en comptes de *http_uri*

"**http_method**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el mètode d'una petició HTTP (GET, POST, etc)

"**http_request_line**": Aquesta opció no té valor associat i sempre s'escriu abans de l'opció "content". Indica que el valor de l'opció "content" ha de coincidir amb el d'alguna línia d'una petició HTTP (ja sigui la línia de petició, -"GET / HTTP/2", per exemple-, com la d'una capçalera de client qualsevol)

"**http_client_body**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el contingut d'una petició HTTP (útil sobre tot, doncs, en peticions de tipus POST).

"**http_header**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el contingut d'alguna capçalera HTTP qualsevol de petició o de resposta

"**http_host**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició anomenada "Host".

"**http_accept**": Aquesta opció no té valor associat i sempre s'escriu abans de qualsevol altra opció. Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició anomenada "Accept".

"**http_accept_lang**": Aquesta opció no té valor associat i sempre s'escriu abans de qualsevol altra opció. Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició anomenada "Accept-Lang".

"**http_content_type**": Aquesta opció no té valor associat i sempre s'escriu abans de qualsevol altra opció. Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició o -més habitual- de resposta anomenada "Content-Type".

"**http_referer**": Aquesta opció no té valor associat i sempre s'escriu abans de qualsevol altra opció. Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició anomenada "Referer".

"**http_user_agent**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició anomenada "User-Agent".

"**http_cookie**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el contingut concretament de la capçalera HTTP de petició o de resposta anomenada "Cookie" (és a dir, el valor de les "cookies" (re)enviades del client al servidor o del servidor al client, respectivament).

"**http_stat_code**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa un codi numèric de resposta HTTP.

"**http_response_line**": Aquesta opció no té valor associat i sempre s'escriu abans de qualsevol altra opció. Indica que el valor de l'opció "content" ha de coincidir amb el d'alguna línia d'una resposta HTTP (ja sigui la línia que conté el codi numèric, -"200 OK", per exemple-, com la d'una capçalera de servidor qualsevol)

"**http_server_body**": Aquesta opció no té valor associat i sempre s'escriu després de l'opció "content". Indica que el valor de l'opció "content" representa el contingut d'una resposta HTTP (útil sobre tot, doncs, per fer "scrapping" del codi HTML/CSS/Javascript de pàgines web)

"**urilen**": El valor d'aquesta opció és un número que representa la longitud (en bytes) de la URI en una petició HTTP. També admet la sintaxi "urilen:>nº", "urilen:<nº" o també "urilen:x<>y" (per indicar el rang "més gran que x però més petit que y")

* Característiques a avaluar del "payload" d'un paquet DNS

Les següents opcions es podran indicar només si el protocol associat a la regla és DNS

"**dns.query**" : Aquesta opció no té valor associat i s'ha d'escriure abans de l'opció "content" per tal d'indicar que el valor d'aquesta darrera opció es correspon a (part d')un domini demanat en una consulta DNS

* Característiques relacionades amb l'obtenció de fitxers

Les següents opcions es podran indicar només si el protocol associat a la regla és HTTP, NFS o SMB. Serveixen per guardar els possibles fitxers transferits en el trànsit detectat d'aquests protocols.

NOTA: Aquestes opcions només funcionaran si a l'arxiu "suricata.yaml" apareix la línia "enabled:yes" sota la secció "file-store" (versió 2). També és recomanable tenir descomentada la subsecció "types->files" dins de la secció "outputs->eve-log" per tal de què es al fitxer "eve.json" es guardi un registre de totes les descàrregues realitzades.

"**filestore**" : Aquesta opció, -que només té com a valor/s associat/s certs modificadors opcionals que no estudiarem- és obligatòria i serveix per indicar que efectivament es vol guardar en disc els fitxers detectats en el tràfic pertanyent a la connexió donada pel protocol, IPOrigen:PortOrigen i IPDestí:PortDestí indicats a la regla en qüestió.

NOTA: Cada fitxer es guarda en una subcarpeta sota la carpeta "**/var/log/suricata/filestore**" el nom de la qual començarà pels dos primers caràcters resultants d'haver calculat el hash SHA256 al seu contingut i el nom amb el què es guardarà el fitxer serà precisament aquest hash SHA256. D'aquesta manera, si es detecta un fitxer amb un contingut idèntic encara que tingui un nom diferent no es tornarà a descarregar (el que sí que es modificarà en aquest cas és la data de darrera modificació, això sí).
NOTA: En el moment de grabar-se un determinat fitxer al disc, dins del fitxer "eve.log" apareixerà un registre anomenat "fileinfo" contenint les metadades del fitxer en qüestió. Es pot fer també que aquestes metadades es guardin en un fitxer JSON a part, juntament amb el propi fitxer guardat, si s'assigna el valor "yes" a l'opció "write-fileinfo" sota la secció "file-store" de l'arxiu de configuració de Suricata.

"**filename**" : Aquesta opció és un filtre per "filestore". El seu valor és una cadena que representa el nom del fitxer que haurà de tenir el fitxer en qüestió per tal de grabar-lo.

"**fileext**" : Aquesta opció és un filtre per "filestore". El seu valor és una cadena que representa l'extensió que haurà de tenir el fitxer en qüestió per tal de grabar-lo.

"**filemagic**" : Aquesta opció és un filtre per "filestore". El seu valor és el número màgic que representa el tipus dels fitxers que es volen grabar. El número màgic d'un fitxer es pot saber a partir de la sortida de la comanda *file*

"**filesize**": Aquesta opció és un filtre per "filestore". El seu valor és un número (en bytes, o bé seguit dels sufixes KB, MB, GB ,etc) que representa el tamany dels fitxers que es volen gravar. També es pot indicar "filesize:>nº", "filesize:<nº" o "filesize:x<>y" (per indicar un rang)

NOTA: Existeixen altres opcions més sofisticades, com ara filemd5, filesha1 o filesha256, etc, que permeten fins i tot comprovar la integritat del contingut del fitxer inspeccionat.
NOTA: Per fer una netejar dels fitxers guardats durant cert temps es pot utilitzar la comanda *suricatactl filestore prune -d /var/log/suricata/filestore -age 5s* (en aquest exemple s'eliminaran els fitxers més antics de 5 segons)

Exemples de regles

*Regla que registra tot el tràfic TCP de la xarxa:
alert tcp any any -> any any (msg:"Sample alert"; sid:1000000;)

*Regla que detecta una signatura maliciosa concreta provinent d'una màquina "x" cap a un servidor Samba:
alert tcp 192.168.1.6 any -> 192.168.1.5 139 (msg: "SMBDie Attempt"; content:"|5c 50 49 50 45|"; sid:1000000;)

*Regla que detecta peticions HTTP amb una URI que inclou la cadena "index.php"
alert http any any -> any any (msg: "Access to index.php"; content:"index.php"; http_uri; sid:1000000;)

*Regla que detecta peticions HTTP diferents de HEAD i TRACE al port 80 de qualsevol màquina:
alert http any any -> any 80 (msg:"Bad method"; content:!"HEAD"; http_method; content:!"TRACE"; http_method; sid:1000000;)

*Regla que detecta respostes HTTP de tipus 403 ("Prohibit"):
alert http any any -> any any (msg:"Prohibited attempt"; content:"403"; http_stat_code; sid:1000000;)

*Regla que detecta peticions DNS detectades a la xarxa que inclouen la paraula "google":
alert dns any any -> any any (msg:"Test dns_query option"; dns.query; content:"google"; nocase; sid:1000000;)

*Regla que detecta (i guarda) fitxers amb extensió pdf transferint-se per la xarxa via HTTP:
alert http any any -> any any (msg:"FILE PDF file claimed"; fileext:"pdf"; filestore; sid:1000000;)

*Regla que detecta (i guarda) fitxers de tipus PDF transferint-se per la xarxa via HTTP:
alert http any any -> any any (msg:"FILE pdf detected"; filemagic:"PDF document"; filestore; sid:10000000;)

EXERCICIS:

0.-a) Instal.la Suricata a una màquina virtual qualsevol que tingui la seva tarja de xarxa en mode "adaptador pont" seguint els passos detallats a la teoria. No inicïis encara el servei.

NOTA: En realitat, els NIDS es solen instal.lar a màquines que fan de "gateway" -és a dir, amb dues tarjes i l'IP-forward + NAT activat-, però per simplificar de moment això ho obviarem

b) Modifica l'arxiu de configuració de Suricata ("/etc/suricata/suricata.yaml") de la següent manera:

-Estableix el valor de la variable HOME_NET a la IP de la tarja *enp0s3* de la teva màquina virtual (hauràs d'especificar una màscara de "**/32**", ja que estem indicant la IP d'una màquina concreta i no pas la d'una xarxa a protegir -llegeix la nota següent-). Comprova, a més, que el valor de la variable EXTERNAL_NET sigui el contrari de HOME_NET (per defecte ja deu ser així).

NOTA: En realitat, normalment, com a valor de la variable HOME_NET s'assigna la IP d'una xarxa sencera (per exemple 192.168.15.0/24), que representa la xarxa interna a la qual Suricata pretén tota ella protegir d'amenaces de l'exterior. En aquest sentit, indicar que en el cas d'estar connectats a xarxes que tinguin una màscara "estranya" (per exemple /20 o /12; normalment són de tipus Wifi), per conèixer la IP de xarxa corresponent a indicar a HOME_NET, recomano executar la comanda *ipcalc la.teva.ip/mask* En tot cas, en aquest exercici ho simplificarem i només protegirem la pròpia màquina Suricata, d'aquí que fem servir sempre la màscara "/32", la qual indica sempre que la IP a la què acompanya representa una única màquina individual
NOTA: Existeixen més variables en aquest arxiu que es poden utilitzar a qualsevol lloc del mateix, com ara HTTP_PORTS, SSH_PORTS o també HTTP_SERVERS, SMTP_SERVERS, etc. Es recomana el seu estudi

-Estableix el valor de la directiva "*interface*" sota la secció "*af-packet:*" al nom de la tarja de xarxa que Suricata "monitoritzarà" (que estem suposant que és *enp0s3*). Atenció: si estàs en sistemes Fedora, caldrà que modifiquis també l'arxiu "/etc/sysconfig/suricata" de la forma adient, tal com s'explica a la teoria.

NOTA: Observa l'existència de la directiva "copy-iface" sota la secció "af-packet". Llegeix els comentaris sobre ella per saber per a què serveix
NOTA: Observa també l'existència d'una altra secció similar a "af-packet" però anomenada "pcap". ¿Quina creus que és la principal diferència entre ambdues? ¿I amb la secció "pcap-file"?

-Guarda els canvis anteriors però no arrenquis encara el dimoni Suricata

c) Executa la comanda *sudo suricata-update* ¿Què fa aquest programa? ¿Tindria sentit realitzar una tasca programada executant aquest programa, per exemple, cada dia?

1.-a) Inicia el servei Suricata. Tot seguit, executa la comanda *sudo tail -f /var/log/suricata/fast.log* i, mentre aquesta comanda està en marxa, en un terminal de la màquina real executa la comanda *ping x.x.x.x* (on "x.x.x.x" representa la IP de la màquina on s'està executant Suricata). ¿Què apareix al 1r terminal? ¿Per què?

NOTA: La raó d'haver de connectar-se a la IP de la màquina on s'està executant Suricata ve del fet de què Suricata ha de poder "veure" el tràfic a inspeccionar. Per tant, estem en el mateix cas que amb el Wireshark, on les solucions a aquesta limitació eren vàries: o instal.lar Suricata en una porta d'enllaç, o utilitzar "hubs" en comptes de "switchos" a la nostra LAN, o fer atacs MitM, o utilitzar "network taps", etc

aII) Mentre mantens el "ping" iniciat a l'apartat anterior, modifica ara la comanda tail anterior per a què ara sigui aquesta: *sudo tail -n0 -f /var/log/suricata/fast.log | while read linia; do notify-send "Ei" "$linia"; done* ¿Què passa? Tingues en compte que aquest apartat només funcionarà si estàs en un sistema amb entorn gràfic

b) Executa a la màquina virtual *grep "GPL ICMP_INFO PING *NIX" /var/lib/suricata/rules/suricata.rules* i digues quin és el significat del valor de l'opció "itype" que hi apareix a la regla mostrada, tenint en compte que es tracta d'una alerta associada a tràfic ICMP provinent d'$EXTERNAL_NET i dirigit a $HOME_NET. ¿I el significat del valor de l'opció "classtype"? ¿I de l'opció "metadata"?

> NOTA: L'opció "content" (i "depth", que l'acompanya) podrien ser eliminades de la regla anterior i l'alerta es continuaria disparant igualment. El seu valor només serveix per identificar els "pings" originats des de màquines de tipus Unix, ja que cada sistema operatiu omple el camp de dades del paquet ICMP "echo-request" de formes diferents i, tal com es podria observar mitjançant Wireshark, els paquets d'aquest tipus generats concretament per sistemes UNIX inclouent els bytes indicats a la regla (10 11 12 13 ...). Teniu més informació a https://book.huihoo.com/iptables-tutorial/x1078.htm

c) ¿Com hauria de ser una regla que servís no per detectar els "echo-request" provinents de fora sinó els "echo-reply" sortints de dins del sistema?

2.-a) Crea un arxiu anomenat "test.rules" dins de la carpeta de regles de Suricata amb el següent contingut...:

> *alert tcp any any -> $HOME_NET 23 (msg:"Intent connexió TELNET"; sid:1000003; rev:1;)*

...i afegeix, sota la secció "rule-files" de l'arxiu "/etc/suricata/suricata.yaml" la línia:

> *- test.rules*

> NOTA: Per comentar una regla i així fer que Suricata no la tingui en compte, cal precedir-la del símbol "#" (i reiniciar)

b) Reinicia el servei Suricata i tot seguit, executa la comanda *tail -f /var/log/suricata/fast.log* Ara, en un 2n terminal de la màquina on s'està executant Suricata, escriu la comanda *sudo nc -l -p 23* (recorda d'aturar primer el tallafocs, si n'hi ha) i tot seguit en un terminal de la màquina real executa la comanda *nc x.x.x.x 23* (on "x.x.x.x" representa la IP de la màquina on s'està executant Suricata) ¿Què veus ara al 1r terminal? ¿Per què? Si tanques la connexió des del client i la tornes a establir, ¿què tornes a veure al 1r terminal?

3.-a) Instal.la a la màquina on s'està executant Suricata el servidor HTTP Apache2 (paquet "apache2" a Ubuntu, "httpd" a Fedora) i posa'l en marxa amb *systemctl*. Afegeix la següent regla dins del mateix arxiu "test.rules" i reinicia el servei Suricata:

alert http any any -> $HOME_NET 80 (msg:"URI contains 'webshell.php'"; content:"webshell.php"; http_uri; sid:1000004; rev:1;)

¿Quin tipus d'"event" creus que detecta aquesta regla? Obre un navegador (o usa *curl*) de la màquina real i escriu alguna URL que sigui adient per tal de generar l'alerta (observa la sortida en temps real de *tail -f /var/log/suricata/fast.log* per comprovar-ho)

> NOTA: Cal tenir en compte que l'opció "http_uri" normalitza urls mal escrites (és a dir, elimina caràcters "estranys" com per exemple "../" o codifica els caràcters %,$,?, etc al format URL, etc). Això vol dir que si volguéssim trobar explícitament aquest tipus de caràcters no els podríem trobar amb "http_uri" perquè ja hauran desaparegut. La solució llavors seria utilitzar l'opció "http_raw_uri", que tracta la url "a pèl" sense fer cap normalització prèvia

b) Busca a l'arxiu de regles predefinit del Suricata la paraula "BlackSun". Hauries de trobar una regla més o menys similar a la següent (si no hi és, escriu-la a l'arxiu "test-rules" i reinicia Suricata per provar-la):

> *alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET USER_AGENTS Suspicious User Agent (BlackSun)";* ***content:"BlackSun"; nocase; http_user_agent; depth:8;*** *reference:url,www.bitdefender.com/VIRUS-1000328-en--Trojan.Pws.Wow.NCY.html; reference:url,doc.emergingthreats.net/bin/view/Main/2008983; classtype:trojan-activity; sid:2008983;)*

¿Quin tipus d'"event" creus que detecta aquesta regla (fixa't en qui és l'origen i qui el destí, en les opcions en negreta i en la nota de sota)? ¿Què veuràs a la sortida de *tail -f /var/log/suricata/fast.log* si, a la màquina on està funcionant Suricata, executes la comanda *curl -A "BlackSun" www.google.com?* ¿Per què (consulta al manual de curl per a què serveix el seu paràmetre -A)?

> NOTA: De vegades, els autors de programari maliciós utilitzen valors de la capçalera "user-agent" com a tokens d'autenticació: el servidor "command-and-control" no emetrà ordres als equips que en facin sol·licituds tret que el client especifiqui l'"user-agent" correcte a la sessió HTTP. D'aquesta manera els "dolents" eviten connexions indesitjades (d'investigadors, per exemple). Però, d'altra banda, aquestes cadenes "user-agent" poden ser molt bons indicadors de presència de programari maliciós en un sistema.

4.-a) ¿Per a què serveix la regla següent? Afegeix-la, reinicia Suricata i fes el necessari per mostrar el missatge d'alerta corresponent a "/var/log/suricata/fast.log".

> *alert http any any -> $HOME_NET !80 (msg: "HTTP but not port 80"; sid:1000007; rev:1;)*

> NOTA: És possible que aparegui, a més del missatge de la regla anterior, un altre dient "Applayer mismatch protocol both directions" o similar. Estudiarem el per què de l'activació d'aquesta regla en l'exercici 9

b) Digues en una sola frase quin tipus de tràfic detectaria cadascuna de les regles següents (pots provar alguna, si vols, i provocar llavors algun esdeveniment que l'activi):

> *alert http any any -> any any (msg:"Bad method"; **content:!"GET"; http_method;** sid:1000000;)*
> *alert http any any -> any any (msg:"Prohibited attempt"; **content:"403"; http_stat_code;** sid:1000000;)*
> *alert dns any any -> any any (msg:"Test dns_query option"; **dns_query; content:"google"; nocase;** sid:1000000;)*
> *alert tcp any any -> $HOME_NET 139 (msg: "SMBDie Attempt"; **content:"|5c 50 49 50 45|"**; sid:1000000;)*
> *alert ip any any -> any any (msg: "Id check returned root"; **content: "uid=0|28|root|29|"**;sid:1000000;)* *

* Cal provar aquesta regla en tràfic no xifrat (p. ex així: *nc -l -p 5555 -e /bin/bash*); amb una connexió SSH no funcionarà

5.-a) Crea una regla dins del mateix arxiu "test.rules" com la següent...:

> *alert http $HOME_NET 80 -> any any (msg:"FILE store all"; filestore; sid:1000005; rev:1;)*

...i configura Suricata per a què emmagatzemi a la carpeta "/var/log/suricata/filestore" tots els fitxers detectats per la regla anterior (és a dir, tots els fitxers que es descarreguin des del servidor web per part de qualsevol client). Per això l'únic que has de fer és, sota la secció "*file-store:*" de l'arxiu "suricata.yaml", establir la línia "*enabled:*" a "*yes*" i descomentar la línia "*dir: filestore"*). Un cop fet això, reinicia Suricata.

> NOTA: Si la subsecció "types->files" dins de la secció "eve-log" està descomentada (per defecte ha d'estar-ho!) es guardarà un registre de totes les descàrregues al fitxer "eve.json" i no caldrà llavors utilitzar cap altre fitxer de registre a banda (cosa que es faria posant l'opció "write-fileinfo" a "yes").

aII) Crea dins de la carpeta "/var/www/html" del servidor Apache que has de tenir funcionant a la mateixa màquina on es troba en marxa el Suricata un fitxer anomenat "hola.txt" amb el contingut "Hola". Seguidament, obre un navegador (o usa *curl*) a la màquina real i descarrega't aquest fitxer escrivint http://ip.maq.Virtual/hola.txt ¿Què és el que observes en executar a la màquina virtual la comanda *grep hola.txt /var/log/suricata/eve.json*? ¿Quins són els dos "event_type", a més del de l'alerta pròpiament dit, que hi apareixen relacionats amb aquest fitxer "hola.txt" i quina informació proporciona en concret l'event de tipus "fileinfo"?

> NOTA: Aquests dos events "extra" a més del de l'alerta no apareixeran si no està l'alerta activada

aIII) Després d'haver realitzat l'apartat anterior, ¿què obtens dins de la carpeta "/var/log/suricata/filestore"? (pots executar la comanda *tree* sobre ella per veure el seu contingut de forma recursiva més fàcilment). ¿Què hi té a veure amb el que veus la sortida de la comanda *sha256sum /var/www/html/hola.txt* ?

b) Canvia la regla del 1r apartat per aquesta altra (els canvis són en negreta). ¿Quins "events" detectaran ara?

*alert http $HOME_NET 80 -> any any (msg:"FILE store some"; **content:"Alarma"; nocase;** filestore; sid:1000005; rev:**2**;)*

bII) Obre de nou un navegador (o usa *curl*) a la màquina real i descarrega't el mateix fitxer que abans com sempre, així: http://ip.maq.Virtual/hola.txt ¿Veus alguna alerta nova a "/var/log/suricata/fast.log"? ¿Per què?

bIII) Canvia ara el contingut de l'arxiu "hola.txt" per a què sigui la paraula "Alarma" i torna a repetir l'apartat anterior. ¿Veus ara alguna alerta nova? ¿I algun fitxer nou guardat dins de "/var/log/suricata/filestore"?

> NOTA: Pots esborrar tots els fitxers guardats a "/var/log/suricata/filestore" (més antics que 1s) de cop amb la comanda: *sudo suricatactl filestore prune -d /var/log/suricata/filestore --age 1s*
> NOTA: No només podem detectar contingut textual de la manera vista als apartats anteriors. Per exemple, sabent que qualsevol fitxer al seu començament té una capçalera binària coneguda que permet al S.O reconèixer el seu tipus (el que se'n diu "file magic"), podríem fer el mateix que hem fet als apartats anteriors per detectar descàrregues de tipus concrets de fitxers, com per exemple, d'executables de Linux escrivint això: *content:"ELF"; distance:0;*)

c) Afegeix les següents dues regles a l'arxiu "test.rules" i fes el necessari per provocar que el missatge d'alerta pertinent aparegui a l'arxiu "/var/log/suricata/fast.log" (a més de guardar-se el fitxer en qüestió sota "/var/log/suricata/filestore")

> *alert http any any -> any any (msg:"Mistery 1"; **fileext:"pdf"; filestore;** sid:1000000;)*
> *alert http any any -> any any (msg:"Mistery 2"; **filemagic:"PNG"; filestore;** sid:1000000;)*

6.-a) Executa de nou la comanda *sudo suricata-update* ¿Observes algun canvi en els missatges que et mostra en relació a la vegada anterior en que el vas executar?

b) Els diferents llocs d'Internet des d'on *suricata-update* es descarrega les (noves) regles es troben llistats en un arxiu disponible a http://openinfosecfoundation.org/rules/index.yaml. Sabent això, ¿què fa la comanda *sudo suricata-update update-sources* i, a partir d'aquí, la comanda *sudo suricata-update list-sources* ?

c) ¿Què fa la comanda *sudo suricata-update enable-source ptresearch/attackdetection* i, a partir d'aquí, la comanda *sudo suricata-update list-enabled-sources*? (per a que sigui efectiu cal executar tot seguit *sudo suricata-update* de nou)

> NOTA: També existeixen les comandes *suricata-update disable-source .../...* o *suricata-update remove-source ../..*
> NOTA: Els fitxers "/etc/suricata/enable.conf", "/etc/suricata/disable.conf", "/etc/suricata/drop.conf" i "/etc/suricata/modify.conf", si existeixen, inclouen filtres a aplicar a les regles descarregades per tenir-les en compte o no. Per més informació, consulteu https://suricata-update.readthedocs.io/en/latest/update.html#rule-matching

7.-a) Llegeix l'apartat https://suricata.readthedocs.io/en/suricata-6.0.10/rules/flow-keywords.html#flow per saber què vol dir l'opció "**flow**" i el quadre adjunt per saber què vol dir l'opció "**threshold**"

We can reduce the number of logged alerts for noisy rules by limiting the number of times a particular event is logged during a specified time interval. The *threshold* keyword can be used to control the rule's alert frequency. It has 3 modes: "threshold", "limit" and "both". Its syntax is: ***threshold: type <threshold|limit|both>, track <by_src|by_dst>, count <N>, seconds <T>***

*"Threshold" type can be used to set a minimum threshold for a rule before it generates alerts. A threshold setting of N means that on the Nth time the rule matches, an alert is generated. For instance, this signature only generates an alert if we get 10 inbound emails or more from the same origin ("track by_src") in a time period of one minute:

*alert tcp !$HOME_NET any -> $HOME_NET 25 (msg:"ET POLICY Inbound Frequent Emails - Possible Spambot Inbound"; *
*flow:established; content:"mail from|3a|"; nocase; *
***threshold: type threshold, track by_src, count 10, seconds 60; ***
reference:url,doc.emergingthreats.net/2002087; classtype:misc-activity; sid:2002087; rev:10;)

*"Limit" type can be used to make sure you're not getting flooded with alerts. If set to limit N, it alerts at most N times. For instance, this signature only generates at most one alert every 180 seconds:

*alert http $HOME_NET any -> any $HTTP_PORTS (msg:"ET USER_AGENTS Internet Explorer 6 in use - Security Risk"; \
flow:to_server,established; content:"|0d 0a|User-Agent|3a| Mozilla/4.0 (compatible|3b| MSIE 6.0|3b|"; \
threshold: type limit, track by_src, seconds 180, count 1; \
reference:url,doc.emergingthreats.net/2010706; classtype:policy-violation; sid:2010706; rev:7;)*

*"Both" type is a combination of the "threshold" and "limit" types. It applies both thresholding and limiting. For instance, this alert will only generate an alert if within 6 minutes there have been 5 or more "SIP/2.0 401 Unauthorized" responses, and it will alert only once in that 6 minutes.

*alert tcp $HOME_NET 5060 -> $EXTERNAL_NET any (msg:"ET VOIP Multiple Unauthorized SIP Responses TCP"; \
flow:established,from_server; content:"SIP/2.0 401 Unauthorized"; depth:24; \
threshold: type both, track by_src, count 5, seconds 360; \
reference:url,doc.emergingthreats.net/2003194; classtype:attempted-dos; sid:2003194; rev:6;)*

In "**/etc/suricata/threshold.config**" file global thresholds can be defined to be applied to every rule if it hasn't an specific threshold configuration. Its format is documented in https://suricata.readthedocs.io/en/suricata-6.0.10/configuration/global-thresholds.html

NOTA: IPS rule actions (*drop* and *reject*) are applied to each packet (not only the one that meets the threshold condition).

b) Un cop sabut allò demanat a l'apartat anterior, digues per a què serveix la regla següent:

alert tcp any any -> $HOME_NET 80 (msg: "Possible DDoS attack"; flags: S+; flow: stateless; threshold: type both, track by_dst, count 200, seconds 1; sid:1000008; rev:1;)

c) Afegeix la regla anterior, reinicia Suricata i fes el necessari per escriure el missatge d'alerta corresponent en l'arxiu "fast.log" (pista: utilitza la comanda *nping* jugant amb el seus paràmetres *–rate*, *-S rand* i *-c*).

Suricata també pot inspeccionar arxius de captures de xarxa ja gravats en format "pcap"/"pcapng" en comptes de tràfic en temps real. Per fer això, primer hauràs d'aturar el servei (*sudo systemctl stop suricata*) i llavors executar la comanda *suricata* directament des del terminal així:

sudo suricata -l /var/log/suricata -c /etc/suricata/suricata.yaml ***-r fitxer.pcap [filtreCaptura]***

NOTA: Com es pot observar, opcionalment es pot escriure com a darrer paràmetre de la comanda *suricata*, un filtre de captura (amb el mateix format que fèiem servir al Wireshark, és a dir, el format BPF). Aquest filtre serviria per dir a Suricata que ignori completament el tràfic que en concordi (i així alleugerir una mica la seva tasca). De fet, aquest filtre es podria indicar també com a opció del servei Suricata quan inspecciona tràfic en temps real: només cal afegir-lo als valor d'OPTIONS indicat a l'arxiu "/etc/sysconfig/suricata". D'altra banda, també es poden escriure aquest/s filtre/s en un fitxer i indicar-li al Suricata amb el paràmetre *-F /ruta/fitxerFiltres* Teniu més informació sobre aquest tema a https://suricata.readthedocs.io/en/suricata-6.0.4/performance/ignoring-traffic.html
NOTA: Altres paràmetres interessants de la comanda *suricata* són: *-s /ruta/fitxer.rules* (per indicar un fitxer de regles a utilitzar a més de l'indicat a l'arxiu de configuració general), *-S /ruta/fitxer.rules* (per indicar un fitxer de regles a utilitzar en lloc de l'indicat a l'arxiu de configuració general), *-T* (per comprovar que la configuració estigui correcta), *-v* (mode verbós), *-D* (per executar-lo en segon pla), etc. Per saber-ne més consulteu la seva pàgina del manual. Cal dir també que es pot executar Suricata com a comanda de terminal (en lloc de com a servei Systemd com hem vist fins ara) també per fer inspecció de tràfic en temps real (no només per analitzar fitxers de captura ja preexistents); en aquest cas, caldria substituir el paràmetre *-r fitxer.pcap* per *-i enp0s3* (o el nom de la tarja de xarxa que es vulgui utilitzar en aquest moment).
NOTA: Una web molt recomanable on hi ha molts casos pràctics que poden servir per adquirir més experiència a l'hora d'identificar amenaces i atacs a partir d'analitzar (amb Wireshark) fitxers de captura pcap descarregables allà mateix i també (amb Suricata) fitxers d'alertes, és http://www.malware-traffic-analysis.net/training-exercises.html Altres llocs que ofereixen fitxers "pcap" (amb contingut maliciós o no) per estudiar-los són: https://docs.securityonion.net/en/2.3/pcaps.html https://www.stratosphereips.org/datasets-overview, https://wiki.wireshark.org/SampleCaptures, http://mawi.wide.ad.jp/mawi/samplepoint-F/2012 , https://www.netresec.com/?page=PcapFiles o https://www.netresec.com/?page=MACCDC entre altres.

8.-a) Descarrega't a la màquina virtual on tens Suricata instal.lat l'arxiu "Captures.zip" que hi ha a la web del centre i descomprimeix-lo. Allà tens diverses captures de xarxa que farem servir en aquest exercici. Per començar, confirma amb la comanda següent (hauràs d'instal.lar-te el paquet "tshark" a Ubuntu o "wireshark-cli" a Fedora si no el tens ja) que el fitxer de captura "with_png.pcap" contingui algun paquet que representi alguna descàrrega (via HTTP) d'imatges PNG fent servir un filtre que busca el "file magic" corresponent: *tshark -Y "http contains 89:50:4e:47" -r with_png.pcap* ¿Quantes fotos hi ha dins de l'arxiu de captura?

b) A continuació, afegeix a l'arxiu "test.rules" la següent regla, atura el servei Suricata i executa la *sudo suricata -l /var/log/suricata -c /etc/suricata/suricata.yaml -r with_png.pcap* ¿Què veus al final de l'arxiu "/var/log/suricata/fast.log"?

> *alert tcp any any -> any any (msg:"PNG Detected"; content:"|89 50 4E 47|"; sid:1000009;)*
>
> NOTA: Podries repetir els anteriors apartats buscant per altres "file magics" en altres fitxers de captura que hi ha dins del "zip". Per exemple, el "file magic" per trobar arxius GIF és "**GIF89a**"; per trobar arxius PDF és "**%PDF**"; per trobar arxius ZIP és "|**50 4B 03 04**|"; per trobar arxius JPEG és "|**FF D8**|"; per trobar arxius MP3 és "|**49 44 33**|"; per trobar arxius RAR és "|**52 61 72 21 1A 07 00**|"; per trobar arxius GZIP és "|**1F 8B 08**|"; etc. Teniu una llista més completa a https://en.wikipedia.org/wiki/List_of_file_signatures

c) Ara afegeix a l'arxiu "test.rules" la següent regla i executa la *sudo suricata -l /var/log/suricata -c /etc/suricata/suricata.yaml -r newtrace.pcap* ¿Què veus al final de l'arxiu "/var/log/suricata/fast.log"?

> *alert tcp any any -> any any (msg:"Full XMAS Scan"; flags: SRAFPU;sid:1000010;)*

cII) ¿I si proves d'afegir aquesta altra regla i tornar a executar la mateixa comanda de l'apartat anterior?

> *alert icmp any any -> any any (msg: "ICMP Destination Unreachable"; itype: 3; sid:1000011;)*
>
> NOTA: El fitxer "newtrace.pcap" ha sigut generat per l'Nmap. Precisament, una bona manera de testejar el funcionament de Suricata és realitzar atacs amb eines ofensives, tal com l'Nmap però també d'altres com ara WFuzz, Metasploit (per exemple amb els seus mòduls "auxiliary/dos/http"), etc