

### Configuración de la práctica y de SNORT 3

1. Cambiar el nombre del usuario 'kali' a enrique:

Para que en las capturas se vea mi nombre he cambiado el archivo hostname de /etc/. Con esta captura se puede ver esto y la versión de snort que voy a usar para la práctica.

```
(kali@enrique)-[~]
$ snort -v

o")~ Snort++ 3.1.82.0

Network Policy : policy id 0 :
Inspection Policy : policy id 0 :
pcap DAQ configured to passive.

host_cache
  memcap: 33554432 bytes

Snort successfully validated the configuration (with 0 warnings).
o")~ Snort exiting
```

2. El primer paso ahora es identificar cual es mi ip y activar el modo promiscuo desde la terminal. (Como he realizado la práctica en días diferentes la ip varía a lo largo de esta memoria)

```
(root@enrique)-[/home/kali]
# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default 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 noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc fq
_codel state UP group default qlen 1000
    link/ether 08:00:27:ad:25:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.39/24 brd 192.168.1.255 scope global dynamic nopr
efixroute eth0
        valid_lft 42683sec preferred_lft 42683sec
    inet6 fe80::56d0:4d2d:4333:977b/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

```
(root@enrique)-[/home/kali]
# ip link set eth0 promisc on
```

3. Ahora he puesto la dirección ip de mi red en el archivo de configuración snort.lua

```
root@enrique: /etc/snort x  kali@enrique: ~ x
GNU nano 8.2 snort.lua *

-- Snort++ configuration

-- there are over 200 modules available to tune your policy.
-- many can be used with defaults w/o any explicit configuration.
-- use this conf as a template for your specific configuration.

-- 1. configure defaults
-- 2. configure inspection
-- 3. configure bindings
-- 4. configure performance
-- 5. configure detection
-- 6. configure filters
-- 7. configure outputs
-- 8. configure tweaks

-- 1. configure defaults

-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '192.168.1.39/24'

-- set up the external network addresses.
-- (leave as "any" in most situations)
EXTERNAL_NET = 'any'

include 'snort_defaults.lua'

-- 2. configure inspection

^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

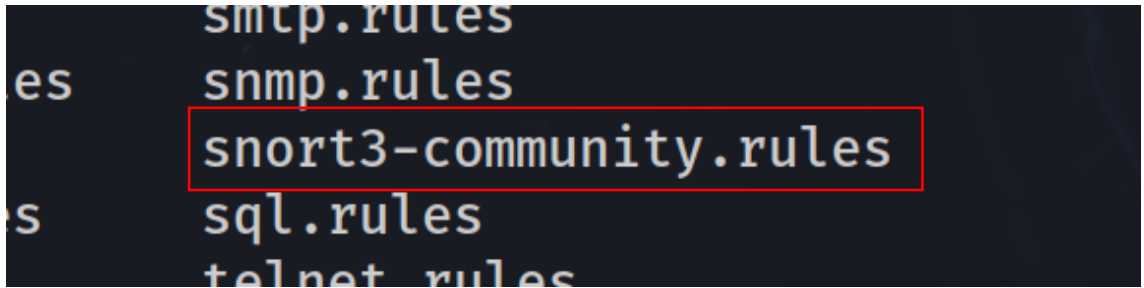
4. Y también he añadido la ruta donde se encuentran las reglas preestablecidas

```
-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '192.168.1.39/24'
RULE_PATH = '/etc/snort/rules'
```

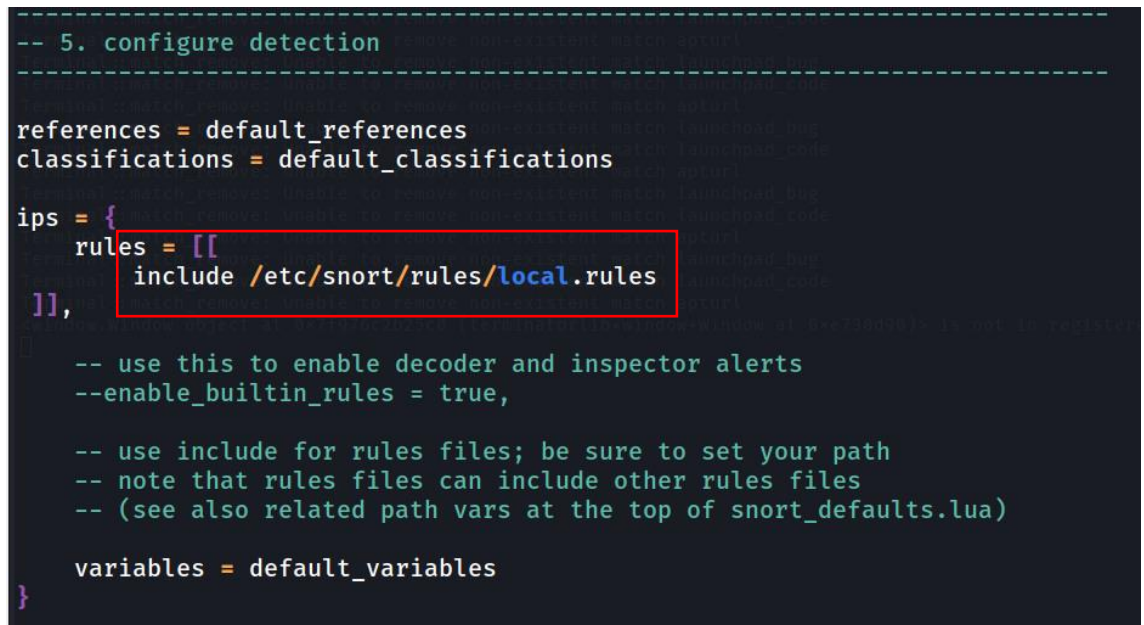
5. Ahora añado las reglas que vienen predeterminadas en snort para poder probarlas y el local.rules que he creado yo.

```
ips = {
  rules = [[
    include /etc/snort/rules/attack-responses.rules
    include /etc/snort/rules/backdoor.rules
    include /etc/snort/rules/bad-traffic.rules
    include /etc/snort/rules/chat.rules
    include /etc/snort/rules/community-bot.rules
    include /etc/snort/rules/community-deleted.rules
```

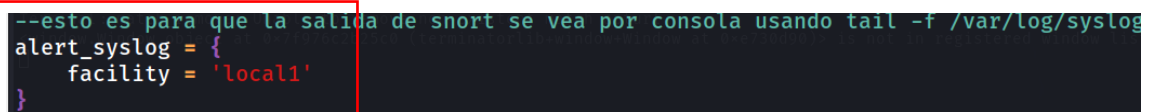
6. He descargado y descomprimido las reglas de la comunidad de snort v3 para poder hacer las pruebas iniciales.



7. Una vez he probado que todo funciona y se generan las alertas correctamente he configurado snort para que solo aplique las reglas de mi archivo de reglas locales.



8. Las reglas que he creado son las siguientes, y para detectar las alertas lo hago mediante el syslog con el comando tail ya que así es como lo he configurado en el snort.lua.



## 9. Mis cinco reglas:

### Detección de un escaneo de puertos:

```
alert tcp any any -> any any (msg:"ALERTA: Escaneo de puertos TCP detectado";
flags:S; threshold:type threshold, track by_src, count 20, seconds 10; sid:1000001;
rev:1;)
```

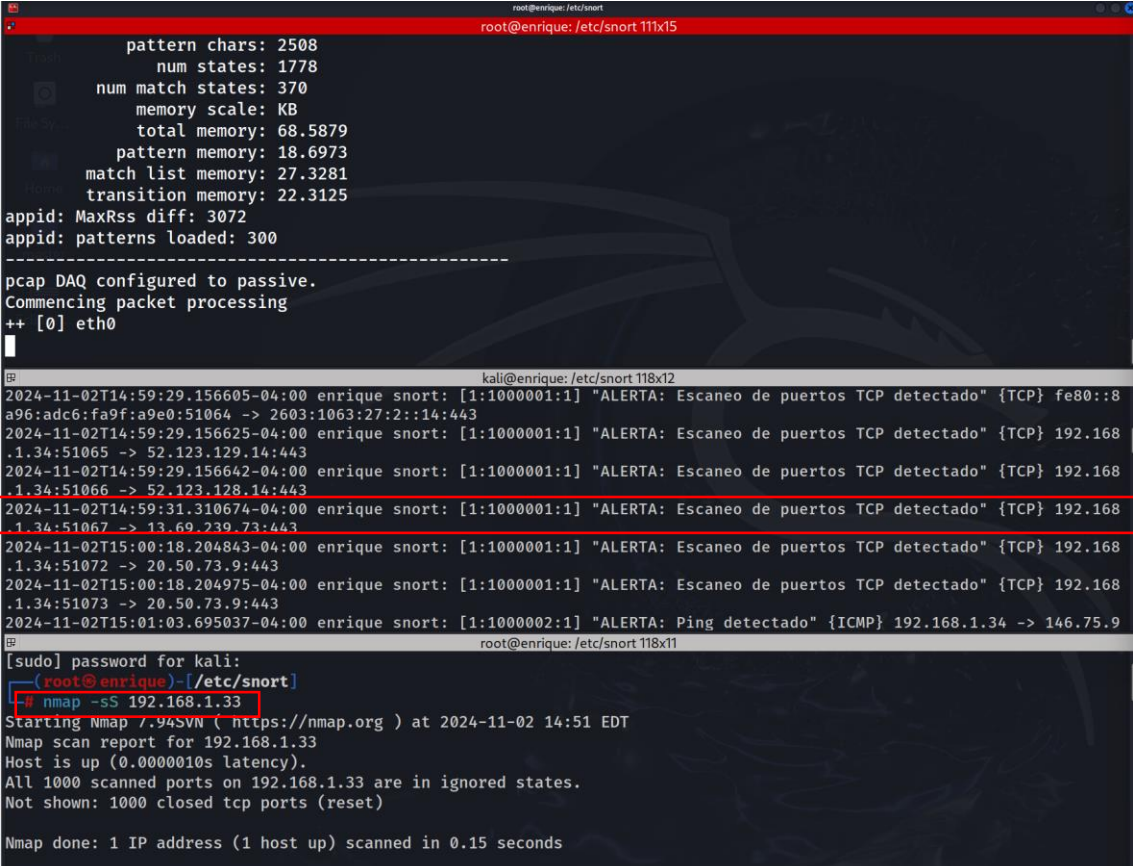
### Explicación:

Si desde una misma IP de origen se envían 20 paquetes SYN en 10 segundos (independientemente del puerto de destino), se genera una alerta.

**type threshold** define un límite para evitar que cada paquete coincidente genere una alerta individual. Esto es útil para detectar patrones como escaneos de puertos. **track by\_src** aplica el umbral basado en la dirección IP de origen. Esto significa que cada dirección IP que envíe paquetes SYN se rastrea por separado. **count 20** hace que la alerta se genere si se detectan 20 paquetes SYN desde la misma IP de origen dentro del período definido. **seconds 10** es el período de tiempo. Si se detectan paquetes SYN desde una misma IP en 10 segundos, se generará la alerta.

### Comprobación:

Con nmap lanzo un escaneo de red con el parámetro -sS que sirve para detectar puertos TCP enviando paquetes SYN para establecer conexiones, pero sin completar el "handshake" de TCP lo que hace que sea menos detectable.



```
root@enrique: /etc/snort
root@enrique: /etc/snort 11x15

pattern chars: 2508
num states: 1778
num match states: 370
memory scale: KB
total memory: 68.5879
pattern memory: 18.6973
match list memory: 27.3281
transition memory: 22.3125
appid: MaxRss diff: 3072
appid: patterns loaded: 300
-----
pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0

kali@enrique: /etc/snort 11x12
2024-11-02T14:59:29.156605-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} fe80::8
a96:adc6:fa9f:a9e0:51064 -> 2603:1063:27:2::14:443
2024-11-02T14:59:29.156625-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51065 -> 52.123.129.14:443
2024-11-02T14:59:29.156642-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51066 -> 52.123.128.14:443
2024-11-02T14:59:31.310674-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51067 -> 13.69.239.73:443
2024-11-02T15:00:18.204843-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51072 -> 20.50.73.9:443
2024-11-02T15:00:18.204975-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51073 -> 20.50.73.9:443
2024-11-02T15:01:03.695037-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
root@enrique: /etc/snort 11x11

[sudo] password for kali:
(root@enrique) - [/etc/snort]
# nmap -sS 192.168.1.33
Starting Nmap .94SVN ( https://nmap.org ) at 2024-11-02 14:51 EDT
Nmap scan report for 192.168.1.33
Host is up (0.0000010s latency).
All 1000 scanned ports on 192.168.1.33 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds
```



**Detección de un ping ICMP:**

alert icmp any any -> any any (msg:"ALERTA: Ping detectado"; itype:8; sid:1000002; rev:1;)

**Explicación:**

La regla detecta cualquier paquete ICMP enviado desde cualquier origen a cualquier destino.

itype:8: Especifica que la regla debe capturar solo paquetes de tipo ICMP Echo Request (ping).

**Comprobación:**

Lanzo un ping a mi dirección ip 192.168.1.33

```

root@enrique: /etc/snort
transition memory: 22.3125
appid: MaxRss diff: 3072
appid: patterns loaded: 300
-----
pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0

kali@enrique: /etc/snort 118x14
0.172
2024-11-02T15:01:03.695380-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51074 -> 20.189.173.15:443
2024-11-02T15:01:03.695392-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
0.172
2024-11-02T15:01:03.695402-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
0.172
2024-11-02T15:01:07.724524-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
0.172
2024-11-02T15:01:07.724587-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
0.172
2024-11-02T15:01:07.724617-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
0.172
2024-11-02T15:01:07.724629-04:00 enrique snort: [1:1000002:1] "ALERTA: Ping detectado" {ICMP} 192.168.1.34 -> 146.75.9
0.172
root@enrique: /etc/snort 118x17
Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds

(root@enrique)-[/etc/snort]
# ping 192.168.1.33
PING 192.168.1.33 (192.168.1.33) 56(84) bytes of data.
64 bytes from 192.168.1.33: icmp_seq=1 ttl=64 time=0.012 ms
64 bytes from 192.168.1.33: icmp_seq=2 ttl=64 time=0.020 ms
64 bytes from 192.168.1.33: icmp_seq=3 ttl=64 time=0.022 ms
64 bytes from 192.168.1.33: icmp_seq=4 ttl=64 time=0.024 ms
64 bytes from 192.168.1.33: icmp_seq=5 ttl=64 time=0.018 ms
64 bytes from 192.168.1.33: icmp_seq=6 ttl=64 time=0.023 ms
64 bytes from 192.168.1.33: icmp_seq=7 ttl=64 time=0.022 ms
64 bytes from 192.168.1.33: icmp_seq=8 ttl=64 time=0.023 ms
64 bytes from 192.168.1.33: icmp_seq=9 ttl=64 time=0.023 ms
64 bytes from 192.168.1.33: icmp_seq=10 ttl=64 time=0.024 ms
64 bytes from 192.168.1.33: icmp_seq=11 ttl=64 time=0.024 ms
64 bytes from 192.168.1.33: icmp_seq=12 ttl=64 time=0.023 ms

```

**Detección de una conexión HTTP:**

```
alert tcp any any -> any 80 (msg:"ALERTA: Conexión HTTP detectada";  
sid:1000004; rev:1;)
```

**Explicación:**

La regla detecta cualquier conexión TCP hacia el puerto 80, que es el puerto estándar para HTTP y genera una alerta cuando se intenta establecer una conexión a un servidor web en ese puerto.

**Comprobación:**

He usado nmap para realizar un escaneo específico sobre el puerto 80 con el comando `nmap -p 80 192.168.1.33`. Aunque el puerto apareció como cerrado en los resultados del escaneo, el tráfico TCP generado por la herramienta fue suficiente para activar la regla configurada en Snort. Esto confirma que la regla está funcionando correctamente y que es capaz de detectar intentos de conexión al puerto 80, incluso si no hay un servicio escuchando en él.

```
root@enrique: /etc/snort
transition memory: 22.3125
appid: MaxRss diff: 3072
appid: patterns loaded: 300
-----
pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0

kali@enrique: /etc/snort 118x14
.1.34:51177 -> 192.0.77.2:443
2024-11-02T15:14:16.676678-04:00 enrique snort: [1:1000004:1] "ALERTA: Conexión HTTP detectada" {TCP} 192.168.1.34:511
78 -> 185.43.182.113:80
2024-11-02T15:14:16.676743-04:00 enrique snort: [1:1000001:1] "ALERTA: Escaneo de puertos TCP detectado" {TCP} 192.168
.1.34:51178 -> 185.43.182.113:80
2024-11-02T15:14:16.676757-04:00 enrique snort: [1:1000004:1] "ALERTA: Conexión HTTP detectada" {TCP} 192.168.1.34:511
78 -> 185.43.182.113:80
2024-11-02T15:14:16.676766-04:00 enrique snort: [1:1000004:1] "ALERTA: Conexión HTTP detectada" {TCP} 192.168.1.34:511
78 -> 185.43.182.113:80
2024-11-02T15:14:18.839071-04:00 enrique snort: [1:1000004:1] "ALERTA: Conexión HTTP detectada" {TCP} 192.168.1.34:511
78 -> 185.43.182.113:80
2024-11-02T15:14:18.839135-04:00 enrique snort: [1:1000004:1] "ALERTA: Conexión HTTP detectada" {TCP} 192.168.1.34:511
78 -> 185.43.182.113:80
2024-11-02T15:14:18.839148-04:00 enrique snort: [1:1000004:1] "ALERTA: Conexión HTTP detectada" {TCP} 192.168.1.34:511
78 -> 185.43.182.113:80

root@enrique: /etc/snort 118x17
--(root@enrique)-[/etc/snort]
# nmap -p 80 192.168.1.33
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-02 14:54 EDT
Nmap scan report for 192.168.1.33
Host is up (0.000026s latency).

PORT      STATE SERVICE
80/tcp    closed http

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds

--(root@enrique)-[/etc/snort]
# nmap -p 80 192.168.1.33
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-02 15:04 EDT
Nmap scan report for 192.168.1.33
Host is up (0.000023s latency).
```

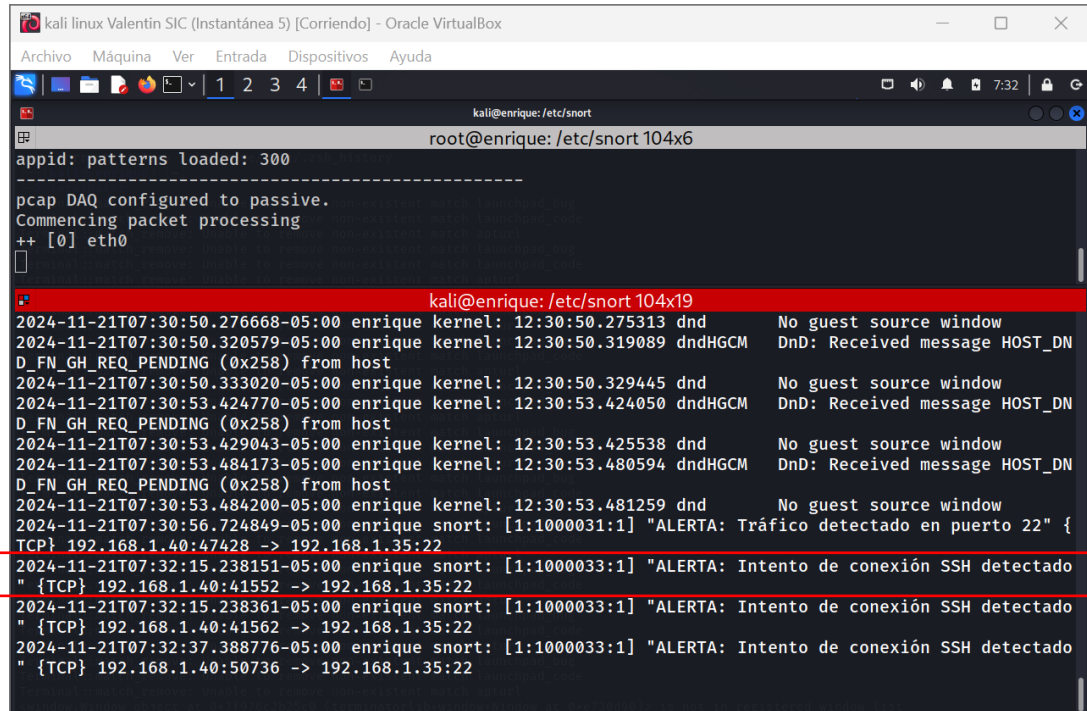


### Detección de una conexión SSH (puerto 22)

alert tcp any any -> any 22 (msg:"ALERTA: Intento de conexión SSH detectado";  
content:"SSH-"; sid:1000033; rev:1;)

### Explicación:

La regla captura cualquier intento de conexión SSH al puerto 22 al buscar la cadena "SSH-" en el tráfico TCP

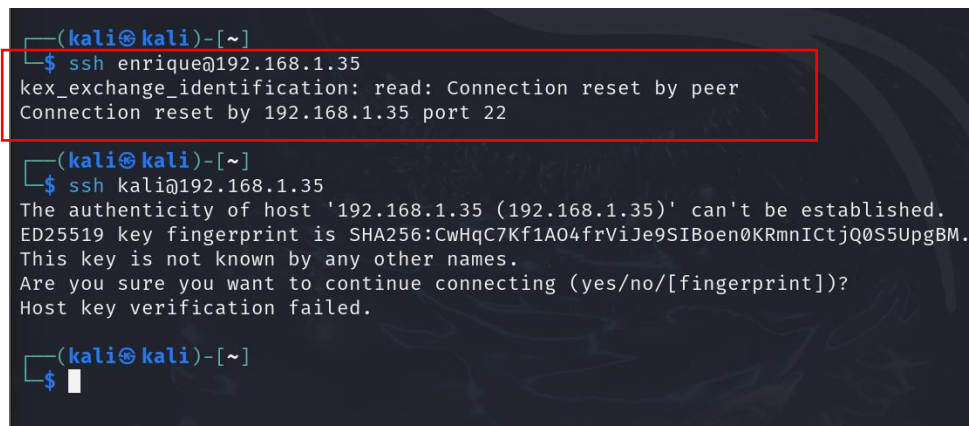


The screenshot shows a terminal window titled 'kali@enrique: /etc/snort' displaying the output of a snort rule engine. The log shows several network events, including kernel notifications and snort alerts. A red box highlights the following alert entries:

```
2024-11-21T07:30:56.724849-05:00 enrique snort: [1:1000033:1] "ALERTA: Tráfico detectado en puerto 22" {  
TCP: 192.168.1.40:47428 -> 192.168.1.35:22  
2024-11-21T07:32:15.238151-05:00 enrique snort: [1:1000033:1] "ALERTA: Intento de conexión SSH detectado  
{TCP} 192.168.1.40:41552 -> 192.168.1.35:22  
2024-11-21T07:32:15.238361-05:00 enrique snort: [1:1000033:1] "ALERTA: Intento de conexión SSH detectado  
{TCP} 192.168.1.40:41562 -> 192.168.1.35:22  
2024-11-21T07:32:37.388776-05:00 enrique snort: [1:1000033:1] "ALERTA: Intento de conexión SSH detectado  
{TCP} 192.168.1.40:50736 -> 192.168.1.35:22
```

### Comprobación:

Inicié un intento de conexión SSH desde una máquina cliente hacia un servidor SSH ubicado en la IP objetivo. Aunque la conexión no fue establecida, el tráfico inicial, que incluye el flag SYN y la cadena "SSH-", fue suficiente para activar la alerta en el puerto 22.



The screenshot shows a terminal window titled '(kali@kali)-[~]' with the following commands and output:

```
(kali@kali)-[~]  
$ ssh enrique@192.168.1.35  
kex_exchange_identification: read: Connection reset by peer  
Connection reset by 192.168.1.35 port 22  
  
(kali@kali)-[~]  
$ ssh kali@192.168.1.35  
The authenticity of host '192.168.1.35 (192.168.1.35)' can't be established.  
ED25519 key fingerprint is SHA256:CwHqC7Kf1AO4frViJe9SIBoen0KRmnICtjQ0S5Up9BM.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])?  
Host key verification failed.  
  
(kali@kali)-[~]  
$
```



## SURICATA

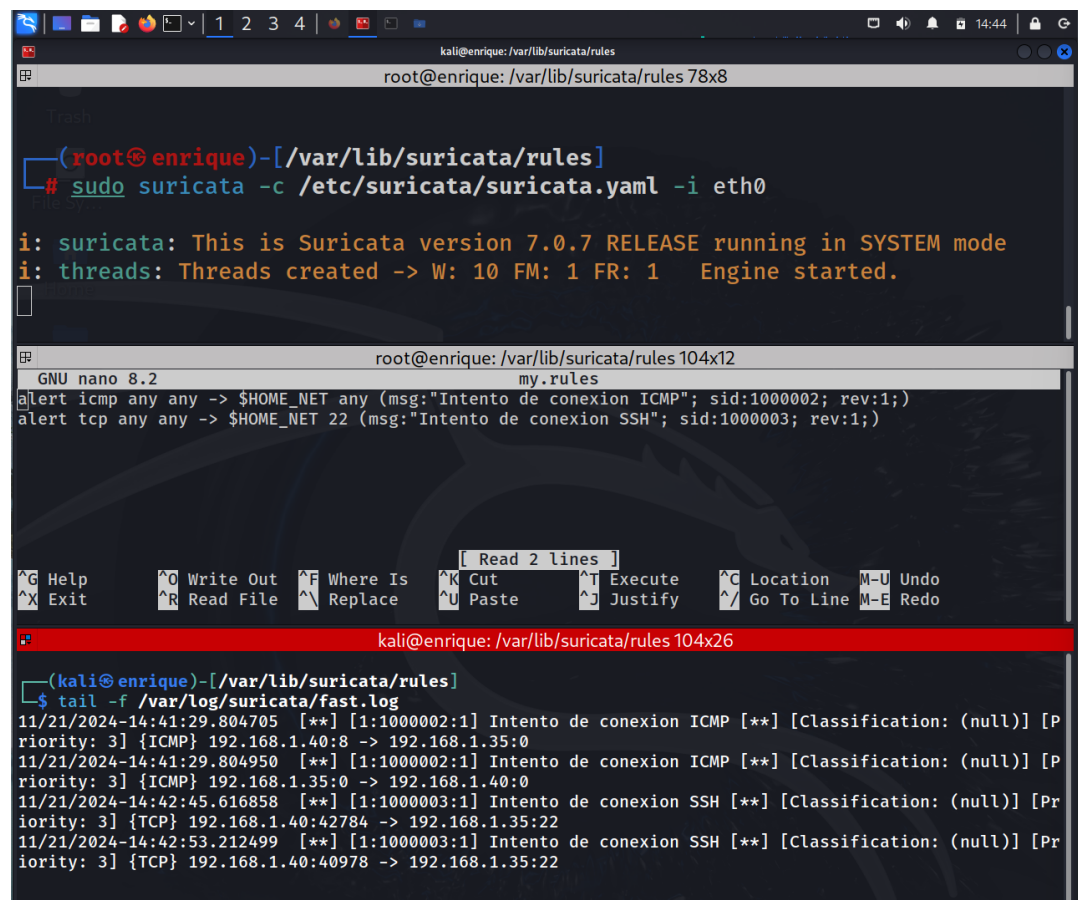
En este caso, he creado dos reglas básicas en Suricata para detectar tráfico:

```
alert icmp any any -> $HOME_NET any (msg:"Intento de conexion ICMP";  
sid:1000001; rev:1;)
```

Esta regla genera una alerta cuando detecta paquetes ICMP como los usados en un ping, dirigidos a cualquier dirección dentro de mi red local (\$HOME\_NET).

```
alert tcp any any -> $HOME_NET 22 (msg:"Intento de conexion SSH"; sid:1000002;  
rev:1;)
```

La segunda regla genera una alerta cuando se detecta tráfico TCP hacia el puerto 22 usado por SSH



The screenshot shows a Kali Linux terminal window with two panes. The top pane shows the execution of the Suricata daemon. The bottom pane shows the contents of the fast.log file, which contains alerts for ICMP and SSH traffic.

```
kali@enrique: /var/lib/suricata/rules
root@enrique: /var/lib/suricata/rules 78x8

(kali@enrique)-[/var/lib/suricata/rules]
# sudo suricata -c /etc/suricata/suricata.yaml -i eth0

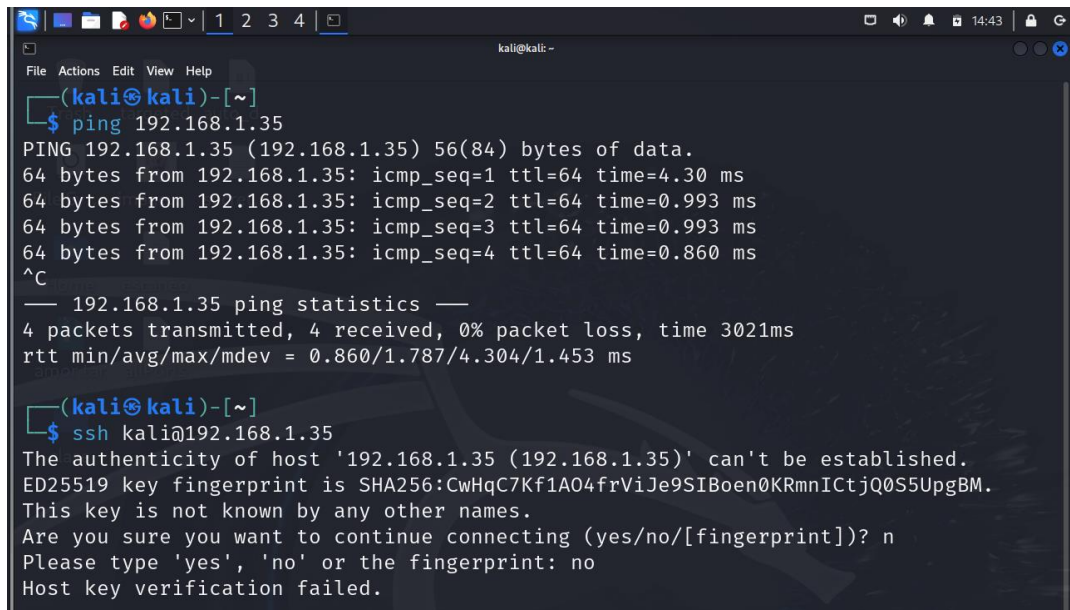
i: suricata: This is Suricata version 7.0.7 RELEASE running in SYSTEM mode
i: threads: Threads created -> W: 10 FM: 1 FR: 1 Engine started.

GNU nano 8.2 root@enrique: /var/lib/suricata/rules 104x12
my.rules
alert icmp any any -> $HOME_NET any (msg:"Intento de conexion ICMP"; sid:1000002; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"Intento de conexion SSH"; sid:1000003; rev:1;)

(kali@enrique)-[/var/lib/suricata/rules]
$ tail -f /var/log/suricata/fast.log
11/21/2024-14:41:29.804705  [**] [1:1000002:1] Intento de conexion ICMP [**] [Classification: (null)] [P
riority: 3] {ICMP} 192.168.1.40:8 -> 192.168.1.35:0
11/21/2024-14:41:29.804950  [**] [1:1000002:1] Intento de conexion ICMP [**] [Classification: (null)] [P
riority: 3] {ICMP} 192.168.1.35:0 -> 192.168.1.40:0
11/21/2024-14:42:45.616858  [**] [1:1000003:1] Intento de conexion SSH [**] [Classification: (null)] [Pr
iority: 3] {TCP} 192.168.1.40:42784 -> 192.168.1.35:22
11/21/2024-14:42:53.212499  [**] [1:1000003:1] Intento de conexion SSH [**] [Classification: (null)] [Pr
iority: 3] {TCP} 192.168.1.40:40978 -> 192.168.1.35:22
```

Ambas reglas están funcionando correctamente, como se ve en la captura, y generan las alertas esperadas en el archivo fast.log.

Para comprobar las dos reglas he realizado desde otra máquina virtual tanto un ping como un intento de conexión por SSH

A screenshot of a Kali Linux terminal window. The window title is 'kali@kali: ~'. The terminal shows a user prompt '(kali@kali)-[~]' followed by a '\$' prompt. The user enters 'ping 192.168.1.35'. The output shows a successful ping with 4 packets transmitted and received, 0% packet loss, and a time of 3021ms. The user then enters 'ssh kali@192.168.1.35'. The output shows a warning about the authenticity of the host, a key fingerprint, and a prompt to continue connecting. The user enters 'no', and the output shows 'Host key verification failed.'

```
kali@kali: ~
(kali@kali)-[~]
$ ping 192.168.1.35
PING 192.168.1.35 (192.168.1.35) 56(84) bytes of data.
64 bytes from 192.168.1.35: icmp_seq=1 ttl=64 time=4.30 ms
64 bytes from 192.168.1.35: icmp_seq=2 ttl=64 time=0.993 ms
64 bytes from 192.168.1.35: icmp_seq=3 ttl=64 time=0.993 ms
64 bytes from 192.168.1.35: icmp_seq=4 ttl=64 time=0.860 ms
^C
— 192.168.1.35 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3021ms
rtt min/avg/max/mdev = 0.860/1.787/4.304/1.453 ms

(kali@kali)-[~]
$ ssh kali@192.168.1.35
The authenticity of host '192.168.1.35 (192.168.1.35)' can't be established.
ED25519 key fingerprint is SHA256:CwHqC7Kf1A04frViJe9SIBoen0KRmnICtjQ0S5UpqBM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? n
Please type 'yes', 'no' or the fingerprint: no
Host key verification failed.
```

### Diferencias entre suricata y snort:

**Sintaxis de reglas:** Aunque las reglas de suricata son similares a las de snort, son más estrictas en cuanto al uso de variables. Por ejemplo, en suricata he tenido varios errores al principio ya que es obligatorio definir claramente las direcciones (->) y usar variables como \$HOME\_NET, mientras que en snort es más flexible o menos estricto en ciertas cosas.

**Archivo de configuración:** Suricata utiliza un archivo en formato YAML suricata.yaml, mientras que snort usa un archivo más clásico snort.conf. En suricata, la estructura es más organizada, con bloques dedicados para las variables de red, los módulos y los logs, lo que facilita mucho encontrar rápido las cosas y cambiarlas.

**Manejo de logs:** Las alertas se guardan automáticamente en fast.log, pero también puede generar registros en formato JSON, que son muy buena opción para integrarlo con herramientas como la vista en clase Elastic Search. En snort, los logs son más simples y requieren herramientas externas como Barnyard2 para exportarlos a otros sistemas.

Además, suricata está diseñado para utilizar múltiples núcleos del procesador, lo que lo hace más eficiente en redes con mucho tráfico, mientras que snort trabaja en un solo núcleo, por lo que en mi opinión, si tuviese que quedarme con uno elegiría a suricata para implementar en un proyecto o en una empresa.