

Anexo. Uso de Suricata

Suricata es un motor de red de alto rendimiento **IDS (Intrusion Detection System)**, **IPS** y seguridad de red, desarrollado por el **OISF**, esta es una aplicación de código abierto multiplataforma y es propiedad de una fundación sin ánimo de lucro de la comunidad **Open Information Security Foundation (OISF)**.



Está basado en un conjunto de reglas desarrolladas externamente para supervisar el tráfico de la red y proporcionar alertas al administrador del sistema cuando se producen eventos sospechosos. Diseñada para ser compatible con los componentes de seguridad de red existentes, ofrece funcionalidad de salida unificada y opciones de biblioteca conectables para aceptar llamadas de otras aplicaciones. Como un motor de múltiples hilos, ofrece una mayor velocidad y eficiencia en el análisis de tráfico de red.

Suricata Es un sistema que permite implementar IDS e IPS. Se configura con una serie de reglas, compatibles con **Snort** y otros sistemas.

La página oficial es: <https://suricata.io>



Suricata is a high performance, open source network analysis and threat detection software used by most private and public organizations, and embedded by major vendors to protect their assets.

Suricata is far more than an IDS/IPS

1. Instalación

Vamos a instalarla en Linux Kali:

En primer lugar, añadimos el repositorio de **Suricata y jq**, que es un lector de registros:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install suricata jq
```

2. Comprobación

Para ver información sobre **Suricata**:

```
whereis suricata
suricata --build-info
service suricata start
service suricata status
```

```
(kali@kali)-[~]
└─$ service suricata status
● suricata.service - Suricata IDS/IDP daemon
   Loaded: loaded (/usr/lib/systemd/system/suricata.service; disabled; preset: disabled)
   Active: active (running) since Wed 2024-04-10 23:19:54 EDT; 56s ago
     Docs: man:suricata(8)
           man:suricatasc(8)
           https://suricata.io/documentation/
  Process: 2382 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml ->
 Main PID: 2385 (Suricata-Main)
    Tasks: 7 (limit: 461)
  Memory: 56.9M (peak: 57.1M swap: 4.0K swap peak: 4.0K)
     CPU: 689ms
    CGroup: /system.slice/suricata.service
            └─2385 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile>

abr 10 23:19:54 kali systemd[1]: Starting suricata.service - Suricata IDS/IDP daemon...
abr 10 23:19:54 kali suricata[2382]: i: suricata: This is Suricata version 7.0.3 RELEASE runn>
abr 10 23:19:54 kali systemd[1]: Started suricata.service - Suricata IDS/IDP daemon.
```

3. Ficheros de configuración.

Al ejecutar `suricata --build-info` observamos donde se encuentran algunos ficheros:

```
Generic build parameters:
Installation prefix:      /usr
Configuration directory: /etc/suricata/
Log directory:           /var/log/suricata/
```

Como hemos visto, el fichero de configuración se encuentra en:

```
/etc/suricata/
```

Veamos lo que se encuentra en el directorio de configuración:

```
ls -l /etc/suricata/
```

```
(kali㉿kali)-[~]
$ ls -l /etc/suricata
total 100
-rw-r--r-- 1 root root 3327 feb  8 04:35 classification.config
-rw-r--r-- 1 root root 1375 feb  8 04:35 reference.config
drwxr-xr-x 2 root root 4096 abr 10 23:14 rules
-rw-r--r-- 1 root root 85175 feb  8 17:22 suricata.yaml
-rw-r--r-- 1 root root 1643 feb  8 04:35 threshold.config
```


El fichero de configuración de suricata es **suricata.yaml**, que está dividido en 4 partes:

```
(kali㉿kali)-[~]  
└─$ cat /etc/suricata/suricata.yaml | grep Step  
## Step 1: Inform Suricata about your network  
## Step 2: Select outputs to enable  
## Step 3: Configure common capture settings  
## Step 4: App Layer Protocol configuration
```

Parte 1. Configuración de redes

Parte 2. Configuración de activación de salidas

Parte 3. Ajuste de las capturas

Parte 4. Configuración de la capa de protocolo de aplicación

Veamos algunas partes de este fichero:

```
Sudo nano /etc/suricata/suricata.yaml
```

Parte 1. Configuración de redes:

Se definen varias variables:

HOME_NET indica el rango de ips que va a considerar dentro de la red interna

EXTERNAL_NET indica que el resto es red externa

```
## Step 1: Inform Suricata about your network
##

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"
```

Parte 2. Configuración de activación de salidas

Se definen varias variables:

Default-log-dir: /var/log/suricata/ define donde se encuentra el fichero donde se almacenan los registros log

Parte 3. Ajuste de las capturas

Hay que indicar cual es nuestra interface de red, por defecto está:

```
#  
# Step 3: Configure common capture settings  
#  
# See "Advanced Capture Options" below for more options, including Netmap  
# and PF_RING.  
#  
  
Linux high speed capture support  
f-packet:  
- interface: eth0  
# Number of receive threads. "auto" uses the number of cores  
#threads: auto  
# Default clusterid. AF_PACKET will load balance packets based on flow.  
cluster-id: 99  
# Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.  
# This is only supported for Linux kernel > 3.1  
# possible value are:  
# * cluster_flow: all packets of a given flow are sent to the same socket  
# * cluster_cpu: all packets treated in kernel by a CPU are sent to the same socket  
# * cluster_qm: all packets linked by network card to a RSS queue are sent to the sa
```


4. Creación de reglas.

En el directorio:

```
/etc/suricata/rules/
```

Se encuentran las reglas definidas.

```
root@pru-VirtualBox:/home/pru# ls /etc/suricata/rules
app-layer-events.rules  dnp3-events.rules  http2-events.rules  kerberos-events.rules  nfs-events.rules  smtp-events.rules  tls-events.rules
decoder-events.rules   dns-events.rules   http-events.rules   modbus-events.rules   ntp-events.rules  ssh-events.rules
dhcp-events.rules      files.rules        ipsec-events.rules  mqtt-events.rules     smb-events.rules  stream-events.rules
root@pru-VirtualBox:/home/pru#
```

Veamos el contenido de un fichero de reglas:

```
root@pru-VirtualBox:/home/pru# cat dhcp-events.rules
cat: dhcp-events.rules: No existe el archivo o el directorio
root@pru-VirtualBox:/home/pru# cat /etc/suricata/rules/dhcp-events.rules
# DHCP app-layer event rules.  See
#
# https://redmine.openinfosecfoundation.org/projects/suricata/wiki/AppLayer
# for SID allocation.

alert dhcp any any -> any any (msg:"SURICATA DHCP malformed options"; app-layer-event:dhcp.malformed_option; classtype:protocol-command-decode; sid:2227000; rev:1;)
alert dhcp any any -> any any (msg:"SURICATA DHCP truncated options"; app-layer-event:dhcp.truncated_option; classtype:protocol-command-decode; sid:2227001; rev:1;)
root@pru-VirtualBox:/home/pru#
```

El **formato de las reglas** es el siguiente:

Acción **Encabezado** **Opciones de la regla**

Acción:

alert. genera una alerta y deja continuar el paquete

pass. Deja de inspeccionar el paquete

drop. Descarta del paquete y genera una alerta (el receptor no recibe ningún mensaje)

reject. Se envía el error RST/ICMP de destino inalcanzable al emisor del paquete

rejectsrc. Se envía el error RST/ICMP de destino inalcanzable al emisor del paquete

rejectdst. Se envía el error RST/ICMP de destino inalcanzable al receptor del paquete

rejectboth. Se envía el error RST/ICMP de destino inalcanzable al emisor y receptor del paquete

Encabezado:

Protocolo **IP_origen** **puerto_origen** **Sentido** **IP_destino** **puerto_destino**

Protocolo

Básicos: tcp, udp, icmp, ip

De aplicación: http, ftp, tls (ssl), smb, dns, dcerpc, ssh, smtp, imap

Otros: modbus, dnp3, enip, ntp, dhcp, rfb, rdp, http2, ...

Opciones de la regla

Dependiendo de cada regla:

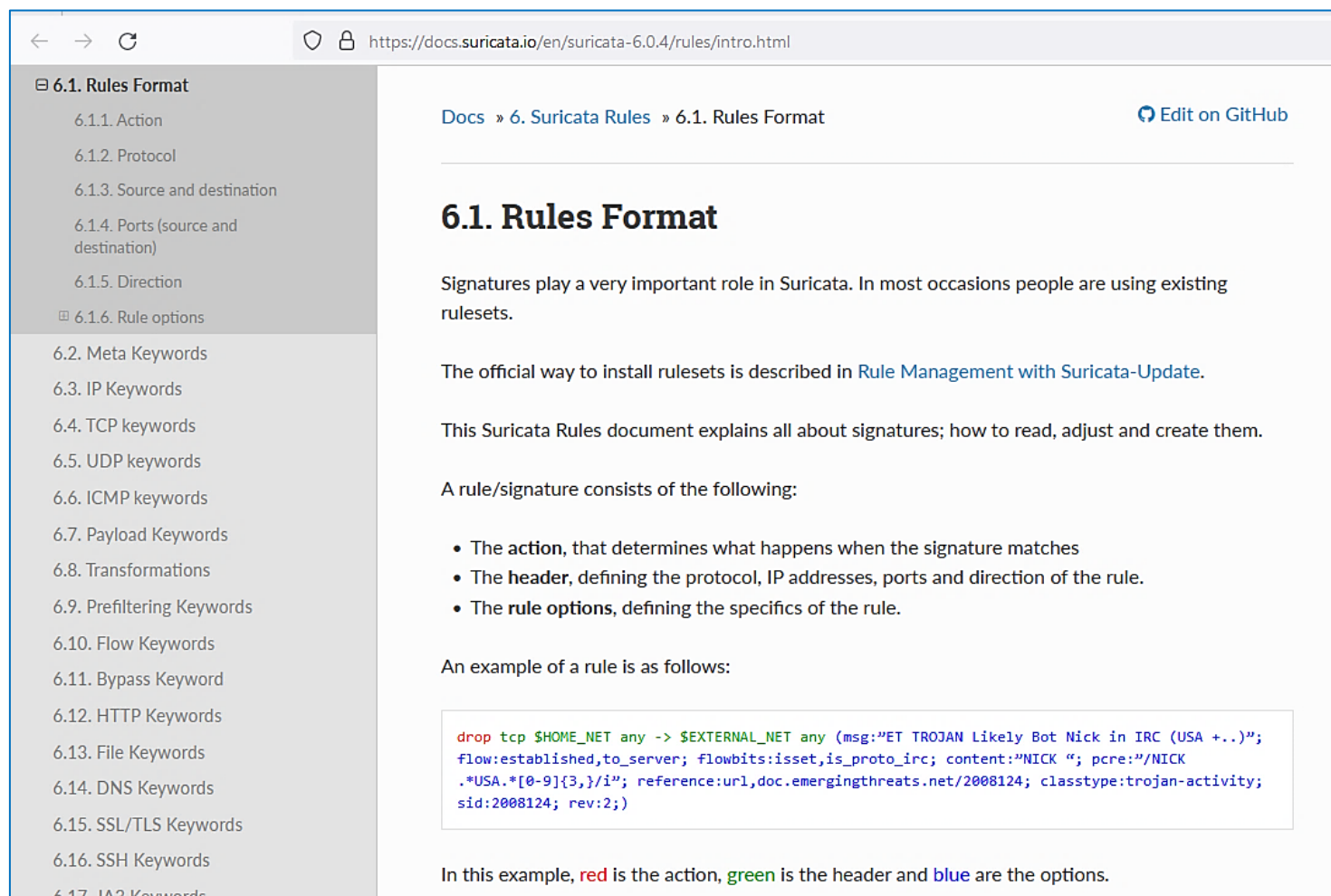
keyword: settings; keyword:settings...

Ejemplo:

alert **icmp** **any any** -> **any any** (msg: "ICMP Packet found"; sid:2000001;rev1;)

El formato lo puedes ver en la documentación de la web de suricata:

[6.1. Rules Format — Suricata 6.0.4 documentation](https://docs.suricata.io/en/suricata-6.0.4/rules/intro.html)



The screenshot shows a web browser displaying the Suricata 6.0.4 documentation page for Rules Format. The browser's address bar shows the URL <https://docs.suricata.io/en/suricata-6.0.4/rules/intro.html>. On the left, a sidebar contains a table of contents with items like 6.1. Rules Format, 6.1.1. Action, 6.1.2. Protocol, etc. The main content area has a breadcrumb trail 'Docs » 6. Suricata Rules » 6.1. Rules Format' and an 'Edit on GitHub' link. The title '6.1. Rules Format' is prominently displayed. The text explains the role of signatures and provides an example rule with color-coded parts: red for action, green for header, and blue for options.

Docs » 6. Suricata Rules » 6.1. Rules Format [Edit on GitHub](#)

6.1. Rules Format

Signatures play a very important role in Suricata. In most occasions people are using existing rulesets.

The official way to install rulesets is described in [Rule Management with Suricata-Update](#).

This Suricata Rules document explains all about signatures; how to read, adjust and create them.

A rule/signature consists of the following:

- The **action**, that determines what happens when the signature matches
- The **header**, defining the protocol, IP addresses, ports and direction of the rule.
- The **rule options**, defining the specifics of the rule.

An example of a rule is as follows:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)";
flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity;
sid:2008124; rev:2;)
```

In this example, **red** is the action, **green** is the header and **blue** are the options.

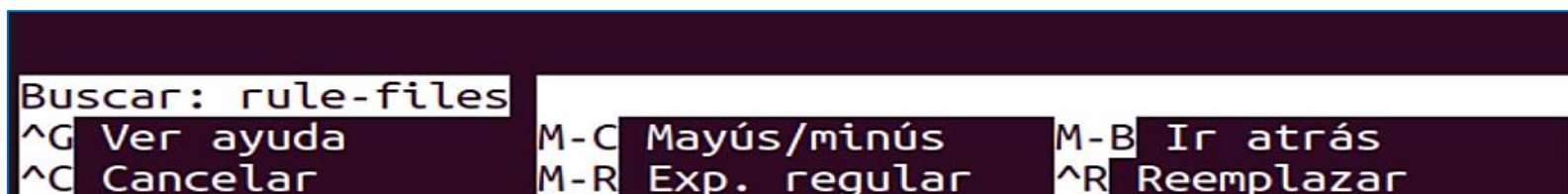
Vamos a crear un fichero con nuestras propias reglas:

```
sudo touch /etc/suricata/rules/my.rules
```

Ahora, editamos el fichero de configuración de suricata (suricata.yaml):

```
sudo nano /etc/suricata/suricata.yaml
```

Buscamos **rule-files**:



Le indicamos el directorio donde estarán las reglas (**/etc/suricata/rules**) y el nombre del fichero de reglas (**my.rules**):

```
#default-rule-path: /var/lib/suricata/rules
default-rule-path: /etc/suricata/rules

rule-files:
# - suricata.rules
- my.rules

##
## Auxiliary configuration files.
##
```

Ahora, reiniciamos el servicio, para que se apliquen los cambios:

```
service suricata restart
```

5. Detectar paquetes ICMP.

Vamos a crear una regla para detectar paquetes ICMP. Para ello, abrimos el fichero de reglas creado:

```
sudo nano /etc/suricata/rules/my.rules
```

Creamos la regla:

```
alert icmp any any -> any any (msg: "Paquete ICMP encontrado"; sid:1; rev:1;)
```

```
#  
# Detección de paquetes ICMP  
#  
alert icmp any any -> any any (msg: "Paquete ICMP encontrado"; sid:1; rev:1;)
```

Primero mostramos el fichero log, para comprobar que está funcionando:

```
tail -f /var/log/suricata/fast.log
```

Ahora reiniciamos el servicio suricata:

```
service suricata restart
```



```

pru@pru-VirtualBox:~$ service suricata stop
pru@pru-VirtualBox:~$ service suricata start
pru@pru-VirtualBox:~$ service suricata status
● suricata.service - LSB: Next Generation IDS/IPS
   Loaded: loaded (/etc/init.d/suricata; generated)
   Active: active (exited) since Fri 2023-05-05 04:28:28 WEST; 4s ago
     Docs: man:systemd-sysv-generator(8)
   Process: 3274 ExecStart=/etc/init.d/suricata start (code=exited, status=0/SUCCESS)

may 05 04:28:28 pru-VirtualBox systemd[1]: Starting LSB: Next Generation IDS/IPS...
may 05 04:28:28 pru-VirtualBox suricata[3274]: Starting suricata in IDS (af-packet) mode... done.
may 05 04:28:28 pru-VirtualBox systemd[1]: Started LSB: Next Generation IDS/IPS.

```

Ahora, desde otro equipo, realizamos un ping:

```

zsh: corrupt history file /home/pru/.zsh_history
(pru@Newkali)-[~]
$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
 64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.358 ms
 64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.420 ms
 64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.295 ms
 64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.385 ms
^C
--- 10.0.2.15 ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 3068ms
 rtt min/avg/max/mdev = 0.295/0.364/0.420/0.045 ms

```

Se observa en el log:

```

root@pru-VirtualBox:/home/pru# tail -f /var/log/suricata/fast.log
03/31/2022-10:05:07.700326  [**] [1:100001:1] ICMP Packet found [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.1.41:8 -> 192.168.1.42:0
03/31/2022-10:05:07.700391  [**] [1:100001:1] ICMP Packet found [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.1.42:0 -> 192.168.1.41:0

```

NOTA:

Si quiero ejecutar suricata con unas reglas determinadas, escribimos el comando:

```
suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/myrules -i enp0s3
```

```
root@pru-VirtualBox:/home/pru# suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/myrules -i enp0s3
31/3/2022 -- 10:22:18 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
31/3/2022 -- 10:22:18 - <Warning> - [ERRCODE: SC_ERR_NO_RULES(42)] - No rule files match the pattern /etc/suricata/
rules.myrules
31/3/2022 -- 10:22:18 - <Notice> - all 1 packet processing threads, 4 management threads initialized, engine starte
d.
```

Indicamos el fichero de reglas.

De esta manera, no es necesario modificar el fichero de configuración y se pueden hacer pruebas con otras configuraciones

6. Detectar conexiones a Facebook.

Vamos a crear una regla para detectar conexiones a facebook.com:

```
alert tcp any any -> any any (msg: "Facebook está bloqueado"; content:"facebook"; sid:2; rev:1;)
```

```
root@pru-VirtualBox: /etc/suricata
root@pru-VirtualBox:/etc/suricata# curl -i facebook.com
```

```
root@pru-VirtualBox:/var/log/suricata# tail -f fast.log
03/31/2022-18:15:02.998265 [wDrop] [**] [1:2:1] facebook está bloqueado [**] [Classification: Pote
ntial Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.2.15:53790 -> 31.13.83.36:80
03/31/2022-18:15:02.999013 [wDrop] [**] [1:2:1] facebook está bloqueado [**] [Classification: Pote
ntial Corporate Privacy Violation] [Priority: 1] {TCP} 31.13.83.36:80 -> 10.0.2.15:53790
```

7. Detectar peticiones GET de Http.

Vamos a crear una regla para detectar peticiones GET de http:

```
alert http $HOME_NET any -> $EXTERNAL_NET 80 (msg: "Petición GET"; flow:established, to_server; content:"GET";http_method; sid:3; rev:1;)
```

Para ello, vamos a hacer una petición con **curl**:

```
curl -i www.gobiernodecanarias.org
```

```
pru@pru-VirtualBox:~$ curl -i www.gobiernodecanarias.org
HTTP/1.1 302 Found
Date: Fri, 05 May 2023 04:55:56 GMT
Server: Apache
Location: http://www.gobiernodecanarias.org/principal/
Content-Length: 228
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.gobiernodecanarias.org/principal/">here</a>.</p>
</body></html>
```

```
root@pru-VirtualBox:/var/log/suricata# tail -f fast.log
03/31/2022-18:29:46.825241  [**] [1:3:2] petición GET [**] [Classification: (null)] [Priority: 3] {
TCP} 10.0.2.15:34828 -> 93.188.136.129:80
```


8. Detectar peticiones conexiones SSH.

Vamos a crear una regla para detectar conexiones SSH:

```
alert tcp any any -> any 22 (msg: "Conexión SSH detectada"; flow:to_server; app-layer-protocol:ssh; sid:4; rev:1;)
```

```
(pru@Newkali)-[~]  
$ ssh 10.0.2.15  
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.  
ED25519 key fingerprint is SHA256:E5YBaUmVYXhDvz+mCZVrR6tXS+vvpEWc/3H9nkB1Un0.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? █
```

```
root@pru-VirtualBox:/var/log/suricata# tail -f fast.log  
03/31/2022-18:48:29.976581  [**] [1:3:2] peticion GET [**] [Classification: (null)] [Priority: 3] {  
TCP} 10.0.2.9:49172 -> 23.210.45.45:80  
03/31/2022-18:49:58.391505  [**] [1:4:1] Conexion SSH detectada [**] [Classification: (null)] [Prio  
rity: 3] {TCP} 10.0.2.7:43512 -> 10.0.2.15:22
```

9. Detectar peticiones un escaneo de puertos.

Vamos a crear una regla para detectar escaneos de puertos:

```
alert tcp any any -> any any (msg:"Nmap FIN Scan"; flags:F; sid:5; rev:1;)
alert tcp any any -> any any (msg:"Nmap NULL Scan"; flags:0; sid:6; rev:1;)
alert udp any any -> any any (msg:"Nmap UDP Scan"; sid:7; rev:1;)
alert tcp any any -> any any (msg:"Nmap XMAS Tree Scan"; flags:FPU; sid:8; rev:1;)
alert tcp any any -> any any (msg:"Nmap TCP Tree Scan"; sid:9; rev:1;)
alert tcp any any -> any any (msg:"Nmap Ping Sweep Scan"; dsize:0; sid:10; rev:1;)
```

Probar con:

```
nmap -sF
nmap -sN
nmap -sU
nmap -sX
nmap -sT
```

```
07/27/2022-05:44:16.172426  [**] [1:7:1] Nmap UDP Scan [**] [Classification: (null)] [Priority: 3] {UDP} 10.0.2.22:42370 -> 80.58.61.250:53
07/27/2022-05:44:16.205474  [**] [1:7:1] Nmap UDP Scan [**] [Classification: (null)] [Priority: 3] {UDP} 80.58.61.250:53 -> 10.0.2.22:42370
07/27/2022-05:44:16.206572  [**] [1:9:1] Nmap TCP Tree Scan [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.2.22:42886 -> 34.120.208.123:443
07/27/2022-05:44:16.206572  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.2.22:42886 -> 34.120.208.123:443
07/27/2022-05:44:16.237011  [**] [1:9:1] Nmap TCP Tree Scan [**] [Classification: (null)] [Priority: 3] {TCP} 34.120.208.123:443 -> 10.0.2.22:42886
07/27/2022-05:44:16.237011  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 34.120.208.123:443 -> 10.0.2.22:42886
07/27/2022-05:44:16.237037  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.2.22:42886 -> 34.120.208.123:443
07/27/2022-05:44:16.274827  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.2.22:42886 -> 34.120.208.123:443
07/27/2022-05:44:16.274842  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.2.22:42886 -> 34.120.208.123:443
07/27/2022-05:44:16.274937  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 10.0.2.22:42886 -> 34.120.208.123:443
07/27/2022-05:44:16.276421  [**] [1:10:1] Nmap Ping Sweep Scan [**] [Classification: (null)] [Priority: 3] {TCP} 34.120.208.123:443 -> 10.0.2.22:42886
```


10. Actuar como IPS

Tenemos que **indicar que el tráfico pase por Suricata**. Para ello, tenemos que **comprobar que Suricata tiene NFQ habilitado**, Para comprobarlo, hacemos:

```
Suricata --build-info
```

```
pru@pru-VirtualBox:~$ suricata --build-info
This is Suricata version 6.0.11 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1
SIMD support: none
Atomic intrinsics: 1 2 4 8 byte(s)
64-bits, Little-endian architecture
GCC version 9.4.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.43, linked against LibHTP v0.5.43

Suricata Configuration:
  AF_PACKET support:          yes
  eBPF support:               no
  XDP support:                no
  PF_RING support:            no
  NFQueue support:            yes
  NFLOG support:              no
  IPFW support:               no
  Netmap support:             no   using new api: no
  DAG enabled:                no
```

Vamos a **configurar Suricata como ips basado en host**. Por tanto, hemos de indicar **que el tráfico de entrada y de salida del host pase por Suricata**. Para ello, configuramos iptables:

```
sudo iptables -I INPUT -j NFQUEUE
sudo iptables -I OUTPUT -j NFQUEUE
```

Con el siguiente comando comprobamos que está funcionando:

```
sudo iptables -vnL
```

```
(root@kali)-[/home/kali]
# sudo iptables -I INPUT -j NFQUEUE

(root@kali)-[/home/kali]
# sudo iptables -I OUTPUT -j NFQUEUE

(root@kali)-[/home/kali]
# sudo iptables -vnL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source               destination           NFQUEUE num 0
    2   56 NFQUEUE    0    --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source               destination           NFQUEUE num 0
    7   616 NFQUEUE    0    --  *      *       0.0.0.0/0            0.0.0.0/0
```

Creamos el fichero my2.rules, le ponemos una regla para bloquear el acceso a Facebook:

```
drop tcp any any -> any any (msg: "Facebook está bloqueado"; content:"facebook"; sid:2;
rev:1;)
```

```
GNU nano 4.8 /etc/suricata/rules/my2.rules
drop tcp any any -> any any (msg: "Facebook está bloqueado"; content:"facebook"; sid:2 ; rev:1;)
```

Vamos a ejecutar suricata ejecutando el fichero my2.rules

```
Suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/my2.rules -q 0
```

Accedemos con un navegador a una página (Google.com, por ejemplo):



Ahora intentamos acceder a Facebook. El fichero fast.log muestra el bloqueo

```
pru@pru-VirtualBox:~$ tail -f /var/log/suricata/fast.log
05/07/2023-17:15:06.567409 [Drop] [**] [1:2:1] Facebook está bloqueado [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.45:38876 -> 157.240.5.35:443
```

Y en el navegador observamos como no se puede acceder:



Vamos a **bloquear el acceso a ssh**:

```
drop tcp any any -> any 22 (msg: "Conexión SSH detectada"; flow:to_server; app-layer-protocol:ssh; sid:2; rev:1;)
```

Si no está bloqueada la conexión, solo se podrá acceder a SSH con suricata conectado, ya que ahora, todo el tráfico pasa por **Suricata**:

```
(kali@kali)-[~]  
$ ssh Administrator@192.168.1.44  
Administrator@192.168.1.44's password:  
Last login: Thu Apr 11 21:25:35 2024 from 192.168.1.34  
-sh-4.3$ ^C  
-sh-4.3$ exit  
logout  
Connection to 192.168.1.44 closed.
```

Una vez bloqueado SSH con la regla indicada, no se podrá conectar vía SSH.

Nota: Para resetear la configuración de iptables escribimos:

```
sudo iptables -F
```

y volveremos a tener conexión a Internet.

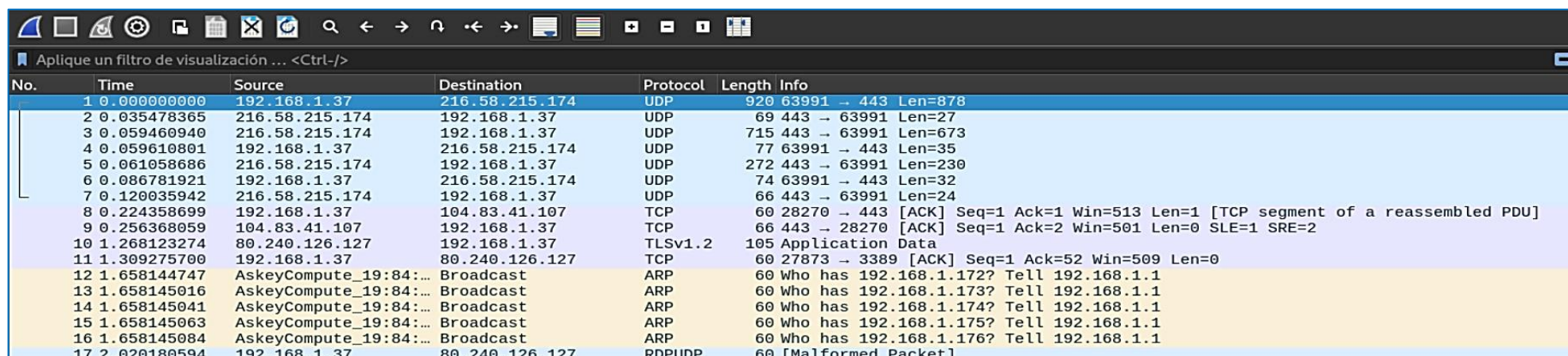
11. Detectar peticiones DNS a dominio

Para detectar una petición a un dominio determinado (en nuestro caso, a whatsapp.com, Facebook.com e Instagram.com). En este caso vamos a hacerlo analizando las capturas hechas en un fichero generado o wireshark (guardado en el fichero **captura.pcap**).

Para ello, ponemos las siguientes reglas:

```
alert dns $HOME_NET any -> $EXTERNAL_NET 53 (msg: "Petición DNS a WhatsApp detectada";  
dns_query; content:"whatsapp.com"; nocase; sid:14; rev:1;)  
alert dns $HOME_NET any -> $EXTERNAL_NET 53 (msg: "Petición DNS a Facebook detectada";  
dns_query; content:"facebook.com"; nocase; sid:15; rev:1;)  
alert dns $HOME_NET any -> $EXTERNAL_NET 53 (msg: "Petición DNS a instagram detectada";  
dns_query; content:"Instagram.com"; nocase; sid:16; rev:1;)  
alert dns $HOME_NET any -> $EXTERNAL_NET 53 (msg: "Petición DNS a TikTok detectada";  
dns_query; content:"tiktok.com"; nocase; sid:16; rev:1;)
```

Tenemos una captura realizada con wireshark (en el fichero **captura.pcap**):



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.37	216.58.215.174	UDP	920	63991 → 443 Len=878
2	0.035478365	216.58.215.174	192.168.1.37	UDP	69	443 → 63991 Len=27
3	0.059460940	216.58.215.174	192.168.1.37	UDP	715	443 → 63991 Len=673
4	0.059610801	192.168.1.37	216.58.215.174	UDP	77	63991 → 443 Len=35
5	0.061058686	216.58.215.174	192.168.1.37	UDP	272	443 → 63991 Len=230
6	0.086781921	192.168.1.37	216.58.215.174	UDP	74	63991 → 443 Len=32
7	0.120035942	216.58.215.174	192.168.1.37	UDP	66	443 → 63991 Len=24
8	0.224358699	192.168.1.37	104.83.41.107	TCP	60	28270 → 443 [ACK] Seq=1 Ack=1 Win=513 Len=1 [TCP segment of a reassembled PDU]
9	0.256368059	104.83.41.107	192.168.1.37	TCP	66	443 → 28270 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
10	1.268123274	80.240.126.127	192.168.1.37	TLSv1.2	105	Application Data
11	1.309275700	192.168.1.37	80.240.126.127	TCP	60	27873 → 3389 [ACK] Seq=1 Ack=52 Win=509 Len=0
12	1.658144747	AskeyCompute_19:84:...	Broadcast	ARP	60	Who has 192.168.1.172? Tell 192.168.1.1
13	1.658145016	AskeyCompute_19:84:...	Broadcast	ARP	60	Who has 192.168.1.173? Tell 192.168.1.1
14	1.658145041	AskeyCompute_19:84:...	Broadcast	ARP	60	Who has 192.168.1.174? Tell 192.168.1.1
15	1.658145063	AskeyCompute_19:84:...	Broadcast	ARP	60	Who has 192.168.1.175? Tell 192.168.1.1
16	1.658145084	AskeyCompute_19:84:...	Broadcast	ARP	60	Who has 192.168.1.176? Tell 192.168.1.1
17	2.020180594	192.168.1.37	80.240.126.127	RDPUDP	60	[Malformed Packet]

Ejecutamos suricata con las siguientes opciones:

```
suricata -r /home/kali/captura.pcap -c /etc/suricata/suricata.yaml -s  
/etc/suricata/rules/my.rules -l /home/kali
```

```
(root@kali)-[/home/pru]  
# suricata -r /home/pru/captura.pcap -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/my3.rules -l /home/pru  
i: suricata: This is Suricata version 7.0.3 RELEASE running in USER mode  
i: threads: Threads created → RX: 1 W: 4 FM: 1 FR: 1 Engine started.  
i: suricata: Signal Received. Stopping engine.  
i: pcap: read 1 file, 9787 packets, 12021008 bytes
```

El resultado lo devuelve en el fichero **fast.log** (en el directorio /home/Kali):

```
04/11/2024-23:15:12.115765 [**] [1:1:1] Detectada petición DNS a Whatsapp [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:60473 → 80.58.61.250:53  
04/11/2024-23:15:12.115783 [**] [1:1:1] Detectada petición DNS a Whatsapp [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:60473 → 80.58.61.250:53  
04/11/2024-23:15:21.556956 [**] [1:16:1] Petición DNS a instagram detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:48713 → 80.58.61.250:53  
04/11/2024-23:15:21.964268 [**] [1:16:1] Petición DNS a instagram detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:48276 → 80.58.61.250:53  
04/11/2024-23:15:21.556975 [**] [1:16:1] Petición DNS a instagram detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:48713 → 80.58.61.250:53  
04/11/2024-23:15:21.964288 [**] [1:16:1] Petición DNS a instagram detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:48276 → 80.58.61.250:53  
04/11/2024-23:15:23.218753 [**] [1:15:1] Petición DNS a Facebook detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:33216 → 80.58.61.250:53  
04/11/2024-23:15:23.218788 [**] [1:15:1] Petición DNS a Facebook detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:33216 → 80.58.61.250:53  
04/11/2024-23:15:31.003061 [**] [1:15:1] Petición DNS a Facebook detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:50119 → 80.58.61.250:53  
04/11/2024-23:15:31.452347 [**] [1:15:1] Petición DNS a Facebook detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:48927 → 80.58.61.250:53  
04/11/2024-23:15:31.452382 [**] [1:15:1] Petición DNS a Facebook detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:48927 → 80.58.61.250:53  
04/11/2024-23:15:31.003082 [**] [1:15:1] Petición DNS a Facebook detectada [**] [Classification: (null)] [Priority: 3] {UDP} 192.168.1.34:50119 → 80.58.61.250:53
```