NIDS (Suricata)

A NIDS ("Network Intruder Detection System") is a system that tries to identify attempts to break into a network or to misuse it. NIDS monitor packets passing over the network and detect suspicious patterns. NIDS systems can be used against malware/virus/trojans/worms as well as DoS/vulnerability-scanner/port-scanner attacks, etc. When suspicious network activities are detected, a NIDS can simply alert users (by sending an email, for instance) but it can even order firewalls to react to these activities.

Many NIDS work through the use of malware signatures. If so, users can customize rule-sets as well as acquire (paid or community) signatures to detect the latest malware threats. So, in this aspect, a NIDS solution is only as good as the available rules it can apply to the monitored traffic. Besides, they can monitor inbound and outbound traffic and identify suspicious or malicious traffic, which may have somehow bypassed your firewall, or it could possibly be originating from inside your network as well.

In summary, NIDS's job is to passively listen to your TCP/IP network traffic (decoding packets and reassembling streams if necessary) and look for signatures/patterns in the data flow that might indicate a security threat to your network. For accomplish this, it uses a rule-based language that describes traffic that it should collect or pass. Users can take advantage of applying new or existing rule-sets provided by the community as well as writing and modifying their own rules. Complex rules can be written to identify just about any type of traffic going across the network and perform some action. As we have already said, these actions can vary between "passive" responses (just logging it or sending an email) to "active" responses (doing something to stop the malicious activity from happening).

> NOTA: Looking for attacks isn't the only use case for IDS, you can also use it to find violations of network policy. IDS will tell you an employee was using Gtalk, uploading to Box, or spending all their time watching Hulu instead of working.

Snort (https://www.snort.org) is the most famous open-source NIDS but its complete ruleset updates are paid by monthly subscription. On the other hand, Suricata (https://suricata-ids.org ; https://suricata.readthedocs.io) is open-source too, it has a wider open ruleset and, moreover, it uses the same rule engine than Snort, so most of the Snort's rule syntax can be directly applied to Suricata's ones.

> NOTA: Tant Snort com Suricata són programes de terminal. Existeix no obstant un front-end gràfic d'Snort anomenat Sguil (https://github.com/bammv/sguil) que permet gestionar les regles i les alertes d'una forma més amigable

Actually, above explanation about NIDS is related to a specific type of NIDS only: the "signature-based" NIDSs. In a "signature-based" NIDS, as we already know, there are rules or patterns of known malicious traffic that it is looking for. Once a match to a signature is found it generates an alert. These alerts can turn up issues such as malware, scanning activity, attacks against servers and much more. It is the simplest detection method: it just compares what it is analyzing to a given list of signatures using string comparison regular expresions. For example: "an HTTP request for '/etc/passwd' addressed to a Linux web server", "an e-mail with an attached file named morw.exe, which is a known form of malware", etc. When a good set of signatures is available there will be a limited amount of false positives and looking for specific pattern on the payload should allow signatures to be ported from one system to another.

There´s another type of NIDS, however: the "anomaly-based" NIDSs. With "anomaly-based NIDSs", the payload of the traffic is far less important than the activity that generated it. An anomaly-based NIDS tool relies on baselines rather than signatures. It will look for unusual activity that deviates from statistical averages of previous activities or activity that has been previously unseen. In other words, anomaly-based NIDSs match what could be considered 'normal' behaviour profiles against the observed events targeting significant deviations. A profile could be for example the number of failed attempts to login to a host, or the number of emails sent by a user, etc; perhaps a server is sending out more HTTP activity than usual or a new host has been seen inside your LAN, etc. Anomaly-based analysis may be able to detect previously unknown threats (zero-day attacks) but there is usually a training period on which the initial profiles are generated. In fact, profiles should be dynamically updated because static profiles eventually become inaccurate since systems and networks change over time.

Building accurate profiles is a complex task and not always possible. When a profile is too tight to the training pattern, legit activity will trigger alerts producing many false positives (masking the real thread , thus). If policies are too loose, stealthy malware could potentially shape its traffic to go undetected. So defining legit behaviour is hard, and must be done per user, per PC, per enviroment. As a result, this technique is rarely implemented.

Zeek (https://www.zeek.org) is the most famous "analysis-based" NIDS. In fact, it is actually a domain-specific language for networking applications in which Zeek is written. So it is difficult to learn comparing with rule-based NIDs. As Zeek is especially effective at traffic analysis, it is often used in forensics and related use cases too.

In summary: signature-based approaches are faster, generate less false positives, and do not require time for baselining that anomaly-based approaches do. That said, because they are reactive in nature, systems entirely dependent on signature-based NIDS are completely exposed to new attacks. As their effectiveness is entirely dependent on a database of pre-existing intrusion signatures, signature-based NIDS are often crippled in the face of novel, sophisticated attacks. An anomaly-based NIDS can be difficult to set up, configure, and “train”, but can be quite effective in its ability to baseline a system at each protocol stack, and to scale accordingly.

EXERCICIS:

1.-a) Instal.la Suricata 5.x a una màquina virtual qualsevol que tingui la seva tarja de xarxa en mode "adaptador pont" (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). Concretament:

A Ubuntu
sudo apt install software-properties-common
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt update && sudo apt install suricata

A Fedora
sudo dnf install make gcc pcre-devel libyaml-devel jansson-devel libpcap-devel python3-pyyaml libcap-ng-devel nspr-devel nss-devel file-devel lz4-devel zlib-devel rustc cargo libnetfilter_queue-devel
wget https://www.openinfosecfoundation.org/download/suricata-5.0.0.tar.gz
tar -xzvf suricata-5.0.0.tar.gz
cd suricata-5.0.0
./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc –localstatedir=/var
make
sudo make install-full
sed -i "s|#EnvironmentFile=-/etc/sysconfig/suricata| EnvironmentFile=-/etc/sysconfig/suricata|" etc/suricata.service
sed -i "s|/sbin/suricata|/usr/bin/suricata|" etc/suricata.service
sudo cp etc/suricata.service /etc/systemd/system
sudo echo "OPTIONS='-i enp0s3'" | sudo tee /etc/sysconfig/suricata
sudo echo "/usr/local/lib" | sudo tee -a /etc/ld.so.conf && sudo ldconfig

> NOTA: En aquesta darrera comanda *caldrà substituir enp0s3 pel nom de la tarja de xarxa per on escoltarà Suricata*

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 teva xarxa local (192.168.15.0/24) i comprova que el valor de la variable EXTERNAL_NET sigui el contrari de HOME_NET

> NOTA: Existeixen més variables en aquest arxiu que es poden utilitzar a qualsevol lloc del mateix, com ara HTTP_PORTS, SSH_PORTS, etc. Es recomana el seu estudi

> NOTA: En el cas de què estigueu connectats a xarxes (normalment de tipus Wifi) que tinguin una màscara "estranya" (per exemple /20 o /12), per conèixer la IP de xarxa corresponent (que és el que heu d'indicar a HOME_NET), recomano que executeu la comanda *ipcalc la.vostra.ip/mask*

-Estableix el valor de la directiva "interface" sota la secció "af-packet" al nom de la tarja de xarxa que Suricata "monitoritzarà"

> NOTA: Observa l'existència de la directiva "copy-iface" sota la secció "af-packet". ¿Per a què creus que serveix? Observa també l'exitència 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"?

A la carpeta "/var/log/suricata" hi ha diversos arxius de log amb diferents funcionalitats...:

*"suricata.log" : emmagatzema els logs del propi programa (inicis, recàrregues, errors...)
*"stats.log" : emmagatzema 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 informació sobre altres paquets detectats (els quals es poden escollir sota la subdirectiva "types:" que apareix a la secció "eve-log" de l'arxiu "suricata.yaml", i que, entre altres, són "alert", "anomaly", "http", "dns", "tls", "smtp", "dhcp", "ssh", "nfs", etc) en format JSON

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.

Abans de posar en marxa Suricata, però, un pas previ imprescindible és aprendre com són i com s'escriuen les regles que generen les alertes que s'emmagatzemaran als arxius log anteriors. Hi ha moltes regles que ja vénen predefinides en una instal.lació estàndar del Suricata, les quals es troben dins del fitxer "/var/lib/suricata/rules/suricata.rules" (la ruta de la carpeta on es troba aquest fitxer de regles es pot controlar mitjançant la directiva "default-rule-path", la qual per defecte té, efectivament, el valor "/var/lib/suricata/rules"; i el propi nom del fitxer del regles es podria controlar mitjançant la directiva "rule-files", que apareix just a sota de "default-rule-path", la qual 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). Cada regla es pot escriure en forma de línia en aquests fitxers amb el següent format:

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

acció : normalment serà la paraula "alert" (on Suricata genera l'alerta si el paquet avaluat té les característiques que coincideixen amb la regla indicada) però també pot ser "pass" (on Suricata deixa d'avaluar el paquet en qüestió si coincideix amb la regla i no fa res més) o "drop"/"reject" (on Suricata, a més de generar l'alerta si el paquet avaluat coincideix amb la regla, ordena 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 -ho veurem més endavant-)
protocol : pot ser la paraula "tcp", "udp", "icmp" o "all". També pot ser qualsevol de les següents paraules si s'ha activat convenientment el protocol associat a l'arxiu "suricata.yaml" : "http", "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
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 número de port per indicar un "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 d'alguna de les opcions possibles, entre moltes d'altres que es poden consultar a https://suricata.readthedocs.io/en/suricata-5.0.0/rules/index.html, és la següent (les úniques opcions obligatòries són "msg" i "sid"):

Metadadades de l'alerta

"**msg**" : El seu valor serà el missatge personalitzat de l'alerta. Normalment sol ser la primera opció que s'indica dins del parèntesi. Els següents caràcters han de ser escapats si s'escriuen dins d'un missatge: ; \ "

NOTA:Molts dels missatges de les alertes predeterminades tenen un conveni on la primera paraula -en majúscules- indica l'entitat/organitzaió 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**" : El seu valor és un número identificador de la regla; per regles personalitzades aquest número ha de ser igual o major a 10000000. Normalment sol ser la darrera opció que s'indica dins del parèntesi, o penúltima si també s'indica l'opció "rev".

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**" : El seu valor és un número de revisió -incremental- de l'alerta, per si aquesta s'ha anat modificant al llarg del temps; útil per portar un control de quantes vegades s'ha anat modificant la regla.

"**classtype**" : 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*) . 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. L'arxiu "classification.config" està format per línies amb el següent format: *config classification: nomDelClasstypeAEscollir, MissatgeQueSortiràAlLog,nºPrioritat* on el número de prioritat pot valer entre 1 (molt poc important) fins 255 (molt important). D'altra banda, tal com es pot veure, ja existeixen classtypes predefinits, com per exemple "icmp-event", entre d'altres.

"**reference**" : 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 "reference.config"

Característiques a avaluar del paquet 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 de Maxmind que Suricata té integrada de sèrie, 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. Només funciona si Suricata s'ha compilat integrant la llibreria "GeoIP-devel"

Característiques a avaluar del paquet 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.

Característiques a avaluar del paquet 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:>30* o *itype:<30*, 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:>30* o *icode:<30*, 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 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 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)

"**depth**": 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. Aquesta opció s'escriu sempre després de l'opció "content".

"**offset**" : El seu valor és un número sencer que indica quants bytes Suricata "saltarà" des del principi 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 sempre després de l'opció "content" i es pot combinar amb "depth" per especificar 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. 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")

"**pcre**" : Aquesta opció és similar a "content" però permet especificar expressions regulars com a valor a buscar dins del contingut del paquet. 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).

"**flags**" : Serveix per identificar paquets TCP amb el flag indicat actiu ("S", "A", "R", "F", "P", "U")

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 pròpiament dita, -"GET / HTTP/1.1", 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 emmagatzemar els possibles fitxers transferits en el trànsit detectat d'aquests protocols.

NOTA IMPORTANT: 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 "type->files" dins de la secció "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 juntament amb el propi fitxer guardat si s'indica l'opció "write-fileinfo" sota la secció "file-store" de l'arxiu de configuració de Suricata a yes.

NOTA: Per fer una netejar dels fitxers guardats durant cert temps es pot utilitzar la comanda *suricatactl filestore prune*

"**filename**" : Aquesta opció és un filtre per "filestore". El seu valor és una cadena que representa el nom del fitxer que es vol grabar

"**fileext**" : Aquesta opció és un filtre per "filestore". El seu valor és una cadena que representa l'extensió dels fitxers que es volen grabar

"**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 grabar. 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.

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 tcp any any -> any 80 (msg:"Bad method"; content:!"HEAD"; http_method; content:!"TRACE"; http_method; sid:100000;)

*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:10000000; rev:1;)

*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; rev:1;)

c) Executa la comanda *sudo suricata-update* (https://github.com/OISF/suricata-update) . ¿Què fa aquest programa? ¿Tindria sentit realitzar una tasca programada executant aquest programa, per exemple, cada dia?

NOTA: Antigament s'utilitzaven programes no oficials per realitzar la mateixa tasca. Exemples coneguts que encara s'utilitzen (sobre tot amb Snort, que té un sistema de regles compatibles amb Suricata) són "Oinkmaster" o "PulledPork"

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

alert icmp any any -> $HOME_NET any (msg:"ICMP connection attempt"; sid:1000002; rev:1;)
alert tcp any any -> $HOME_NET 23 (msg:"TELNET connection attempt"; sid:1000003; rev:1;)

dII) Afegeix, sota la secció "rule-files" de l'arxiu "/etc/suricata/suricata.yaml" la línia: - *test.rules*

dIII) Reinicia el servei Suricata (*sudo systemctl restart suricata*). Tot seguit, executa la comanda *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 -c 4 x.x.x.x* (on "x.x.x.x" representa la IP de la màquina on s'està executant Suricata). ¿Què veus al primer terminal? ¿Per què? Si ara, en un segon terminal de la màquina on s'està executant Suricata escrius la comanda *nc -l -p 23* i en el terminal de la màquina real executes 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 primer terminal? ¿Per què?

> NOTA: També es pot executar Suricata directament des del terminal simplement escrivint: *sudo suricata -c /etc/suricata/suricata.yaml -i enp0s3* , i afegint el paràmetre *-D* si volem que es posi en segon pla.
>
> 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

2.-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*. Crea una regla dins del mateix arxiu "test.rules" que detecti quan algú faci una petició a aquest servidor que inclogui la cadena "abc" (la qual suposarem que és molt sospitosa) dins de la URL. Un cop reiniciat Suricata, obre un navegador (o usa *curl*) de la màquina real i escriu alguna URL amb aquesta cadena. ¿S'ha generat l'alerta?

> 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 volem trobar explícitament aquest tipus de caràcters no ho podem trobar amb "http_uri" perquè ja hauran desaparescut. La solució llavors és utilitzar l'opció "http_raw_uri", que tracta la url "a pèl" sense fer cap normalització prèvia

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

alert http any any -> $HOME_NET 80 (msg:"FILE store all"; filestore; sid:111111111; rev:1;)

...i configura Suricata per a què emmagatzemi a "/var/log/suricata/filestore" tots els fitxers detectats per la regla anterior. Assegura't també que la subsecció "types->files" dins de la secció "eve-log" estigui descomentada (per defecte ha d'estar-ho) per tal de què es guardi un registre de totes les descàrregues al fitxer "eve.json". Reinicia Suricata.

bII) 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 tres "event_type" que hi apareixen relacionats amb aquest fitxer "hola.txt"? D'altra banda, ¿què és el que obtens dins de la carpeta "/var/log/suricata/filestore"?

c) 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:"User-Agent|3a| BlackSun"; nocase; http_header; 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 sentit tindria, en relació a la regla anterior, executar la comanda *curl -A "BlackSun" www.google.com?*

NOTA:User-agent strings are sometimes used by malware authors as an authentication token -- the command-and-control server will not issue commands to computers that make requests of it unless the correct user-agent string is specified by the client in the HTTP session. This is one way malware authors evade malware researchers. Luckily for security professionals, these user-agent strings can be very good indicators of malware presence on a system. This is why user-agent strings are included in Suricata rules.

3.- 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) Per defecte suricata-update utilitza una llista de llocs webs garantits que ofereixen regles de forma lliure. Aquesta llista es troba disponible al lloc web d'Emerging Threats (https://rules.emergingthreats.net) . 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: Look for /etc/suricata/enable.conf, /etc/suricata/disable.conf, /etc/suricata/drop.conf, and /etc/suricata/modify.conf to look for filters to apply to the downloaded rules (these files are optional and do not need to exist)
NOTA: Òbviament, també existeixen les comandes *sudo suricata-update disable-source .../...* o *sudo suricata-update remove-source ../..*

4.-a) Suposant que estàs executant Suricata sobre un sistema que fa de "gateway", ¿per a què serviria la següent parella de regles?

alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg: "Outbound ICMP detected"; sid:1000001;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "Inbound ICMP detected"; sid:1000002;)

b) Suposant que estàs executant Suricata sobre un sistema amb IP 192.168.15.123 i on s'està executant un servidor HTTP,, ¿per a què serviria la següent regla?

alert http any any -> 192.168.15.123 !80 (msg: "HTTP but not port 80"; sid:1000003; rev:1;)

c) Suposant que tens un servidor web a la teva xarxa interna i un IDS Suricata amb la següent regla definida, ¿què és el que et permet detectar?

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:1000001; rev:1;)

NOTA: Consultar, si és necessari, https://suricata.readthedocs.io/en/suricata-5.0.0/rules/flow-keywords.html i https://suricata.readthedocs.io/en/suricata-5.0.0/rules/thresholding.html

d) Prova una de les regles d'exemple (la que tu vulguis) indicades al quadre dels paràgrafs blaus i fes el necessari per provocar l'alerta corresponent

5.-a) Observa la subsecció "outputs" dins de la secció "logging" de l'arxiu "suricata.yaml" per esbrinar si, a més de l'arxiu "suricata.log", el registre de funcionament de Suricata es guarda en algun altre lloc.

b) Llegeix els comentaris o la documentació oficial per deduir quins fitxers es crearien (i amb quin contingut) si s'activen les subseccions "http-log", "tls-log" i "pcap-log" de la secció "outputs" de l'arxiu "suricata.yaml"

c) ¿Per a què serveix la comanda *suricatactl* descrita aquí: https://suricata.readthedocs.io/en/suricata-5.0.0/unix-socket.html ? Pista: Un "Unix socket" no és res més que un mecanisme de comunicació entre processos executant-se a la mateixa màquina

6.-a) ¿Què és https://evebox.org i què hi té a veure ELK amb ell? Pista: pots veure una demo a https://demo.evebox.org D'altra banda, ¿què és http://rocknsm.io ?

b) ¿Què és https://github.com/StamusNetworks/Scirius? ¿Què hi té a veure https://www.stamus-networks.com/open-source amb Scirius i Evebox?

7.-a) ¿Què fa aquest programa: https://github.com/fail2ban/fail2ban ?

b) ¿Què fa aquest programa : https://github.com/mrash/psad ? Esbrina si funciona només amb Iptables o també amb Nftables