Universidad de Alcalá

Escuela Politécnica Superior

Sistema de Detección de Intrusiones
Snort + Suricata

Sistemas de Información para la Ciberseguridad

Autora:
Gisela Alascio del Moral

21 de noviembre de 2025

Índice

1. Tramo A — Snort.

1.1. Instalación y verificación de la interfaz de escucha.

El primer paso de este tramo consiste en la instalación y configuración básica de Snort en una máquina virtual de Ubuntu.

Antes de la instalación, se identifica la configuración de la máquina virtual con el comando `ip adrr`.

```
vboxuser@GiselaAlascio:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:9c:bb:3f brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 86331sec preferred_lft 86331sec
    inet6 fd17:625c:f037:2:d442:e7db:3959:2e61/64 scope global temporary dynamic

       valid_lft 86374sec preferred_lft 14374sec
    inet6 fd17:625c:f037:2:a00:27ff:fe9c:bb3f/64 scope global dynamic mngtmpaddr

       valid_lft 86374sec preferred_lft 14374sec
    inet6 fe80::a00:27ff:fe9c:bb3f/64 scope link
       valid_lft forever preferred_lft forever
```

Figura 1: Identificación de interfaces de red y direccionamiento IP mediante el comando `ip addr`.

Como se puede observar en la figura 1, se identifica la interfaz principal *Ethernet* (`enp0s3`) con la dirección IP 10.0.2.15/24. En base a esto, se define la `HOME_NET` como la red completa a proteger (10.0.2.0/24).

Además, se observa la interfaz de *loopback* (`lo`), que es relevante para la práctica, ya que se utilizará posteriormente para inyectar tráfico y validar una de las reglas.

Entonces, se procede a instalar Snort desde los repositorios oficiales de Ubuntu.

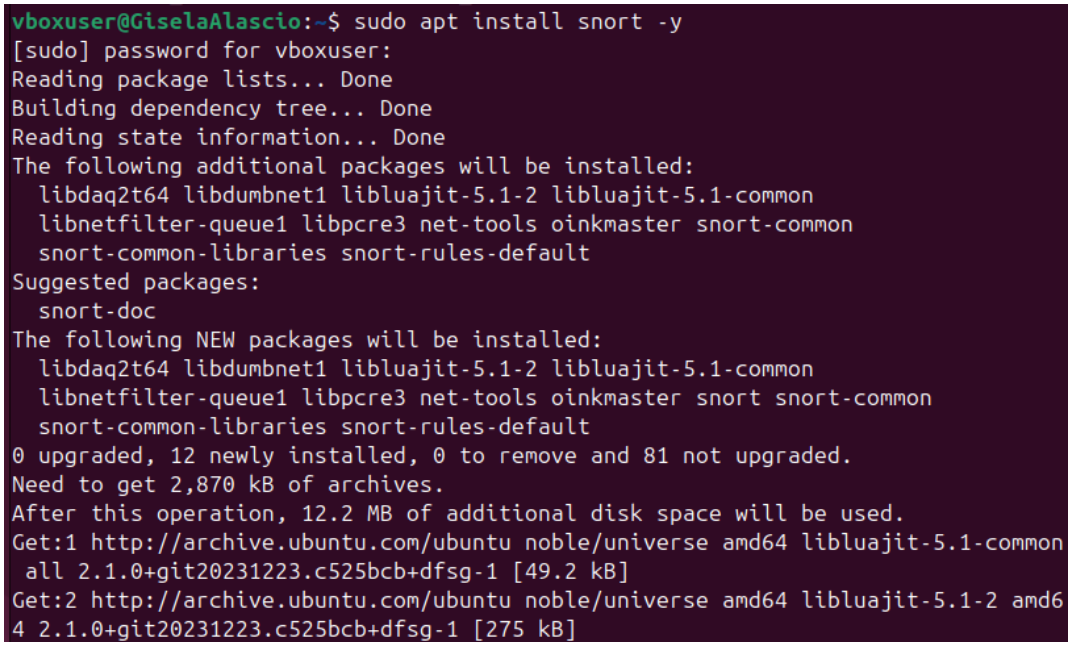

Figura 2: Ejecución del comando de instalación de Snort.

Durante la instalación, aparece una ventana para configurar el HOME_NET con el valor 10.0.2.0/24 identificado anteriormente.

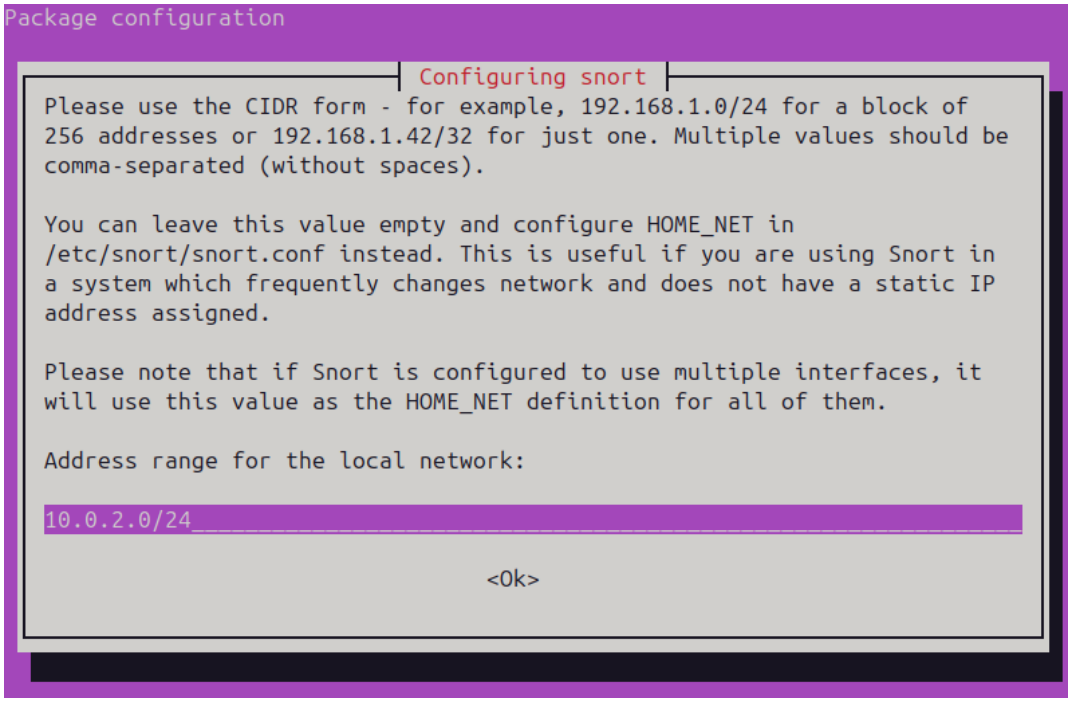

Figura 3: Configuración inicial de Snort; establecimiento del direccionamiento IP de la red local.

Para asegurar que Snort pueda capturar todo el tráfico de la red, se activa el modo promiscuo en la interfaz principal.

```
vboxuser@GiselaAlascio:~$ sudo ip link set enp0s3 promisc on
```

Figura 4: Configuración de la interfaz principal en modo promiscuo mediante el comando `ip link set`.

Este comando configura la interfaz `enp0s3` para que escuche todo el tráfico del segmento de red, no solo el destinado a la IP 10.0.2.15.

1.2. Configuración de la ruta de reglas.

Una vez instalado Snort, el siguiente paso es configurar el fichero principal (`/etc/snort/snort.conf`) para definir qué red proteger y qué reglas cargar.

```
vboxuser@GiselaAlascio:~$ sudo nano /etc/snort/snort.conf
```

Figura 5: Acceso al archivo de configuración de Snort mediante el editor de texto `nano`.

Tras ejecutar el comando `sudo nano /etc/snort/snort.conf` se abre el editor de texto `nano` cargando el fichero de configuración de Snort.

I. Configuración de variables de red:

- `ipvar HOME_NET`: se modifica para que coincida con la red de la máquina virtual.
- `ipvar EXTERNAL_NET`: se configura como `!HOME_NET` para indicar que "externo" es cualquier red que no sea la local.
- `ipvar DNS_SERVERS`: por defecto, se encontraba establecida como `$HOME_NET`, lo que significa que cualquier IP dentro de la red local como un servidor DNS legítimo.

```
 GNU nano 7.2                  /etc/snort/snort.conf
#
#ipvar HOME_NET any
ipvar HOME_NET 10.0.2.0/24

# Set up the external network addresses. Leave as "any" in most situations
#ipvar EXTERNAL_NET any
ipvar EXTERNAL_NET !$HOME_NET
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET
```

Figura 6: Configuración de variables de red.

II. Configuración de rutas de reglas:

Para aislar las alertas y probar únicamente nuestras 5 reglas personalizadas, se modifica la sección de inclusión de reglas (sección 7 del fichero `snort.conf`).

Entonces, se comentan, añadiendo `#`, todas las líneas `include \$RULE_PATH/...` (figura 7) que cargan las reglas por defecto de Snort. Así se elimina el "ruido" y se evitan falsos positivos de reglas que no se están probando. Además, se debe comprobar que la línea `include \$RULE_PATH/local.rules` esté activa, sin comentar, ya que es la directiva que le dice a Snort que cargue el fichero donde se van a definir las reglas personalizadas.

```
 GNU nano 7.2                    /etc/snort/snort.conf *
##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
##################################################

# Note to Debian users: The rules preinstalled in the system
# can be *very* out of date. For more information please read
# the /usr/share/doc/snort-rules-default/README.Debian file

#
# If you install the official VRT Sourcefire rules please review this
# configuration file and re-enable (remove the comment in the first line) those
# rules files that are available in your system (in the /etc/snort/rules
# directory)

# site specific rules
include $RULE_PATH/local.rules
```

```
 GNU nano 7.2                    /etc/snort/snort.conf *
# The include files commented below have been disabled
# because they are not available in the stock Debian
# rules. If you install the Sourcefire VRT please make
# sure you re-enable them again:

#include $RULE_PATH/app-detect.rules
#include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/backdoor.rules
#include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/blacklist.rules
#include $RULE_PATH/botnet-cnc.rules
#include $RULE_PATH/browser-chrome.rules
#include $RULE_PATH/browser-firefox.rules
#include $RULE_PATH/browser-ie.rules
#include $RULE_PATH/browser-other.rules
#include $RULE_PATH/browser-plugins.rules
#include $RULE_PATH/browser-webkit.rules
#include $RULE_PATH/chat.rules
#include $RULE_PATH/content-replace.rules
#include $RULE_PATH/ddos.rules
```

```
 GNU nano 7.2                    /etc/snort/snort.conf *
#include $RULE_PATH/dos.rules
#include $RULE_PATH/experimental.rules
#include $RULE_PATH/exploit-kit.rules
#include $RULE_PATH/exploit.rules
#include $RULE_PATH/file-executable.rules
#include $RULE_PATH/file-flash.rules
#include $RULE_PATH/file-identify.rules
#include $RULE_PATH/file-image.rules
#include $RULE_PATH/file-multimedia.rules
#include $RULE_PATH/file-office.rules
#include $RULE_PATH/file-other.rules
#include $RULE_PATH/file-pdf.rules
#include $RULE_PATH/finger.rules
#include $RULE_PATH/ftp.rules
# ICMP standard information queries will trigger these rules, they are very
# chatty, only enable if you need them
#include $RULE_PATH/icmp-info.rules
#include $RULE_PATH/icmp.rules
#include $RULE_PATH/imap.rules
#include $RULE_PATH/indicator-compromise.rules
```

```
 GNU nano 7.2                    /etc/snort/snort.conf *
#include $RULE_PATH/malware-cnc.rules
#include $RULE_PATH/malware-other.rules
#include $RULE_PATH/malware-tools.rules
#include $RULE_PATH/misc.rules
#include $RULE_PATH/multimedia.rules
#include $RULE_PATH/mysql.rules
#include $RULE_PATH/netbios.rules
#include $RULE_PATH/nntp.rules
#include $RULE_PATH/oracle.rules
#include $RULE_PATH/os-linux.rules
#include $RULE_PATH/os-other.rules
#include $RULE_PATH/os-solaris.rules
#include $RULE_PATH/os-windows.rules
#include $RULE_PATH/other-ids.rules
#include $RULE_PATH/p2p.rules
#include $RULE_PATH/phishing-spam.rules
#include $RULE_PATH/policy-multimedia.rules
#include $RULE_PATH/policy-other.rules
#include $RULE_PATH/policy.rules
#include $RULE_PATH/policy-social.rules
```

```
 GNU nano 7.2                    /etc/snort/snort.conf *
#include $RULE_PATH/policy-spam.rules
#include $RULE_PATH/pop2.rules
#include $RULE_PATH/pop3.rules
#include $RULE_PATH/protocol-finger.rules
#include $RULE_PATH/protocol-ftp.rules
#include $RULE_PATH/protocol-icmp.rules
#include $RULE_PATH/protocol-imap.rules
#include $RULE_PATH/protocol-pop.rules
#include $RULE_PATH/protocol-services.rules
#include $RULE_PATH/protocol-voip.rules
#include $RULE_PATH/pua-adware.rules
#include $RULE_PATH/pua-other.rules
#include $RULE_PATH/pua-p2p.rules
#include $RULE_PATH/pua-toolbars.rules
#include $RULE_PATH/rpc.rules
#include $RULE_PATH/rservices.rules
#include $RULE_PATH/scada.rules
#include $RULE_PATH/scan.rules
#include $RULE_PATH/server-apache.rules
#include $RULE_PATH/server-iis.rules
```

```
 GNU nano 7.2                    /etc/snort/snort.conf *
#include $RULE_PATH/server-mail.rules
#include $RULE_PATH/server-mssql.rules
#include $RULE_PATH/server-mysql.rules
#include $RULE_PATH/server-oracle.rules
#include $RULE_PATH/server-other.rules
#include $RULE_PATH/server-webapp.rules
# Note: These rules are disable by default as they are
# too coarse grained. Enabling them causes a large
# performance impact
#include $RULE_PATH/shellcode.rules
#include $RULE_PATH/smtp.rules
#include $RULE_PATH/snmp.rules
#include $RULE_PATH/specific-threats.rules
#include $RULE_PATH/spyware-put.rules
#include $RULE_PATH/sql.rules
#include $RULE_PATH/telnet.rules
#include $RULE_PATH/tftp.rules
#include $RULE_PATH/virus.rules
#include $RULE_PATH/voip.rules
#include $RULE_PATH/web-activex.rules
```

```
 GNU nano 7.2                    /etc/snort/snort.conf *
#include $RULE_PATH/web-attacks.rules
#include $RULE_PATH/web-cgi.rules
#include $RULE_PATH/web-client.rules
#include $RULE_PATH/web-coldfusion.rules
#include $RULE_PATH/web-frontpage.rules
#include $RULE_PATH/web-iis.rules
#include $RULE_PATH/web-misc.rules
#include $RULE_PATH/web-php.rules
#include $RULE_PATH/x11.rules
#include $RULE_PATH/community-sql-injection.rules
#include $RULE_PATH/community-web-client.rules
#include $RULE_PATH/community-web-dos.rules
#include $RULE_PATH/community-web-iis.rules
#include $RULE_PATH/community-web-misc.rules
#include $RULE_PATH/community-web-php.rules
#include $RULE_PATH/community-sql-injection.rules
#include $RULE_PATH/community-web-client.rules
#include $RULE_PATH/community-web-dos.rules
#include $RULE_PATH/community-web-iis.rules
#include $RULE_PATH/community-web-misc.rules
```

Figura 7: Modificación de las directivas `include` en `snort.conf` para deshabilitar las reglas predeterminadas y habilitar exclusivamente el fichero de reglas personalizadas (`local.rules`).

III. Configuración de salida:

Se comprueba que la línea `output unified2: filename snort.log, limit 128` esté activa, descomentada, que le dice a Snort que guarde las alertas en `unified2`, un formato binario muy rápido, en vez de usar un archivo de texto normal.

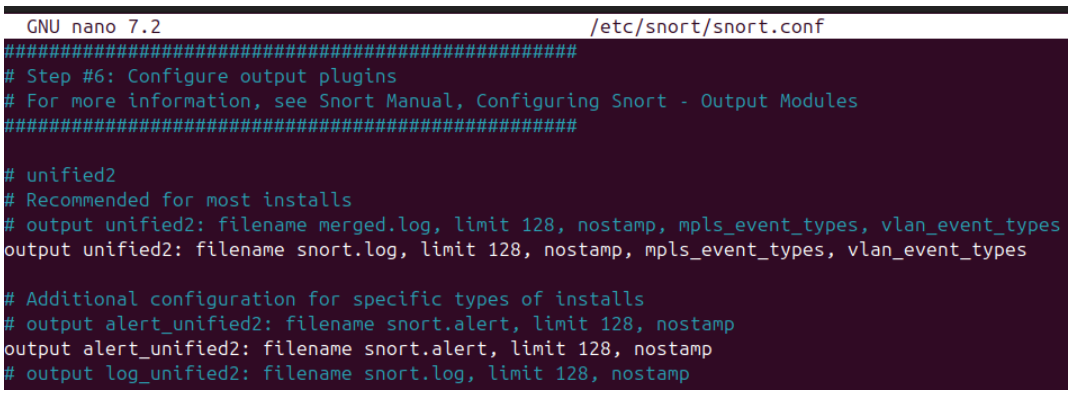

Figura 8: Configuración de salida de logs.

1.3. Definición de reglas.

Tras configurar `snort.conf` para que cargue exclusivamente el fichero `local.rules`, se procedió a definir las 5 reglas personalizadas requeridas por la práctica.

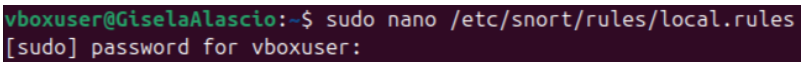

Figura 9: Apertura del archivo de reglas personalizadas de Snort.

El comando `sudo nano /etc/snort/rules/local.rules` abre el editor de texto `nano` y permite escribir las reglas directamente en el fichero:

- **Regla 1. Detección de *flags* TCP (escaneo nmap tipo Xmas).**

La regla alerta sobre un intento de escaneo de puertos de tipo "Xmas" (`-sX`) proveniente de la red externa y dirigido a nuestra `HOME_NET`.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Detección de escaneo TCP tipo XMAS dirigido a la red interna"; flags:FPU; sid:1000001; rev:1;)
```

- `alert tcp` le dice a Snort que solo inspeccione paquetes del protocolo TCP.
- `$EXTERNAL_NET any -> $HOME_NET any` define la dirección, busca tráfico que venga de fuera de la red local (`$EXTERNAL_NET`) a dentro de la red local (`$HOME_NET`).
- `flags:FPU;` le pide a Snort que mire los *flags* o indicadores en la cabecera del paquete TCP. FPU significa que la regla solo se disparará si los *flags* FIN (*Finish*), PSH (*Push*) y URG (*Urgent*) están todos activados al mismo tiempo.

Un paquete TCP legítimo que inicia una conexión no usa esta combinación de flags. El sondeo Xmas (`-sX`) de Nmap es una técnica de escaneo que fija los flags FIN, PSH y URG.

Se usa para identificar puertos ("abierto|filtrado") sin una conexión completa, basándose en si el puerto devuelve un RST (cerrado) o no (abierto). [1]

Para generar la evidencia requerida en el siguiente apartado, fue necesario simular el ataque utilizando la máquina virtual con la misma definición de red interna (`$HOME_NET`).

Snort interpreta que el tráfico de origen (`$HOME_NET`) no cumple con el requisito de la regla original, definida para tráfico procedente de la red externa (`$EXTERNAL_NET`). Al intentar escanear la propia IP desde la VM, el tráfico se clasifica como originado en `$HOME_NET`, lo que impedía la activación de la regla.

Por lo tanto, se modificó la variable de origen de la regla, pasando del valor restrictivo `$EXTERNAL_NET` al valor universal `any`.

> alert tcp any any -> $HOME_NET any (msg:"Detección de escaneo TCP tipo XMAS dirigido a la red interna"; flags:FPU; sid:1000001; rev:2;)

- **Regla 2. Detección de contenido 'porn' en URL (patrón).**

La regla detecta tráfico TCP que contenga la cadena "porn", sin importar si está en mayúsculas o minúsculas.

> alert tcp any any -> any any (msg:"Detección de solicitud HTTP que contiene la cadena 'porn' en la URL"; content:"porn"; nocase; sid:1000002; rev:1;)

- `alert tcp any any -> any any`: la regla se aplica a todo el tráfico TCP, sin importar su origen o destino.
- `content:"porn";` le dice a Snort que lea los datos reales del paquete y busque la secuencia de caracteres "porn".
- `nocase;` hace que la búsqueda de `content` sea insensible a mayúsculas y minúsculas, por lo que detectará "porn", "Porn", "PORN", etc.

Esta regla demuestra cómo un IDS puede ir más allá de los puertos y protocolos para inspeccionar el contenido real del tráfico.

- **Regla 3. Detección de *ICMP Echo Request*.**

La regla genera una alerta cada vez que se detecta un *ping* (*ICMP Echo Request*) saliendo desde la red interna.

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Echo Request saliente detectado desde la red interna"; itype:8; sid:1000003; rev:1;)
```

- `alert icmp` le dice a Snort que solo inspeccione paquetes del protocolo ICMP.
- `$HOME_NET any -> $EXTERNAL_NET any` define la dirección, alertando sobre los pings que salen de la red local.
- `itype:8;` le dice a Snort que busque solo un tipo específico de paquete ICMP. El `itype:8` es el código para un "Echo Request", es decir, la solicitud de *ping*.

Monitorizar *pings* salientes puede ser útil, ya que, a veces, son usados por *malware* para comprobar la conectividad a Internet o como parte de un escaneo de reconocimiento.

- **Regla 4. Petición DNS a servidor no autorizado.**

La regla alerta si un cliente de la `HOME_NET` intenta realizar una consulta DNS (protocolo UDP, puerto 53) a un servidor que no está en nuestra lista de confianza (`!$DNS_SERVERS`)

```
alert udp $HOME_NET any -> !$DNS_SERVERS 53 (msg:"Consulta DNS detectada, dirigida a un servidor no autorizado"; sid:1000004; rev:1;)
```

- `alert udp`: se enfoca en el protocolo UDP, ya que la mayoría de las consultas DNS estándar lo usan.
- `$HOME_NET any -> !$DNS_SERVERS 53` define la dirección y el puerto, buscando tráfico que sale de la red local (`$HOME_NET`) y va al puerto de destino 53, que es el puerto estándar de DNS.
- `!$DNS_SERVERS` considera que solo las direcciones IP locales, definidas por `$HOME_NET`, son consideradas servidores DNS legítimos, por lo que el filtro detecta correctamente cualquier consulta dirigida a una dirección IP externa (tráfico saliente).

El *malware* a menudo usa servidores DNS externos, como, por ejemplo, 8.8.8.8 o servidores maliciosos, para saltarse los filtros de seguridad de la red. Entonces, esta regla detecta ese comportamiento extraño.

- **Regla 5. Detección de ataque DoS (*ping flood*).**

La regla detecta un posible ataque de inundación de pings (*ping flood*).

> alert icmp $HOME_NET any -> any 8 (msg:“Posible ping flood detectado”; detection_filter: track by_src, count 10, seconds 10; sid:1000005; rev:1;)

- `alert icmp $HOME_NET any -> any 8` es similar a la regla 3, detecta *pings* (`itype:8`).
- `detection_filter: ...` es una opción de regla de Snort que se usa para limitar la tasa de alertas (*rate limiting*).
- `track by_src` le dice a Snort que “vigile” y cuente los paquetes basándose en su IP de origen (`by_src`).
- `count 10, seconds 10` es el umbral; solo genera una alerta después de ver 10 *pings* (`count 10`) de la misma IP de origen en un periodo de 10 segundos (`seconds 10`).

Esta regla es un gran ejemplo de un ajuste ingenioso. A diferencia de la regla 3, que es “ruidosa” y salta con cada *ping*, esta usa `detection_filter`, lo cual le permite ignorar el tráfico normal de diagnóstico y solo disparar la alarma cuando detecta un ataque real, un *ping flood* o DoS. De esta manera, se reducen mucho los falsos positivos y el ruido en los *logs*.

```
  GNU nano 7.2                          /etc/snort/rules/local.rules *
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

# Regla 1: Deteccion de flags TCP (escaneo nmap tipo Xmas).
alert tcp any any -> $HOME_NET any (msg:"Deteccion de escaneo TCP tipo Xmas dirigido a la red interna"; flags:FPU; \
 sid:1000001; rev:2;)

# Regla 2: Deteccion de contenido 'porn' en URL (Patrón)
alert tcp any any -> any any (msg:"Deteccion de solicitud HTTP que contiene la cadena 'porn' en la URL"; \
 content:"porn"; nocase; sid:1000002; rev:1;)

# Regla 3: Deteccion de ICMP Echo Request.
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Echo Request saliente detectado desde la red interna"; \
 itype:8; sid:1000003; rev:1;)

# Regla 4: Peticion DNS a servidor no autorizado.
alert udp $HOME_NET any -> !$DNS_SERVERS 53 (msg:"Consulta DNS detectada, dirigida a un servidor no autorizado"; \
 sid:1000004; rev:1;)

# Regla 5: Deteccion de ataque DoS (ping flood)
alert icmp $HOME_NET any -> any 8 (msg:"Posible ping flood detectado"; detection_filter: track by_src, count 10, \
 seconds 10; sid:1000005; rev:1;)
```

Figura 10: Contenido del fichero `local.rules` mostrando las cinco reglas Snort definidas.

1.4. Generación tráfico de prueba y comprobación de *logs*.

Para comprobar que las 5 reglas personalizadas funcionan correctamente, se procede a generar tráfico de prueba específico para cada una.

1.4.1. Metodología de las pruebas.

Se utiliza un entorno de dos terminales simultáneas para poder observar la causa (ataque) y el efecto (alerta) en tiempo real:

- **Terminal 1 - Vigilante:** se utiliza para el monitoreo y la detección, ejecutando Snort en modo consola. Para ello, se apunta al fichero de configuración, `snort.conf`, y se utiliza el parámetro `-A console`, que imprime las alertas directamente en la terminal, sirviendo como nuestro "*log*" en tiempo real, en lugar de escribir a `fast.log`.

```
sudo snort -A console -c /etc/snort/snort.conf -i <interfaz>
```

La interfaz de red seleccionada para Snort se alternó dependiendo del tipo de tráfico de prueba.

- **Regla 1 (Detección de escaneo Xmas):** para esta regla específica, Snort se inicia sobre la interfaz `lo`, ya que el tráfico de prueba, al ser generado y dirigido dentro del propio *host* local, no transita por la interfaz de red principal (`enp0s3`). Por lo tanto, la interfaz `lo` es la única que puede capturar y monitorear ese tráfico "intracomputadora" y permitir comprobar correctamente la firma de escaneo.

```
vboxuser@GiselaAlascio:~$ sudo snort -A console -c /etc/sno
rt/snort.conf -i lo
[sudo] password for vboxuser:
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901
 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 484
8 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 80
14 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8
300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371
 34443:34444 41080 50002 55555 ]
```

Figura 11: Comando de inicio de Snort en modo consola, especificando la interfaz `lo`.

- **Reglas 2, 3, 4 y 5:** para el resto de reglas que monitorean tráfico de red entrante y saliente, Snort se inicia y se mantiene escuchando en la interfaz de red principal, (`enp0s3`), que gestiona la comunicacion con la terminal 2 y la red externa simulada.

```
vboxuser@GiselaAlascio:~$ sudo snort -A console -c /etc/sno
rt/snort.conf -i enp0s3
[sudo] password for vboxuser:
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901
 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 484
8 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 80
14 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8
300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371
 34443:34444 41080 50002 55555 ]
```

Figura 12: Comando de inicio de Snort en modo consola, especificando la interfaz `enp0s3`.

- **Terminal 2 - Atacante:** se utiliza exclusivamente para generar el tráfico de prueba controlado. En ella, se instalan y ejecutan las herramientas necesarias para simular los ataques y las peticiones, incluyendo `curl`, para simular peticiones HTTP, `nmap`, para lanzar escaneos de flags, y `hping3`, para simular *ping flood*.

  La ejecución de comandos en esta terminal sirve como la "causa" que desencadena la alerta inmediata en la terminal 1.

```
vboxuser@GiselaAlascio:~$ sudo apt install curl nmap hping3
[sudo] password for vboxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  ncat ndiff zenmap
The following NEW packages will be installed:
  curl hping3 nmap
0 upgraded, 3 newly installed, 0 to remove and 80 not upgraded.
Need to get 2,021 kB of archives.
After this operation, 4,939 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 curl amd64 8.5.0-2ubuntu10.6 [226 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 hping3 amd64 3.a2.ds2-10build2 [100.0 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/universe amd64 nmap amd64 7.94+git20230807.3be01efb1+dfsg-3build2 [1,694 kB
]
Fetched 2,021 kB in 1s (1,451 kB/s)
Selecting previously unselected package curl.
(Reading database ... 151699 files and director
ies currently installed.)
Preparing to unpack .../curl_8.5.0-2ubuntu10.6_
amd64.deb ...
Unpacking curl (8.5.0-2ubuntu10.6) ...
Selecting previously unselected package hping3.
Preparing to unpack .../hping3_3.a2.ds2-10build
2_amd64.deb ...
Unpacking hping3 (3.a2.ds2-10build2) ...
Selecting previously unselected package nmap.
Preparing to unpack .../nmap_7.94+git20230807.3
be01efb1+dfsg-3build2_amd64.deb ...
Unpacking nmap (7.94+git20230807.3be01efb1+dfsg
-3build2) ...
```

Figura 13: Instalación de utilidades de prueba (`curl`, `nmap`, `hping3`).

1.4.2. Pruebas y evidencias.

A continuación, se detalla la prueba realizada para cada regla:

- **Prueba de la regla 1.**

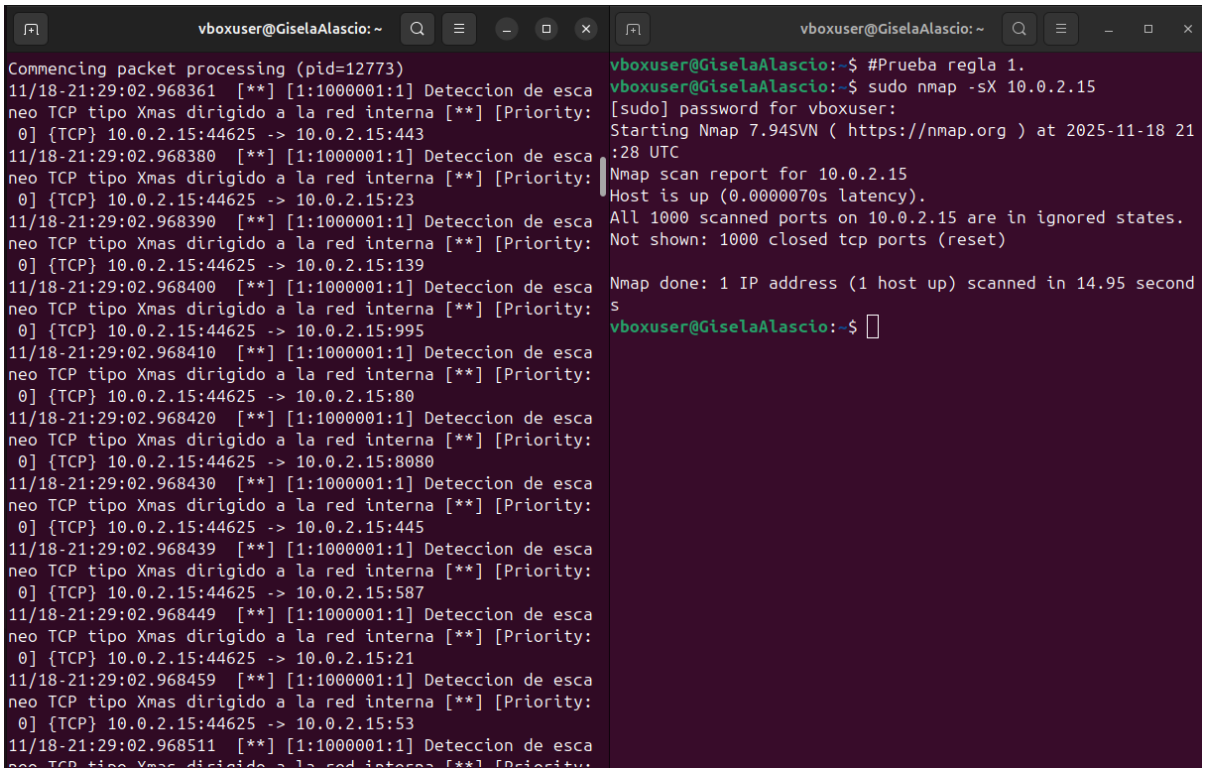

Figura 14: Ejecución del ataque (terminal 2 - derecha) y generación inmediata de la alerta por el IDS (terminal 1 - izquierda) de la regla 1.

Para simular el ataque, se ejecutó el comando `nmap` con el argumento `-sX` desde la terminal 2, dirigido hacia la IP de la propia máquina (10.0.2.15).

```
sudo nmap -sX 10.0.2.15
```

La figura 14 valida la correcta activación de la regla. La configuración de doble terminal demuestra la relación causa-efecto en tiempo real:

- **Terminal 2 (Causa)**: muestra la ejecución del comando `nmap -sX` y la finalización exitosa del escaneo de 1000 puertos.
- **Terminal 1 (Efecto)**: Snort registró múltiples alertas con el mensaje: "*Deteccion de escaneo TCP tipo Xmas dirigido a la red interna*". El registro de alerta (*log*) muestra el `sid:1000001` y el *timestamp* correspondiente a la detección.

Se observa que las alertas muestran la dirección IP de origen (10.0.2.15) y la de destino (10.0.2.15) como la misma, lo cual valida la modificación de la regla a `any` y el uso de la interfaz `lo` para tráfico interno.

La generación de múltiples alertas se debe a que `nmap` lanza paquetes Xmas a diversos puertos de destino y la regla se dispara individualmente por cada paquete que cumple la regla. Por lo tanto, la prueba demuestra que la regla, definida por el `flags:FPU`, es funcional y capaz de detectar reglas de escaneo sigiloso.

- **Prueba de la regla 2.**

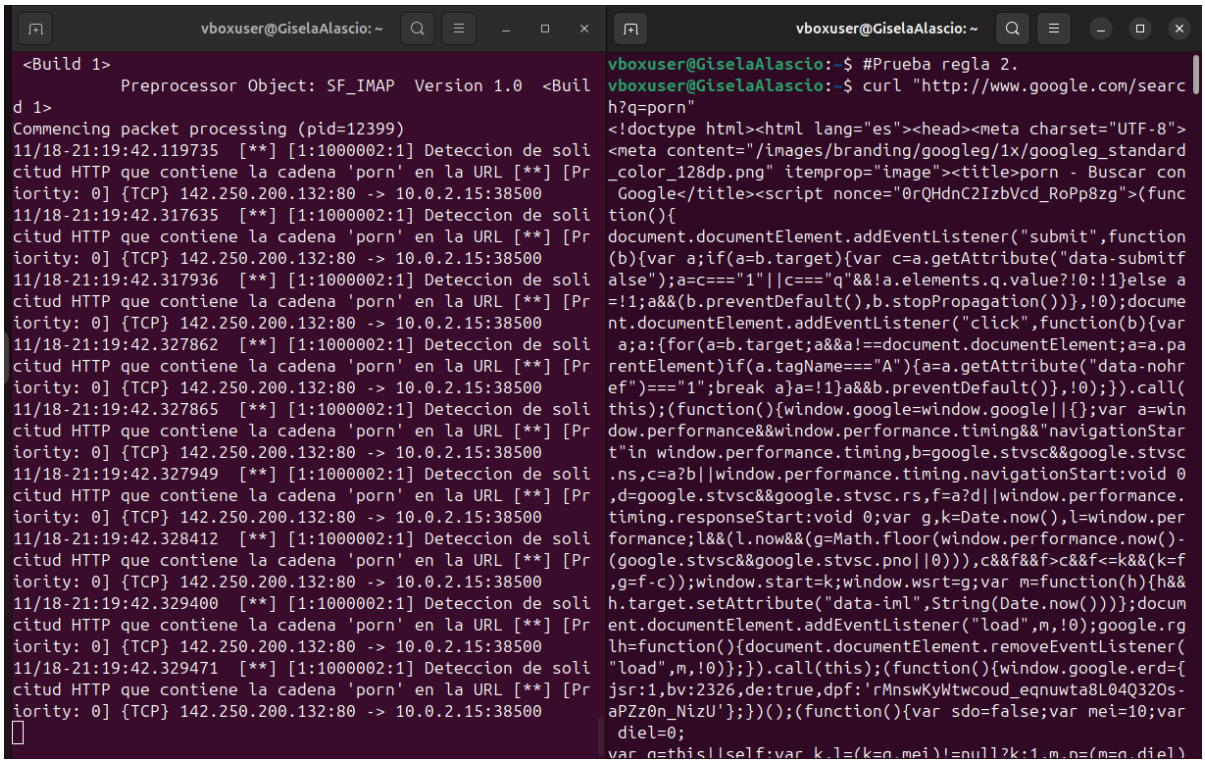

Figura 15: Ejecución del ataque (terminal 2 - derecha) y generación inmediata de la alerta por el IDS (terminal 1 - izquierda) de la regla 2.

Se simula una petición HTTP que incluye el patrón buscado en la URL. El comando se ejecuta desde la terminal 2, dirigida hacia un servidor web externo simulado:

```
curl "https://www.google.com/search?q=porn"
```

La figura 15 valida la detección de contenido. La configuración de doble terminal demuestra la relación causa-efecto en tiempo real:

- **Terminal 2 (Causa)**: muestra la ejecución de la petición web.
- **Terminal 1 (Efecto)**: Snort registró múltiples alertas con el mensaje: '*Detección de solicitud HTTP que contiene la cadena 'porn' en la URL*".

El registro de alerta (*log*) muestra el `sid:1000002` asociado a múltiples detecciones de tráfico TCP y la opción `content:"porn"` identificó satisfactoriamente la cadena de texto en la carga útil (*payload*) del paquete HTTP.

La prueba demuestra la funcionalidad de la detección basada en patrones de contenido utilizando la palabra clave `content`, que permite identificar solicitudes prohibidas o maliciosas sin necesidad de conocer los puertos o IP exactos.

- **Prueba de la regla 3.**

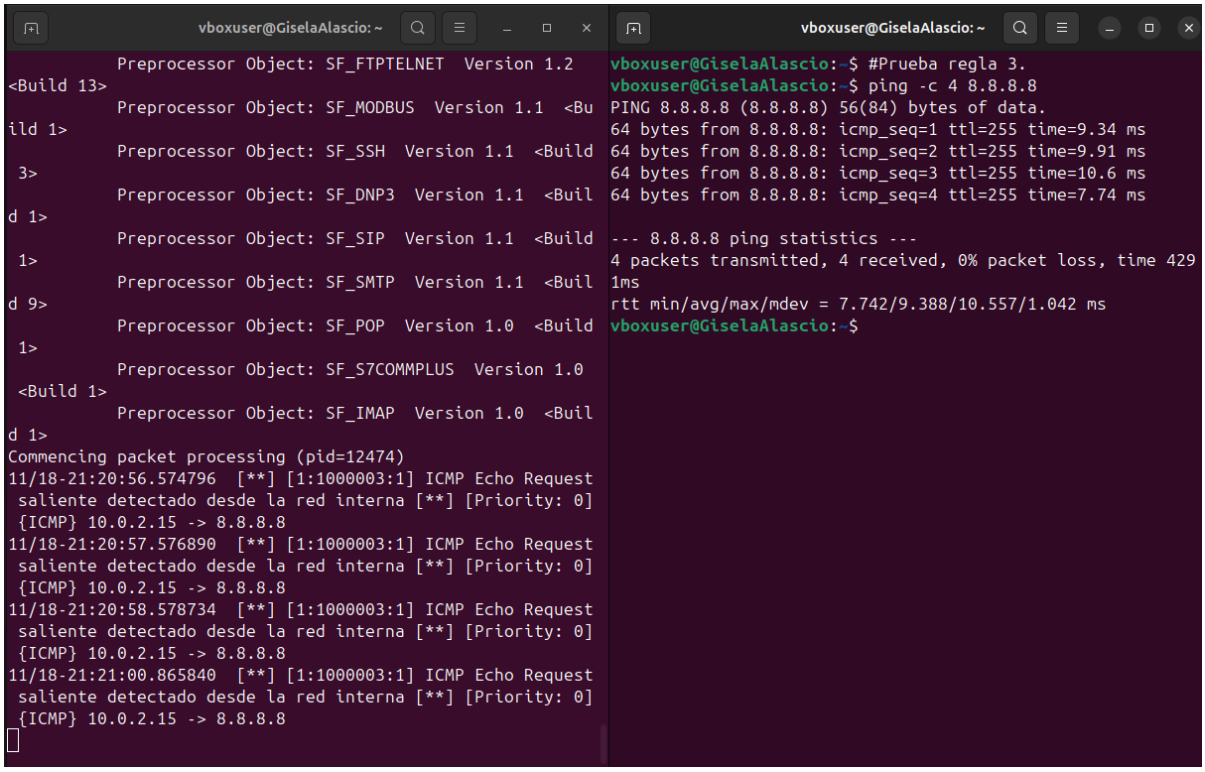

Figura 16: Ejecución del ataque (terminal 2 - derecha) y generación inmediata de la alerta por el IDS (terminal 1 - izquierda) de la regla 3.

La prueba se ejecuta desde la terminal 2 simulando la actividad normal de un usuario que verifica la conectividad de la red, enviando un número limitado de paquetes de prueba a un servidor externo conocido (8.8.8.8).

```
1      ping -c 4 8.8.8.8
```

El comando envia exactamente cuatro paquetes ICMP con el tipo *Echo Request*(`itype:8`) desde la máquina interna hacia el exterior.

La figura 16 valida la detección demostrando la relación causa-efecto en tiempo real:

- **Terminal 2 (Causa)**: muestra la ejecución del comando y el resultado de cuatro paquetes transmitidos y recibidos (0 % pérdida).
- **Terminal 1 (Efecto)**: Snort registró cuatro alertas consecutivas, una por cada paquete ICMP enviado, con el mensaje: "*ICMP Echo Request saliente detectado desde la red interna*". El registro de alerta (*log*) muestra el `sid:1000003` asociado al protocolo ICMP.

La coincidencia entre los paquetes enviados (4) y las alertas registradas (4) confirma que la regla detectó de manera precisa el tráfico de salida con el tipo `itype:8` definido. Por lo tanto, la prueba demuestra la capacidad de filtrar tráfico basado en el protocolo `icmp` y las opciones específicas de protocolo, `itype`, confirmando que la regla es funcional para el monitoreo de comunicaciones básicas de red.

- **Prueba de la regla 4.**

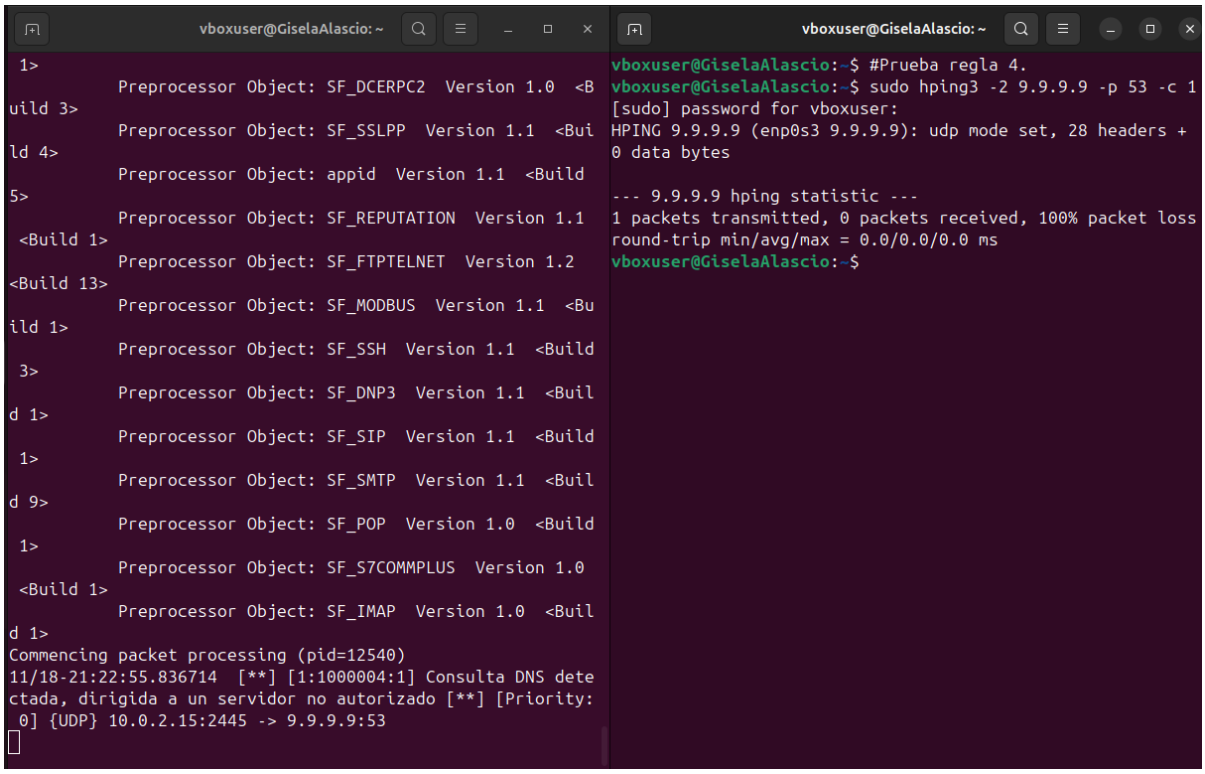

Figura 17: Ejecución del ataque (terminal 2 - derecha) y generación inmediata de la alerta por el IDS (terminal 1 - izquierda) de la regla 4.

Se utiliza el comando `hping3` para simular un cliente de la red interna intentando realizar una consulta DNS a un servidor público externo (9.9.9.9), el cual no está incluido en la lista de `$DNS_SERVERS`.

```
1      sudo hping3 -2 9.9.9.9 -p 53 -c 1
```

El comando obliga a Snort a procesar un paquete UDP (-2) dirigido al puerto 53 (`-p 53`). Snort valida que el destino del paquete no coincidía con la variable `$DNS_SERVERS` disparando la alerta.

La figura 17 valida la detección basada en la negación de variables de red:

- **Terminal 2 (Causa)**: muestra la ejecución del comando `hping3` enviando un único paquete de prueba (`-c 1`) hacia la IP no autorizada.
- **Terminal 1 (Efecto)**: Snort registró una única alerta con el mensaje: "*Consulta DNS detectada, dirigida a un servidor no autorizado*".

  El registro de alerta (*log*) muestra el `sid:1000004` asociado al protocolo UDP y la alerta indica la IP de origen interna (10.0.2.15) y la IP de destino no autorizada (9.9.9.9).

La prueba confirma la funcionalidad de la sintaxis `!$DNS_SERVERS`, demostrando la capacidad de Snort para controlar el flujo de tráfico mediante la negación de variables de red.

- **Prueba de la regla 5.**

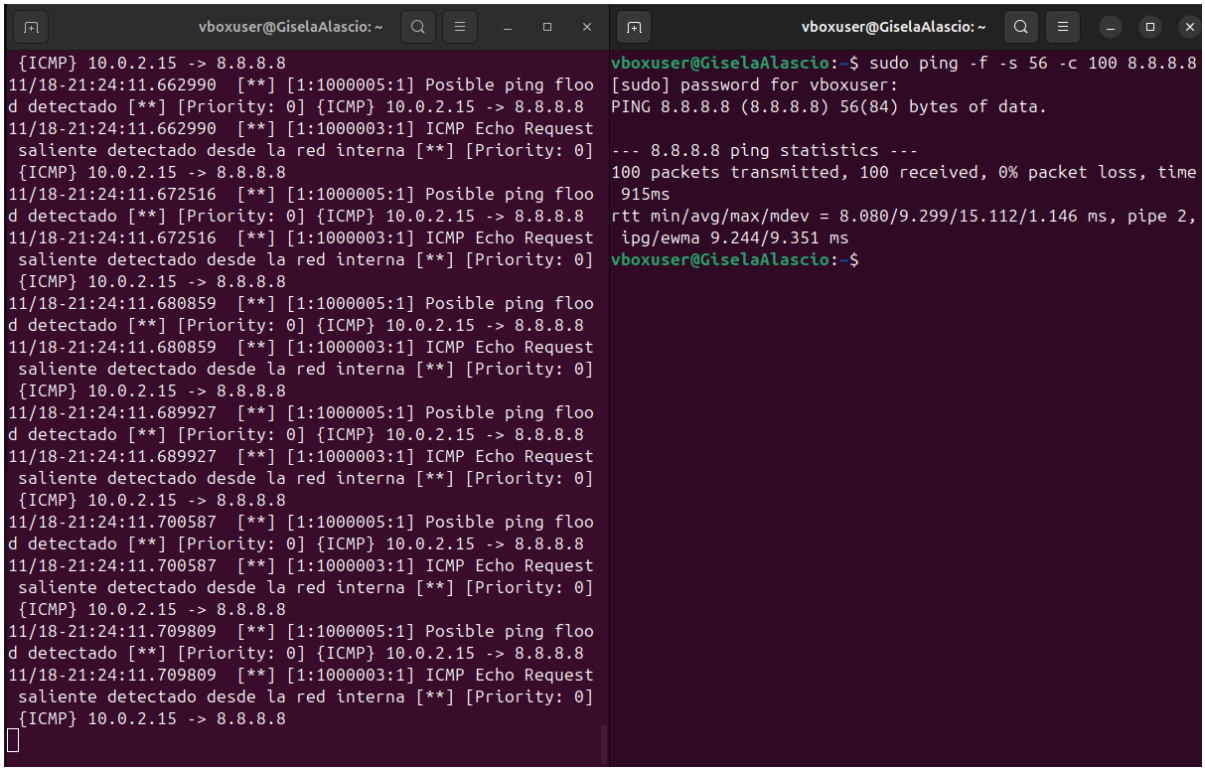

Figura 18: Ejecución del ataque (terminal 2 - derecha) y generación inmediata de la alerta por el IDS (terminal 1 - izquierda) de la regla 5.

Se utiliza la terminal 2 para simular una ráfaga intensa de tráfico ICMP, superando el umbral de detección por volumen.

```
1      sudo ping -f -s 56 -c 100 8.8.8.8
```

El comando envia 100 paquetes ICMP (`-c 100`) a máxima velocidad (`-f`) hacia el destino externo y Snort procesa cada paquete individualmente. El parámetro `-s 56` especifica que el tamaño de los datos en el paquete ICMP es de 56 bytes.

La figura 18 valida el funcionamiento del filtro de detección por tasa:

- **Terminal 2 (Causa)**: muestra la ejecución del ataque de ping en modo *flood* (`-f`). La ráfaga de 100 paquetes ICMP fue enviada a la máxima velocidad posible para superar el umbral establecido por el filtro de detección por tasa.
- **Terminal 1 (Efecto)**:
  - **Fase de conteo**: inicialmente (durante los 9 primeros paquetes) Snort no muestra la alerta `sid:1000005`, sino que cuenta el tráfico del origen (`track by_src`) y dispara únicamente la alerta más general (Regla 3: *ICMP Echo Request saliente...*), como se puede ver en la figura 19.
  - **Activación del filtro**: una vez de supera el umbral de `count 10` dentro del plazo de `seconds 10`, la alerta *Posible ping flood detectado* (`sid:1000005`) se dispara inmediatamente.

Entonces, la terminal 1 se inunda con la alerta `sid:1000005` por cada paquete siguiente al décimo, confirmando que el `detection_filter` pasó del modo de conteo al modo de alerta activa.

La prueba demuestra la funcionalidad y el valor de la palabra clave `detection_filter` para aplicar políticas de umbral, permitiendo mitigar el "ruido" y evitando así una alerta por cada *ping* normal y falsos positivos.

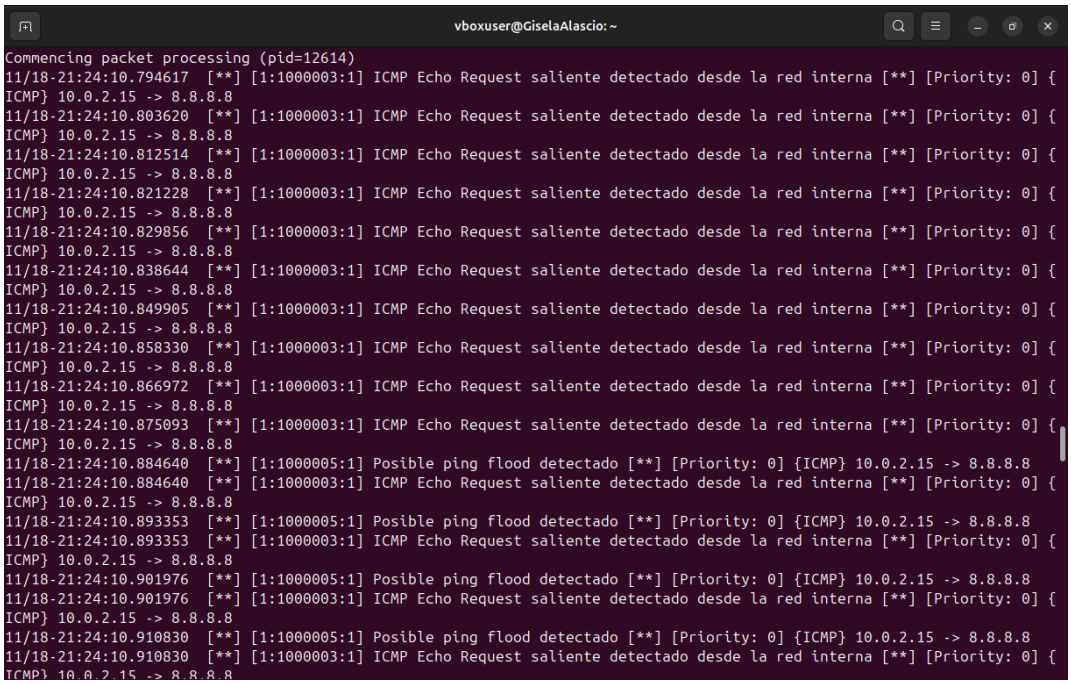

Figura 19: Activación de la alerta tras superar el umbral del filtro de detección por tasa.

2. Tramo B — Suricata

El objetivo de este tramo es implementar un *Sistema de Detección de Intrusione*s (IDS) alternativo a Snort, en este caso, Suricata, que permitirá configurar una regla de detección para realizar una comparativa técnica sobre sintaxis, configuración y formato de *logs* entre ambas herramientas.

2.1. Instalación y configuración básica.

El proceso se llevó a cabo en la misma máquina virtual de laboratorio empleada en el Tramo A.

Entonces, se utilizó el PPA (*Personal Package Archive*) oficial del equipo OISF (*Open Information Security Foundation*) para asegurar la instalación de la versión más reciente y estable de Suricata.

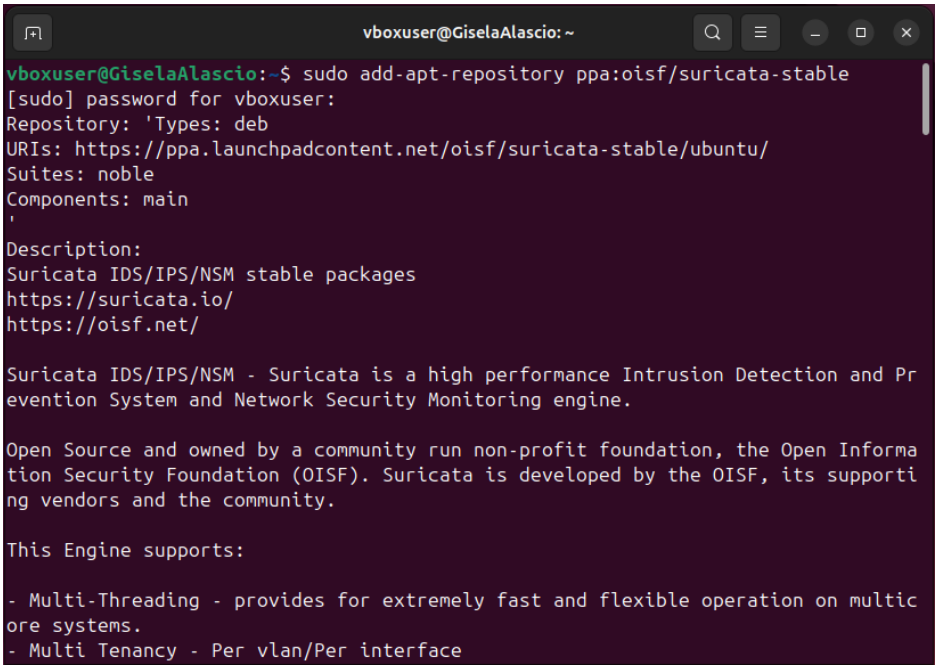

Figura 20: Adición del repositorio PPA oficial para obtener los paquetes de Suricata.

A continuación, se procedió a actualizar la lista de paquetes del sistema y se instaló el paquete `suricata-dbg`.

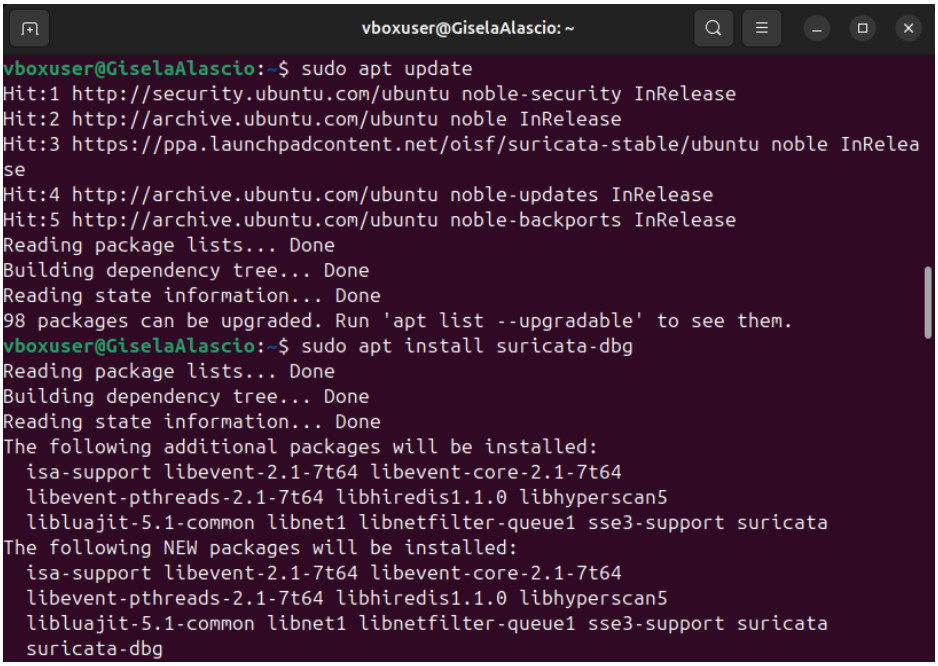

Figura 21: Actualización de la lista de paquetes e instalación del paquete `suricata-dbg`.

2.2. Configuración de la ruta de reglas.

A diferencia de Snort, cuya configuración está fragmentada en múltiples ficheros, Suricata utiliza un único fichero centralizado (`suricata.yaml`) para definir todas las variables de red, la interfaz de escucha y las rutas de reglas. Se realizaron las siguientes modificaciones:

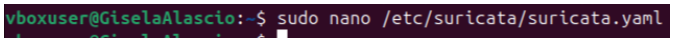

Figura 22: Inicio del archivo de configuración principal,`suricata.yaml`.

1. **Definición de red e interfaz**: se ajustan las variables de red para que coincidan con la configuración de Snort y se especifica la interfaz de escucha.
   - **Variables de red**: se configura la variable `HOME_NET` con la red interna de la máquina virtual (10.0.2.0/24) y se comprueba que `EXTERNAL_NET` esté definida como la negación de la red interna `!$HOME_NET`.

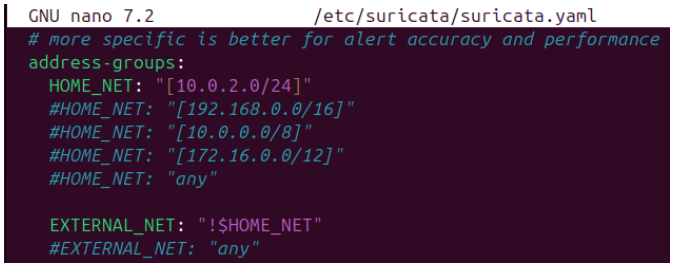

Figura 23: Configuración de las variables de red.

- **Modo de captura** (`af-packet`): se especifica la interfaz `enp0s3` como punto de escucha.

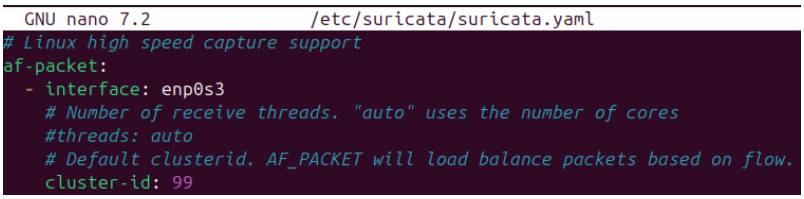

Figura 24: Configuración de la sección `af-packet`.

2. **Configuración de reglas personalizadas**: se establece la ruta base `default-rule-path` a `/etc/suricata/rules`, que asegura la correcta carga del archivo de reglas personalizadas, `practica.rules`, incluido en la sección `rule-files`.

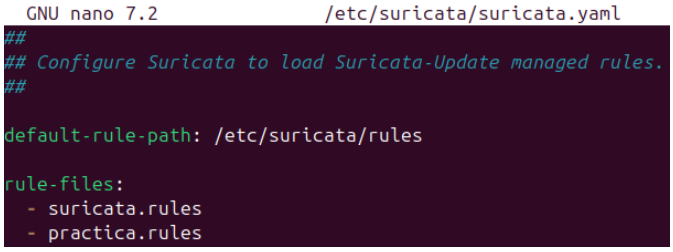

Figura 25: Configuración de las rutas de reglas.

2.3. Definición de la regla.

Para configurar las reglas personalizadas, se crea el directorio específico donde se alojará la regla de la práctica con el comando `sudo mkdir -p`, que garantiza la existencia del directorio `/etc/suricata/rules`, que es la ruta definida en el parámetro `default-rule-path` del archivo `suricata.yaml`.

Entonces, se abre el archivo `practica.rules` en el editor de texto `nano` y se implementa una regla de detección de tráfico ICMP, que es literalmente idéntica a la regla 3 definida previamente para Snort, demostrando la similitud sintáctica entre ambos motores de detección.

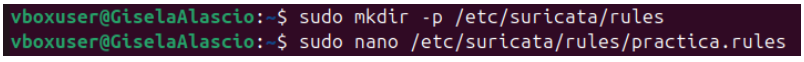

Figura 26: Creación del directorio de reglas y apertura del archivo `practica.rules`.

La regla definida para la prueba fue la siguiente:

> alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:“ICMP Echo Request saliente detectado desde la red interna”; itype:8; sid:2000001; rev:1;)

Esta regla detecta peticiones de *ping* que salen de la red protegida (`$HOME_NET`) hacia el exterior (`$EXTERNAL_NET`), replicando la función de detección ICMP de Snort. Ambos motores utilizan la misma estructura de *Action + Protocol + Header + Options*.

2.4. Generación tráfico de prueba y comprobación de *logs*.

2.4.1. Metodología de las pruebas.

Se utilizan simultáneamente tres terminales para simular un entorno de IDS en tiempo real:

- **Terminal 1 - Vigilante**: monitorea el archivo de alertas (`fast.log`).
- **Terminal 2 - Motor IDS**: inicia el motor Suricata en modo de escucha.
- **Terminal 3 - Atacante**: lanza el tráfico de prueba.

En primer lugar, en la terminal 1, se inicia la monitorización continua del archivo de logs utilizando el comando:

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

Luego, en la terminal 2, se arranca el motor Suricata en modo de escucha, indicando el archivo de configuración y la interfaz de red (`enp0s3`):

```
1     sudo suricata -c /etc/suricata/suricata.yaml -i enp0s3
```

Antes de generar tráfico de prueba, se espera la confirmación de que el motor ha iniciado correctamente ("*Engine started*").

2.4.2. Pruebas y evidencias.

Una vez el motor está activo, desde la terminal 3, se lanza un *ping* al servidor externo, solicitando el envío de cuatro paquetes ICMP:

```
1     ping -c 4 8.8.8.8
```

Al ejecutarse el *ping*, el sistema IDS registra la detección de la regla. Al enviar cuatro paquetes ICMP, el *log* en la terminal 1 registra exactamente cuatro alertas, una por cada paquete enviado.

Por lo tanto, la detección exitosa de cuatro alertas de "ping saliente detectado" valida la funcionalidad y la precisión de la regla.

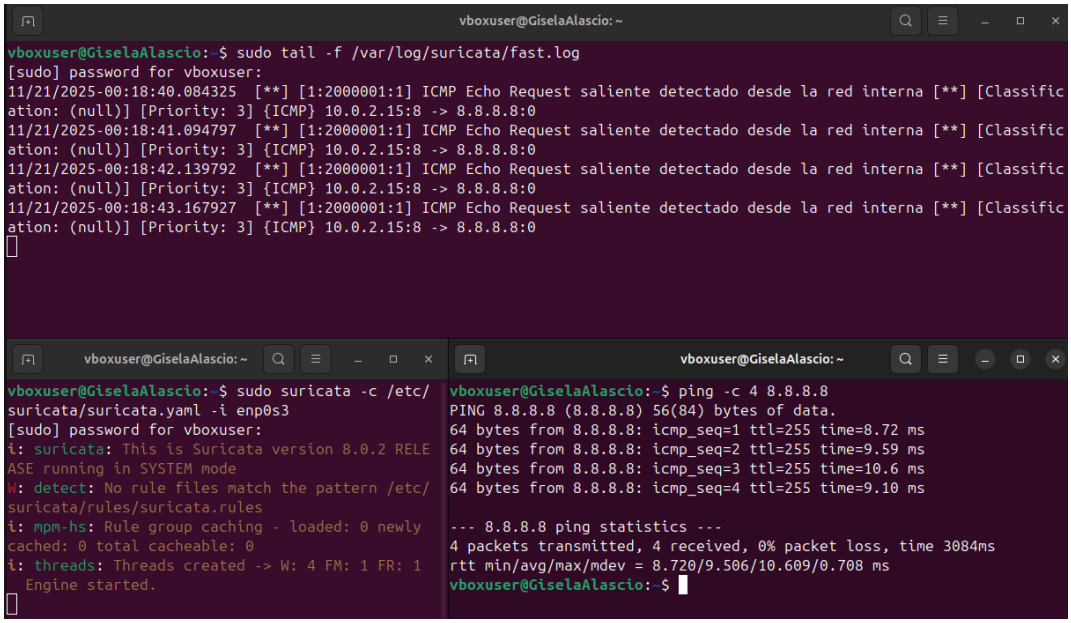

Figura 27: Demostración de la prueba de *ping* y *logs* en tres terminales.

3. Análisis comparativo — Snort vs. Suricata

A lo largo de la práctica, al crear reglas en ambos sistemas, descubrimos que Snort y Suricata son muy parecidos superficialmente, lo que facilita la portabilidad de reglas de uno a otro. Sin embargo, sus diferencias más grandes están en cómo están construidos por dentro, lo cual es clave para el rendimiento en redes grandes.

3.1. Similitud del lenguaje (sintaxis)

La sintaxis para escribir las reglas es prácticamente idéntica, ya que Snort fue el motor pionero en establecer el lenguaje de reglas y Suricata lo adoptó para ser compatible.

Por lo tanto, si se aprende a escribir una regla en Snort, como la de ICMP, funciona de manera análoga en Suricata, demostrando la alta portabilidad del set de reglas básico entre ambos sistemas.

3.2. Rendimiento y arquitectura

La diferencia fundamental reside en el diseño del motor, impactando directamente en la capacidad de manejar tráfico en entornos reales:

- **Snort**: tradicionalmente opera con un modelo monohilo (*single-thread*), por lo que, aunque la máquina virtual tenga muchos núcleos de CPU, Snort utiliza solo uno para procesar los paquetes, convirtiéndolo en un cuello de botella en redes con mucho tráfico y restringiendo su velocidad máxima. [2]

- **Suricata**: soporta un modelo multihilo nativo (*multi-threading*). El motor puede usar todos los núcleos del CPU de forma paralela, permitiendo procesar el tráfico a altas velocidades, dándole una ventaja superior en términos de rendimiento y estabilidad operativa. [2]

3.3. Formato de *logs*

La forma en que el IDS entrega la información es tan importante como la forma en que la detecta. Aquí, Suricata ofrece una ventaja significativa:

- **Snort**: utiliza formatos más tradicionales como ASCII (`fast.log`) y `Unified2` (binario) cuya integración con SIEM requiere un procesador intermedio (como `Barnyard2`). [3]
- **Suricata**: utiliza un formato JSON estructurado, que facilita enormemente la integración con herramientas SIEM. [3]

3.4. Facilidad de uso

Snort se enfoca más en la detección basada en puertos estándar y requiere una configuración más manual para la detección de protocolos en puertos no estándar. No obstante, Suricata posee una detección avanzada de aplicaciones, como la detección HTTP o SSH en puertos no estándar, que permite la extracción de archivos del tráfico de red para análisis forense. [4]

3.5. Conclusión final

Snort es la solución histórica, robusta y simple para entornos de red pequeños y controlados y se considera ideal para la educación por su simplicidad de uso inicial y su amplia documentación disponible. La práctica realizada confirma esto, ya que la implementación y verificación de las 5 reglas definidas resultó ser un proceso directo que permite aprender la lógica fundamental de un sistema de detección. Por lo tanto, el modelo de un solo hilo de Snort es eficiente para esta etapa de aprendizaje y su bajo consumo de recursos lo hace ideal para máquinas virtuales de laboratorio.

Sin embargo, Suricata es la herramienta más moderna, potente y escalable, diseñada para cubrir las necesidades de seguridad en entornos de producción de gran volumen. Mientras que Snort solo puede utilizar un procesador a la vez, Suricata puede aprovechar todos los núcleos de la máquina para procesar el tráfico en paralelo, lo cual le otorga una velocidad y rendimiento superiores y le permite manejar eficazmente grandes volúmenes de datos sin saturarse.

Es importante notar que, dado que en el entorno de la práctica solo se manejaron unos pocos paquetes y un tráfico muy reducido, la superioridad de velocidad de Suricata sobre Snort apenas fue perceptible. Sin embargo, esta capacidad es fundamental para el entorno profesional, donde el rendimiento multihilo es importante para el monitoreo de tráfico a gran escala.

4. Evidencia audiovisual.

Para plasmar el procedimiento explicado en esta práctica, se ha elaborado un vídeo que documenta visualmente los pasos esenciales del laboratorio. Este material audiovisual incluye la instalación de ambos sistemas IDS, la configuración clave de rutas y variables y las demostraciones en vivo de la activación de las reglas definidas en el Tramo A y Tramo B.

El vídeo puede ser visualizado en el siguiente enlace:

Ver video aquí

Referencias

[1] Nmap.org. *Técnicas de sondeo de puertos | Guía de referencia de Nmap (Página de manual)* [En línea]. Disponible en: https://nmap.org/man/es/man-port-scanning-techniques.html.

[2] Medium. *Suricata vs snort: guía detallada de los programas* [En línea]. Disponible en: https://medium.com/@redfanatic7/suricata-vs-snort-detailed-guide-to-the-programs-c331cff452a1.

[3] Suricata. *XFF and Syslog Output* [En línea]. Disponible en: https://forum.suricata.io/t/xff-and-syslog-output/1621.

[4] NextdoorSEC - Penetration Testing Worldwide. *Suricata vs. Snort: Choosing the Right IDS* [En línea]. Disponible en: https://nextdoorsec.com/suricata-vs-snort/.