Laboratorul 6
Protecție avansată bazată pe ids/ips a mașinilor/nodurilor/mașinilor virtuale de orice tip din nor sau de sine stătătoare

configurație standard pentru testare
- o mașină virtuală debian bullseye curată - îi faceți și un copie de rezervă (snapshot) pentru o eventuală restaurare (rollback) în caz de prostii prea mari aflată în rețea internă virtuală cu o alta pentru pentest - poate să fie același tip de mașină unde începeți să instalați instrumente pentru pentest dar pentru fii Windows-ului merge și Kali.

1. Zeek Network Security Monitor - https://zeek.org/

Inițial era denumit Bro și a rămas un software gratuit sub licență BSD. Acesta are o abordare ușor diferită față de IDS-urile standard. O abilitate constă în aceea că a fost construit pentru flexibilitate maximă. Ca rezultat el va permite, cu ajutorul unor limbaje de tip script care sunt specifice în funcție de domeniu, dezvoltarea de module care permite monitorizarea particularizată pentru fiecare site aflat sub control. El este dedicat pentru gestiunea rețelelor și site-urilor de mare performanță și încărcare. Nu se bazează pe abordări clasice cu privire la seturi de semnături sau pe anumite scheme de detecție standard. Rezultă că nici acesta nu este recomandat utilizatorilor medii. Ca și în cazul anterior furnizează jurnale detailate cu privire la activitățile din rețea permițându-ne totodată analize semantice de nivel superior asupra acestora prin intermediul modulelor dedicate puse la dispoziție. De asemenea monitorizează extensiv nivelul aplicației din modelul ISO-OSI. A fost gândit cu interfețe deschise ceea ce permite interoperabilitate aproape nelimitată inclusiv schimb în timp real de informații cu alte aplicații de profil.
evident că ne mai trebuie niște biblioteci (în special pe o mașină deabia instalată)

```
apt install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3 python3-dev swig zlib1g-dev
```

pentru putea utiliza și funcționalitățile extinse ale acestuia mai am nevoie de:

```
apt install python3-git python3-semantic-version
pip3 install GitPython semantic-version
apt install libmaxminddb-de sendmail curl libjemalloc-dev libkrb5-dev
```

ne mai trebuie și o deviere (branch) de la instrumentele de performanță google gperftools de la
https://github.com/gperftools/gperftools
dacă se doresște utilizarea unui cluster dedicat (PF_RING) vezi la
https://docs.zeek.org/en/current/cluster-setup.html#pf-ring-config
pentru analiza pachetelor de rețea Ipsumdump vezi la
https://github.com/kohler/ipsumdump
pentru geolocație automată în afară de biblioteca suport mai vedeți la
https://docs.zeek.org/en/current/frameworks/geoip.html#geolocation

instalarea propriuzisă

```
cd sources
git clone --recursive https://github.com/zeek/zeek
cd zeek
./configure && make && sudo make install
```

apoi

```
nano hello.zeek
```

unde punem un stupid people program de testare a.k.a veșnicul ”hei lume”

```
# File "hello.zeek"
event zeek_init()
	{
	print "Hei Lume!";
```

```
}
```
salvăm și apoi testarea este imediată
/usr/local/zeek/bin/zeek hallo.zeek
o documentație introductivă pentru scriptingul suportat de zeek se găsește la
https://try.zeek.org/

Tema 1. instalați și testați zeek. Încercați să creați un declanșator asociat unei raportări dintr-un jurnal de securitate

2. https://suricata-ids.org/

Poate fi văzut ca succesorul Snort dar cu o serie de abilități pe care acesta încă nu le-a dobândit. Mai mult nivelul de integrare nativă cu serviciile și instrumentele de tip nor (”cloud”) îi dă un mare avantaj în fața acestuia. Din nefericire fiind relativ nou suportul oferit de comunitate este relativ scăzut și atunci pentru mulți administratori utilizarea lui este relativ limitată la utilizarea consolă sau a componentelor suplimentare care permit o gestionare vizuală. Aceste lucruri nu îi diminuează valoarea curentă dar face greoaie dezvoltarea de instrumente mai particularizate pentru necesitățile utilizatorului. Este orientat pe monitorizarea rețelei dar nivelul de reactivitate la atacuri pe care îl posedă este superior snort-ului. Este din categoria “real time intrusion detection”, “inline intrusion detection”, “network security monitoring NSM” dar și procesare offline PCAP. Pentru detectarea unor amenințări mai complexe există un suport pentru script-uri Lua. Datorită faptului că utilizează YAML și JSON pentru fișierele de intrare și ieșire este compatibilă cu instrumente cum ar fi SIEMs, Splunk, Logstash/Elasticsearch, Kibana, precum și alte baze de date.

Instalare Suricata

Ca de obicei pe bullseye & su. Tot ca de obicei vom instala minimele necesare instalării. Dacă le aveți nu este necesar să încercați reinstalarea. Dacă mergeți pe copy paste în bunele obiceiuri prin reinstalare uneori s-ar putea sa vă modificați o serie de configurări personalizate. Dacă aveți numai o parte din pachete particularizați corespunzător linia de comandă. Pentru mașinile de abia instalate vă mai trebuie și autocompletarea la nivel de bash. Pentru aceasta:

```
apt-get install bash-completion
nano ~/.bashrc
```

și adăugați la sfârșitul fișierului:

```
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
fi
```

apoi autocomplete în bash merge (logout/login). Acum ne putem întoarce la treburile noastre.

```
apt -y install libpcre3 libpcre3-dbg libpcre3-dev build-essential autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libjansson4 pkg-config wget
```

și

```
apt -y install libnspr4-dev libnss3-dev liblz4-dev rustc cargo python3-distutils python3-pip
```

După cum am discutat la curs orice astfel de sistem utilizează un set minimal de reguli care poate fi ulterior extins. Deci ar fi o bună idee să-l instalăm și pe acesta.

```
pip3 install --upgrade suricata-update
```

și

```
ln -s /usr/local/bin/suricata-update /usr/bin/suricata-update
```

Dacă doresc să instalez și abilitățile de tip IPS ale suricatei (ea vine implicit numai ca IDS) atunci mai este nevoie de următoarele biblioteci:

```
apt -y install libnetfilter-queue-dev libnetfilter-queue1 libnfnetlink-dev libnfnetlink0
```

În sfârșit putem aduce fișierele sursă ale aplicației

```
cd sources
```

Dacă nu există creați-l ca și utilizarea opt pentru looser face parte din bunele practici. Vă rog verificați la respectiva adresă dacă între timp nu a mai apărut o versiune mai nouă și să modificați corespunzător comanda. În situații de producție este bine să aduceți și fișierul cu semnătura pentru a verifica că nu există alterări binevoitoare - pentru laborator nu este nevoie.

wget https://www.openinfosecfoundation.org/download/suricata-6.0.3.tar.gz

Acum îl dezarhivăm

tar zxvf suricata-6.0.3.tar.gz

și trecem la configurarea pentru compilare

cd suricata-6.0.3

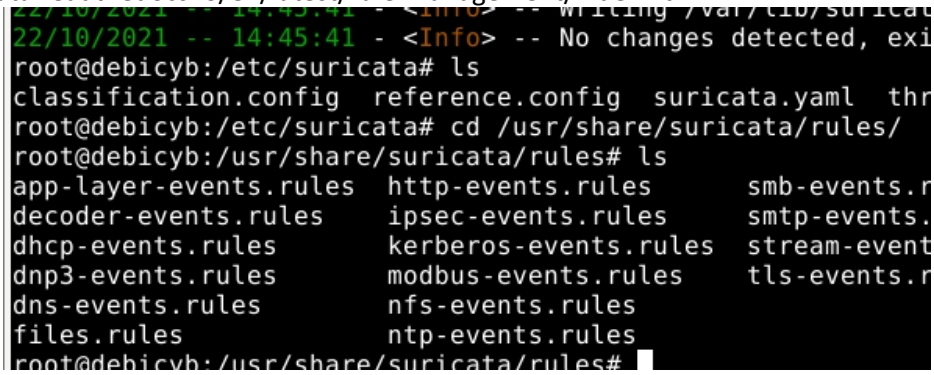
acum se poate trece la următorul pas în configurarea suricatei si apoi compilare și instalare

./configure --enable-nfqueue --prefix=/usr/ --sysconfdir=/etc/ --localstatedir=/var/

make install-full

Comanda make install-full doar va instala suricata (în configurație inițială) și apoi setul de reguli pentru acesta cu ajutorul comenzii dedicate suricata-update. Dacă instalarea este efectuată cu succes atunci se va obține o ieșire de acest tip:

```
You can now start suricata by running as root something like:
   /usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth0
If a library like libhtp.so is not found, you can run suricata with:
   LD_LIBRARY_PATH=/usr/lib /usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth0
For more information please see:
   https://suricata.readthedocs.io/en/latest/rule-management/index.html
```



Se observă că regulile se găsesc în directorul

/usr/share/suricata/rules/

iar fișierele de configurare la

/etc/suricata/suricata.yml

Configurarea Suricatei

Ca de obicei în cazul unui sistem de protecție al rețelei trebuie ținut cont în configurare de diferența între rețeaua internă și cea externă. Pentru aceasta avem la dispoziție două variabile HOME_NET și EXTERNAL_NET. Se va deschide suricata.yaml și se vor modifica corespunzător cele două variabile dacă este cazul. Informațiile despre rețele interne și externe le aflăm cu ip a (vezi laboratoarele anterioare).

```
root@debicyb:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state U
P group default qlen 1000
    link/ether 00:0c:29:5e:a6:6d brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 172.16.63.129/24 brd 172.16.63.255 scope global dynamic noprefixrout
e ens33
       valid_lft 1680sec preferred_lft 1680sec
    inet6 fe80::20c:29ff:fe5e:a66d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Se observă ca ens33 este rețeaua externă și deoarece nu suntem pe un server cu separare corectă pe două sau mai multe dispozitive de rețea a subrețelelor atunci putem pentru testare ca să stabilim EXTERNAL_NET ca fiind !$HOME_NET ceea ce înseamnă că orice adresă de ip în afara celei/celor din grupul HOME_NET va fi considerată ca aparținând grupului EXTERNAL_NET.
Acum putem trece la modificarea, dacă este cazul, a variabilelor din fișierului de configurare
nano /etc/suricata/suricata.yaml

```
  GNU nano 5.4              /etc/suricata/suricata.yaml
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

Un exemplu de activare pentru aducere la zi a regulilor a tutoru surselor de reguli mai utilizate
suricata-update enable-source oisf/trafficid
suricata-update enable-source etnetera/aggressive
suricata-update enable-source sslbl/ssl-fp-blacklist
suricata-update enable-source et/open
suricata-update enable-source tgreen/hunting
suricata-update enable-source sslbl/ja3-fingerprints
suricata-update enable-source ptresearch/attackdetection
cu aceste noi surse mai lansez un suricata-update iar ca să văd toate sursele active la un moment dat
suricata-update list-enabled-sources
dacă doresc s dezactivez o anumita sursă
suricata-update disable-source et/pro
dacă doresc să o șterg complet din listă
suricata-update remove-source et/pro

Funcționarea automată
ln -s /usr/bin/suricata /sbin/suricata

Atunci când suricata este instalată din susrse ca în acest caz nu este furnizată automat o integrare cu systemd totuși este prevazută o formă standard de serviciu care poate fi instalat dacă este necesar și anume suricata.service.
cp ~/sources/suricata-6.0.3/etc/suricata.service /etc/systemd/system/
apoi editați serviciul
nano /etc/systemd/system/suricata.service
acolo decomentați Environment files apoi editați fișierul de configurare
nano /etc/default/suricata
și adăugați
OPTIONS="-q 0"
salvați și apoi
systemctl daemon-reload
systemctl enable suricata
systemctl start suricata
pentru o testare rapida
apt install hping3
acum de pe altă mașină kali sau nu se poate realiza o testare
hping3 -c 10000 -d 120 -S -w 64 -p 22 --flood "IP_MASINA_VIRTUALA"
După aceea vă uitați în jurnalele suricatei (vezi în suricata.yaml unde este directorul cu jurnalele produse de aceasta). Ar fi bine să citiți cu foarte mare atenție acest fișier pentru că vă oferă toate informațiile necesare configurării mai fine a suricatei. Pe moment ce am făcut până acum a pus suricata în modul IPS, evident că potențialul ei este atins când este activat și modul IDS.

Tăiem porcul sau nu? (vezi reprezentare snort)
Unul din motivele adopției greoaie a suricata este faptul ca are doar o interoperare primară cu elastic search kibana (evident se poate și cu mariodb si apache) dar nivelul de control si modificare preferat de experti este încă scăzut iar instrumentele clasice pentru snort (ex. Snorby & barnyard2) sunt cam vechi. Totuși există deja noi elemente în ecosistem cum ar fi atsp care este o abordare bazată pe Golang pentru a înlocui pe Oinkmaster și PulledPork care sunt specifice snort dar care sunt necesare.
apt install golang-go
cd sources
git clone https://github.com/pathaugen/atsp
cd atsp
go test
go build
./atsp

dacă nu puteți rămâne pe abordările mai vechi dar mai lente și depășite cum ar fi stăpânul guițatului.
- **Oink(Oink)Master.** Instalarea este imediată
apt install oinkmaster

apoi trebuie editat fișierul /etc/oinkmaster.conf pentru a spune de unde trebuie aduse regulile exemplu
url = http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz
deci
nano /etc/oinkmaster.conf
și apoi puteți adăuga o sursă (poate citiți și comentariile din respectivul fișier de configurare). Acum putem executa evident cu specificarea zonei de păstrare a regulilor /usr/share/suricata/rules adică
oinkmaster -C /etc/oinkmaster.conf -o /usr/share/suricata/rules
Cei cu ”copy-paste” nu săriți în sus vedeți dacă doriți să utilizați acest instrument și dacă da verificați unde este calea regulilor conform instalării mangustei. Comanda poate fi utilizată și pentru menținerea la zi a regulilor și atunci poate fi utilizat de exemplu un crontab la nivel supervisor.
0 1 * * * oinkmaster -C /etc/oinkamaster.conf –o /etc/suricata/rules
Nu uitați că deja suricata are propria ei comandă de menținere la zi deci vedeți care sunt diferențele între cele două abordări pentru a putea lua decizii în cunoștință de cauză mai ales că barnyard este cam oale și ulcele.

Gestiunea jurnalelor produse de mangustă
Atât la nivelul clusterelor, al gridului sau al abordărilor moderne de tip nor supravegherea centralizată sau distribuită a unor ferme mari de mașini (pot fi și milioane în cazul norului) se realizează cu ajutorul unor sisteme de gestiune a bazelor de date. Alegerea SGBD-ului se face conform celor învățate la materia respectivă ținând cont de următorii factori - numărul de mașini și cantitatea/diversitatea jurnalelor produse de o mașină. Arhitectura abordată pentru această monitorizare și control - adică dacă la nivelul SGBD-ului se mai fac și o serie de procesări complexe și sau triggere (declanșatoare) sau nu împreună cu interacțiunea aplicațiile vizuale pentru comandă și control. Astfel există interoperări imediate cu elastic search kibana etc pentru situațiile comune fără mare importanță (în special din punct vedere al securității) și abordările profesionale care presupun proiectarea unui sistem dedicat pentru această gestiune. Pentru acest ultim caz o abordare cu strat arhitectural dedicat pentru adaptare/interoperare cu diverse SGBD-uri este necesar (caz clasic hibernate sau n-hibernate). Pentru aceasta ar trebui utilizate soluții specifice. În contextul abordărilor open source pe care eu le favorizez o posibilă soluție ar putea să fie utilizarea lui meer care este un spooler pentru suricata sau sagan. Evident tot soluțiile cu proiectare de la zero sunt viitorul. Acesta are diverse plugin-uri (inserții) care ne oferă următoarele facilități de interoperare cu
- MySQL/MariaDB - utilizează o schemă a bazei de date similară cu cea din Snort/Barnyard2 ceea ce conduce la compatibilitate și cu abordări depășite cum ar fi Snorby, Sguil, BASE, etc. Totuși ținând cont de influența abordărilor de tip nor schema primară a fost extinsă cu metadate utile în generarea alertelor comune (de ex ‘flow’, ‘http’, ‘smtp’, ‘tls’, ‘ssh’).
- PostreSQL - care are aceeași abordare ca anteriorul numai că permite interoperarea cu PostgreSQL.
- Redis - poate interopera cu un SGBD Redis într-o manieră similară cu mangusta (adică operații de tipul list/lpush, rpush, channel/publish sau set).
- abordări dedicate - Acceptă apelul oricărei aplicații la apariția unor evenimente bazate pe reguli iar EVE/JSON sunt gestionate utilizând stdin.
- suport accesare prin pipe-uri - permite scrierea de EVE/JSON către un pipe UNIX anume sau FIFO.
- elasticsearch - convertește EVE/JSON (produse de Sagan & Suricata) către elasticsearch.
Mai multe detalii la https://github.com/quadrantsec/meer

Instare redis - un fel de kafka mai prost
cd sources
git clone https://github.com/redis/redis
cd redis
apt install libssl-dev

apt install tcl-tls
make BUILD_TLS=yes
dacă ați avut erori dați
make disclean
înainte de a reîncerca procesul și apoi
make install
./utils/gen-test-certs.sh
./runtest --tls

Instalare meer

apt install libjson-c-dev
apt install libhiredis-dev
cd sources
git clone https://github.com/quadrantsec/meer
cd meer
./autogen.sh
./configure
make
make install

Pentru restul sgbd-urilor ați avut exemple la laboratoarele de pp și sd.

Testarea comportamentului Suricata în cazul unui atac de tip DDoS

Deși am adus din mai multe surse o serie de seturi de reguli dacă ne uităm în /usr/share/suricata/rules nu vedem un fișier special pentru tratarea atacurilor ddos. Va trebui deci să adăugăm seturi particulare de reguli. Procesul este similar pentru orice set particularizat de reguli care trebuie adăugate. Mai întâi creăm un nou fișier de reguli denumit să zicem test-ddos.rules. Pentru aceasta ne uităm în suricata.yaml ca să aflăm care este directorul pentru regulile implicite care în cazul nostru este default-rule-path: /var/lib/suricata/rules. Atunci creez fișierul

nano /var/lib/suricata/rules/test-ddos.rules

și în el vom adăuga o regulă simplă

alert tcp any any -> $HOME_NET 80 (msg: "Posibil DDoS"; flags: S; flow: stateless; threshold: type both, track by_dst, count 200, seconds 1; sid:1000001; rev:1;)

Apoi acest fișier trebuie încărcat de mangustă deci pentru a ști că acesta există trebuie să îl referim în suricata.yaml. Editățm acest fișier și căutăm zona default-rule-path unde vom adăuga numele fișierului nostru.

```
 GNU nano 5.4                /etc/suricata/suricata.yaml *
    #
    # See Napatech NTPL documentation other hashmodes and details on their us>
    #
    # This parameter has no effect if auto-config is disabled.
    #
    hashmode: hash5tuplesorted

##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /var/lib/suricata/rules

rule-files:
  - suricata.rules
  - test-ddos.rules
##
## Auxiliary configuration files.
##

classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

Acum putem, după ce pornim suricata să verificăm eficiența acestei noi reguli, dar înainte de aceasta pentru simplitatea testării trebuie să dezactivăm funcționalitățile care gestionează supraîncărcarea cu pachete a dispozitivului nostru de rețea (ens33) - pe care ascultă mangusta (vezi configurările anterioare).

ethtool -K ens33 gro off lro off

Pentru a verifica ca aceasta comanda a avut efect putem da

ethtool -k ens33 | grep large

Înainte de a trece la intercepție activă a traficului - așa numitul mod promiscuu trebuie să configurăm explicit VmWare ca să permită o astfel de utilizare pentru o mașină virtuală. De pe sistemul de operare gazdă (de unde se lansează vmware) - evident Linux se va da următoarea comanda - la nivel supervisor desigur.

chmod a+rw /dev/vmnet0 (inital are numai rw by owner) si nu numai

Dacă fenomenul de blocare persistă atunci trebuie să le schimbăm parametrii la toate dispozitivele utilizate de VmWare.

```
root@[redacted]:~# ls /dev | grep vmnet
vmnet0
vmnet1
vmnet8
root@[redacted]:~# chmod a+rw /dev/vmnet1
root@[redacted]:~# chmod a+rw /dev/vmnet8
root@[redacted]:~# chmod a+rw /dev/vmnet0
root@[redacted]:~#
```

care permite oricărui utilizator să aibă mașini virtuale care utilizează modul promiscuu de lucru. Pentru Windows trebuie să vedeți voi - fie vă întreabă direct dacă permiteți intrarea în acest mod iar dacă nu găsiți instrucțiuni la VmWare.

Acum se poate lansa mangusta în modul de captură în timp real a pachetelor - PCAP live - cu ajutorul următoarei comenzi

```
suricata -D -c /etc/suricata/suricata.yaml -i ens33
```

acum din altă mașină virtuală poate fi lansată o testare a funcționării

```
hping3 -c 10000 -d 120 -S -w 64 -p 22 --flood "IP_MASINA_VIRTUALA"
```

și vă uitați în **/var/log/suricata/fast.log**

Tema 2

Instalați și configurați meer și suricata. Efectuați un test cu un atac simplu al mașinii.Apoi realizați și integrarea cu o bază de date la alegere.

Temă pe acasă

Activați modul IDS al mangustei apoi instalați lanțul complet de aplicații (preluare jurnale, urmărire, reacții și anunțuri/alerte automate (bazate pe declanșatoare) - cu meer - fie pentru zeek fie pentru mangusta fie pentru amândouă.

Tema pentru insomniaci - evident opțională pentru restul studenților

Instalați și Elasticsearch, Logstash, și Kibana pe o a treia mașină virtuală apoi integrați jurnalele suricatei sau și ale lui zeek (de exemplu - fiecare pe o masină virtuală separată), adăugați și declanșatoare suplimentare și aveți fluxul complet în stil nor!