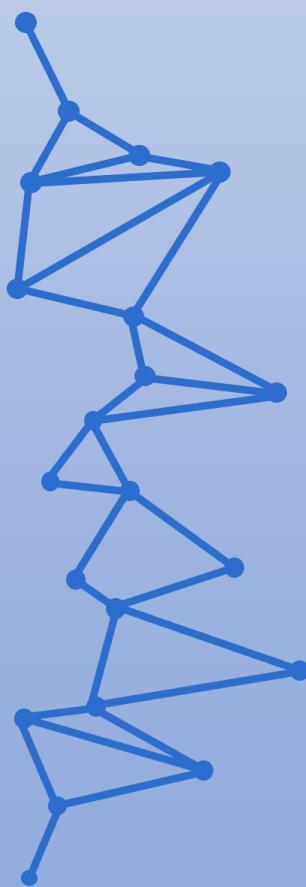




Curso de Especialización de Ciberseguridad en Entornos de Tecnología de la Información (CETI)



Incidentes de Ciberseguridad

UD06. Detección multipunto de
Incidentes.
Tarea Online.

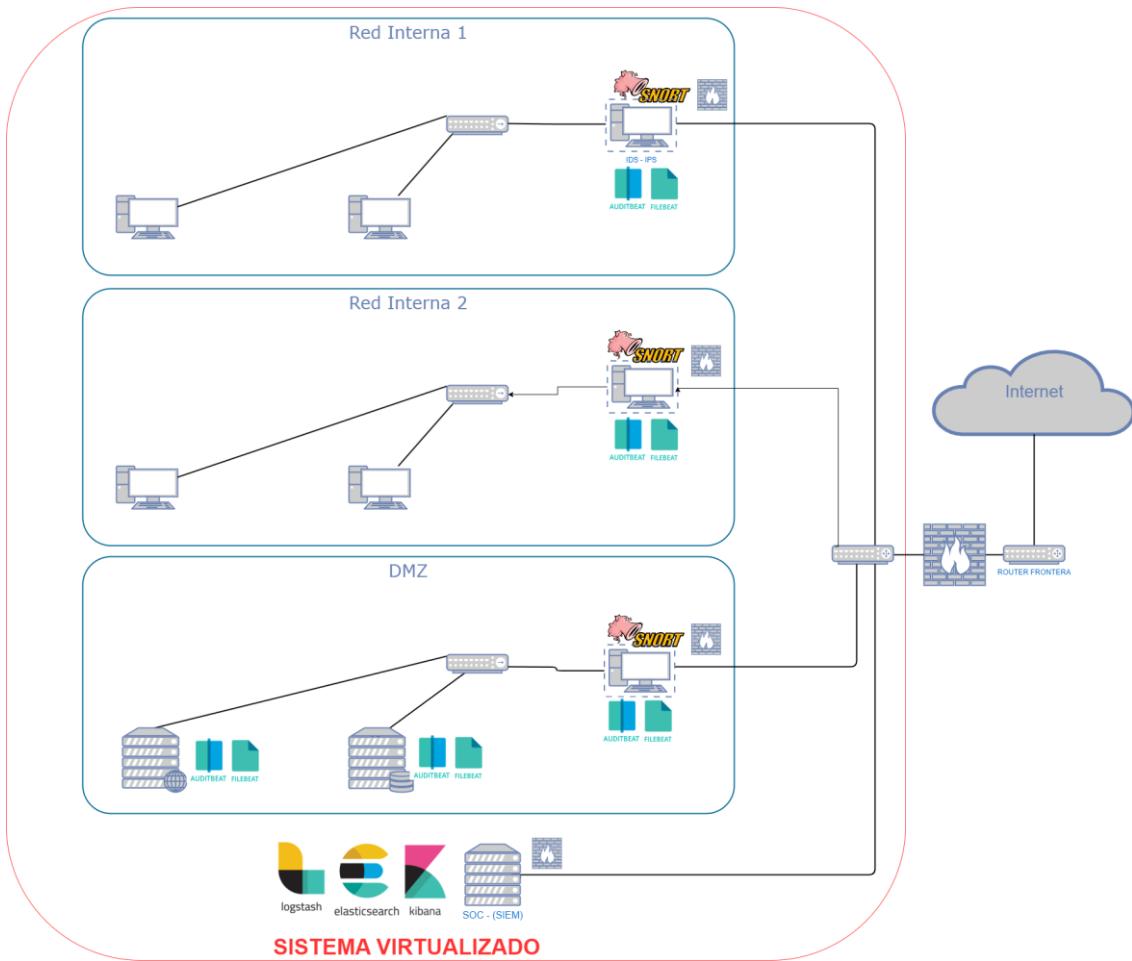
JUAN ANTONIO GARCIA MUELAS

INDICE

	Pag
1. Descripción de la tarea	2
2. Configurar las máquinas virtuales para que tengan comunicación completa	3
3. Configuración del IDS para registrar el tráfico de red. (SNORT)	8
4. Pruebas de las alertas generadas	16
5. Webgrafía	18

1.- Descripción de la tarea.

La Detección Multipunto de Incidentes



José Antonio Santos Gómez *Esquema Maqueta Tareas 6 y 7 (CCO)*

En la Unidad 6 hemos estudiado cómo instalar y configurar el IDS Snort, situándolo en la misma máquina en la que estará el SIEM que procesará su información una vez filtrada y almacenada.

Sin embargo, aunque esta configuración es habitual en los laboratorios, no es la corriente en las instalaciones reales. En cualquier entorno productivo suele haber una sonda Snort en cada una de las máquinas perimetrales, comprometidas, vulnerables, etc., cuya información de logging se ha de redirigir hacia una única máquina en la que estará instalado el SIEM (Unidad 7).

En esta tarea abordaremos el registro de logs en los diferentes agentes IDS en tiempo real utilizando la aplicación SNORT.

¿Qué te pedimos que hagas?

Para el desarrollo de la práctica nos centraremos en la red DMZ, en concreto sobre el SNORT situado en dicha zona y una de las máquinas, la cual, tiene instalado los servicios SSH, HTTP y MySQL. Esta última máquina se proporciona en esta tarea ([WebServer](#)).

✓ **Apartado 1: Configurar las máquinas virtuales para que tengan comunicación completa.**

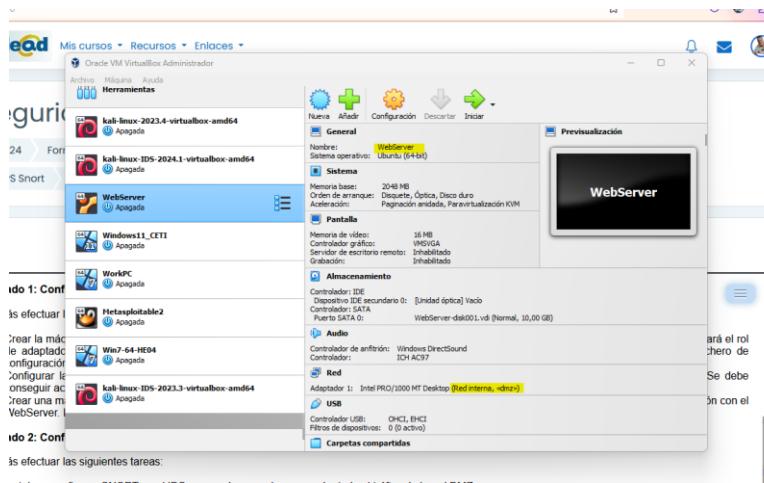
Deberás efectuar las siguientes tareas:

- Crear la máquina IDS (Snort) con dos interfaces de red y configurarla para que permita la comunicación completa entre ambas interfaces. Una tomará el rol de adaptador puente con la red externa y la otra interfaz sería la puerta de enlace predeterminada de la red DMZ. Se debe mostrar el fichero de configuración de las interfaces de red. *Se recomienda el uso de Ubuntu SERVER o DEBIAN.*
- Configurar la máquina IDS para que las máquinas de la red interna DMZ (WebServer) se puedan comunicar correctamente con el exterior. Se debe conseguir acceso a internet y a la red externa.
- Crear una máquina virtual que se denomine SOC, la cual esté conectada a la red externa (adaptador puente). Esta máquina debe tener comunicación con el WebServer. La máquina SOC debe tener interfaz gráfica, por lo que se recomienda la instalación de Ubuntu Desktop.

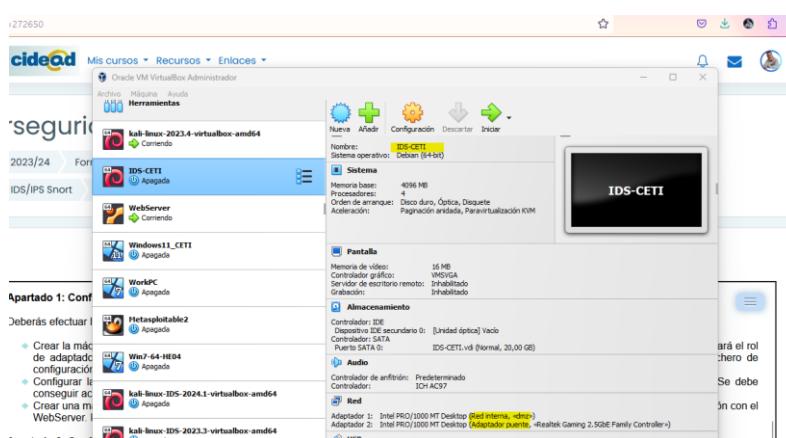
Para esta tarea vamos a descargar la Ubuntu Server facilitada para utilizarse como **webserver**, una máquina con Debian 12 para hacer de servidor **IDS** y una Kali como **SOC**, que configuraremos a continuación.

Primero revisamos las configuraciones de red de las máquinas.

La del **webserver** ya está configurada con el adaptador de Red Interna, DMZ.



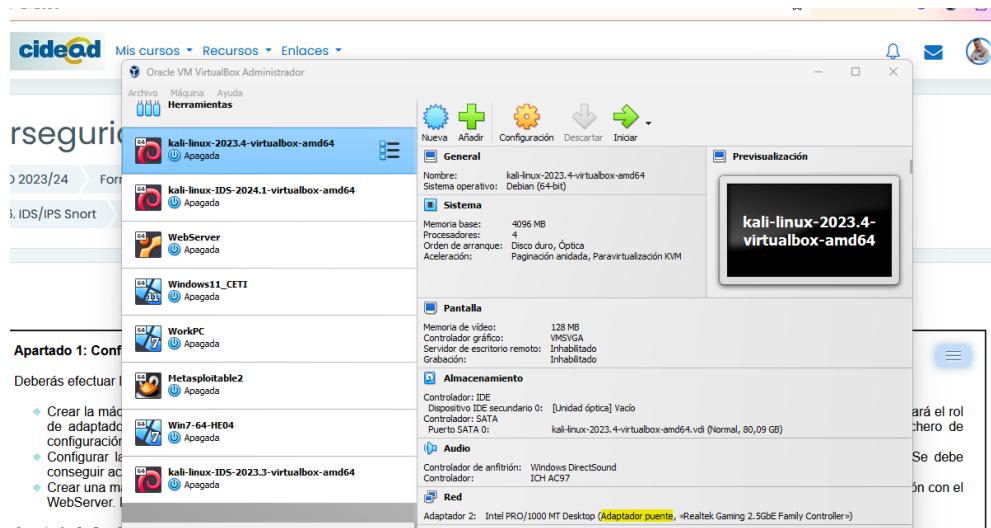
Para la máquina **IDS**, habilitamos los adaptadores Red Interna-DMZ y puente.



Incidentes de Ciberseguridad

Tarea Online UD06.

En la máquina que hace de **SOC**, se habilita el adaptador puente.



Arrancamos las tres máquinas y comprobamos la configuración del **webserver**.

/etc/netplan/00-installer-config.yaml

```
GNU nano 6.2. 00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  renderer: networkd
  ethernets:
    ens3:
      dhcp4: no
      addresses: [192.168.10.100/24]
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
      routes:
        - to: default
          via: 192.168.10.1
version: 2
```

Al finalizar el desarrollo de la práctica nos centremos en la configuración de la red. Se ha habilitado la red externa y se ha añadido los servicios SSH, HTTP y MySQL. Esta configuración es necesaria para poder acceder a la máquina desde el exterior.

Apartado 1: Configurar las máquinas virtuales

Deberás efectuar las siguientes tareas:

- ◆ Crear la máquina IDS (Snort) con dos interfaces de red. Una de adaptador puente con la red externa y la otra interfaz será la puerta de enlace predeterminada de la red DMZ. Se recomienda el uso de Ubuntu SERVER o DEBIAN.
- ◆ Configurar la máquina IDS para que las máquinas de la red interna DMZ (WebServer) se puedan comunicar correctamente con el exterior. Se debe conseguir acceso a internet y a la red externa.
- ◆ Crear una máquina virtual que se denominé SOC, la cual esté conectada a la red externa (adaptador puente). Esta máquina debe tener comunicación con el exterior.

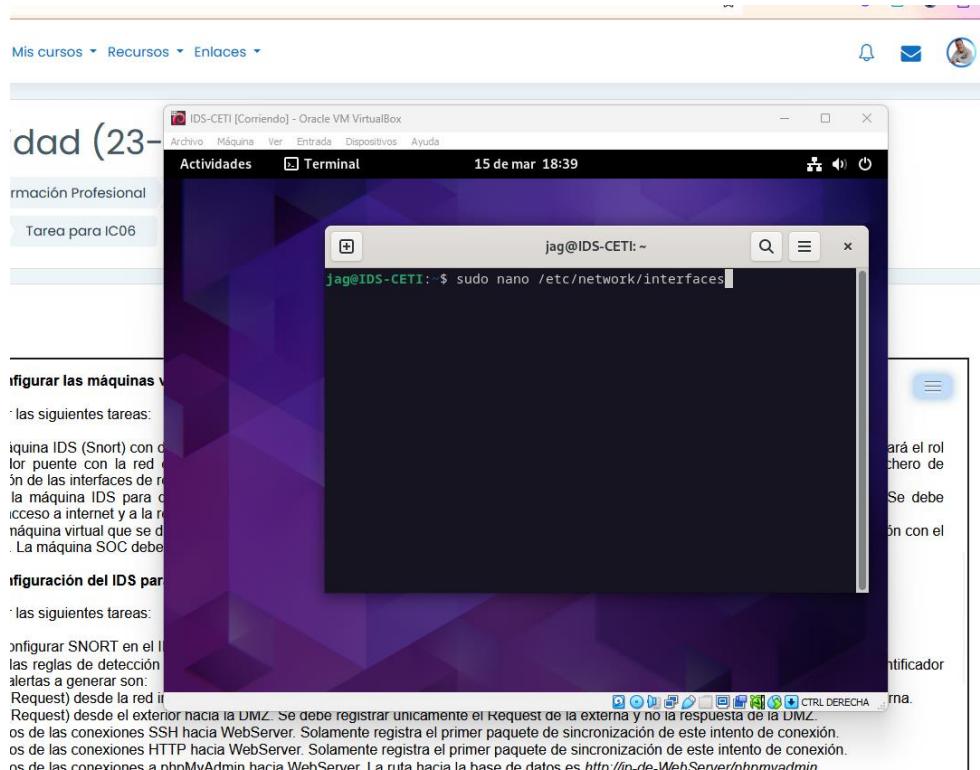
Teniendo en cuenta la dirección de esta máquina, vamos al **IDS** a editar el archivo de configuración.

cd /etc/network

sudo nano interfaces

O

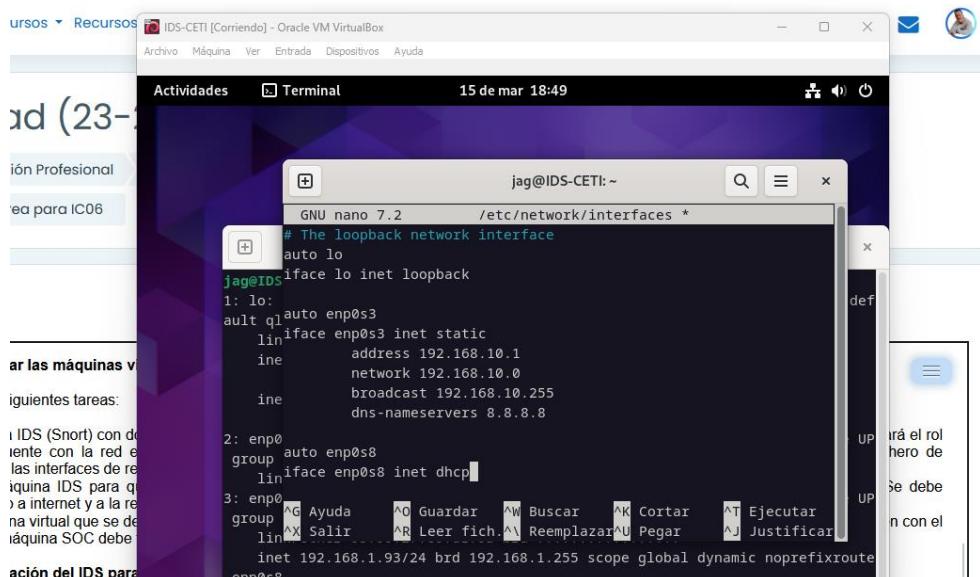
sudo nano /etc/network/interfaces



Añadimos las nuevas configuraciones al fichero, atendiendo a que hemos creado la red Interna en el adaptador 1 y el Puente en el adaptador 2, y guardamos.

```
auto enp0s3
iface enp0s3 inet static
    address 192.168.10.1
    network 192.168.10.0
    netmask 255.255.255.0
    broadcast 192.168.10.255
    dns-nameservers 8.8.8.8

auto enp0s8
iface enp0s8 inet dhcp
```



Hacemos un **ifconfig** o un **ip a** para comprobar.

```
jag@IDS-CETI:~$ 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 brd 00:00:00:00:00:00 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 brd 00:00:00:00:00:00 scope host
                valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group JUERADE
    qlen 1000
        link/ether 08:00:27:9a:2d:5f brd ff:ff:ff:ff:ff:ff
        inet 192.168.10.2/24 brd 192.168.10.255 scope global enp0s3
            valid_lft forever preferred_lft forever
            inet6 fe80::a00:27ff:fe9a:2d5f/64 scope link
                valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group JUERADE
    qlen 1000
        link/ether 08:00:27:c0:d1:a0 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.93/24 brd 192.168.1.255 scope global dynamic enp0s8
            valid_lft 36741sec preferred_lft 36741sec
            inet6 fe80::a00:27ff:fe0d:a10a/64 scope link
                valid_lft forever preferred_lft forever
jag@IDS-CETI:~$
```

Creamos ahora un servicio de IPTABLES, dejando habilitado de forma permanente el bit de forward para que el firewall se mantenga activo tras cada reinicio de la máquina.

Le indicamos que debe aplicarlo al instante, añadimos la regla (o reglas) y comprobamos.

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -p
sudo iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
```

```
jag@IDS-CETI:/etc/network$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] contraseña para jag:
net.ipv4.ip_forward = 1
jag@IDS-CETI:/etc/network$ sudo sysctl -p
jag@IDS-CETI:/etc/network$ sudo iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
Chain PREROUTING (policy ACCEPT)
target     prot opt source          destination
Chain INPUT (policy ACCEPT)
target     prot opt source          destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source          destination
Chain POSTROUTING (policy ACCEPT)
target     prot opt source          destination
MASQUERADE all  --  anywhere       anywhere
MASQUERADE all  --  anywhere       anywhere
jag@IDS-CETI:/etc/network$
```

Vamos al SOC y tras hacer un **ifconfig** para comprobar la configuración actual, revisamos las rutas.

Añadimos una nueva, para que se comunique el webserver (IP **192.168.10.0/24**) a través de la máquina IDS (IP **192.168.1.93**)

`sudo ip route add 192.168.10.0/24 via 192.168.1.93`

```

kali@kali:~$ ip route
default via 192.168.1.1 dev eth0 proto dhcp src 192.168.1.85 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.85 metric 100

(kali㉿kali)-[~]
$ sudo ip route add 192.168.10.0/24 via 192.168.1.93
[sudo] contraseña para kali:
Lo siento, prueba otra vez.
[sudo] contraseña para kali:

(kali㉿kali)-[~]
$ ip route
default via 192.168.1.1 dev eth0 proto dhcp src 192.168.1.85 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.85 metric 100
192.168.10.0/24 via 192.168.1.93 dev eth0

```

Comprobamos la correcta comunicación haciendo varios ping hacia la red interna y al exterior.

```

(kali㉿kali)-[~]
$ ping 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.786 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=64 time=0.880 ms
^C
--- 192.168.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1064ms
rtt min/avg/max/mdev = 0.786/0.833/0.880/0.047 ms
2: ouf
(kali㉿kali)-[~]
$ ping 192.168.10.100
PING 192.168.10.100 (192.168.10.100) 56(84) bytes of data.
64 bytes from 192.168.10.100: icmp_seq=1 ttl=63 time=0.934 ms
64 bytes from 192.168.10.100: icmp_seq=2 ttl=63 time=2.99 ms
64 bytes from 192.168.10.100: icmp_seq=3 ttl=63 time=2.28 ms
^C
--- 192.168.10.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.934/2.068/2.992/0.853 ms
3: ouf
(kali㉿kali)-[~]
$ ping google.com
PING google.com (142.250.184.174) 56(84) bytes of data.
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=1 ttl=116 time=9.86 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=2 ttl=116 time=9.44 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1041ms
rtt min/avg/max/mdev = 9.436/9.646/9.856/0.210 ms
(kali㉿kali)-[~]
$ ping 192.168.1.93
PING 192.168.1.93 (192.168.1.93) 56(84) bytes of data.
64 bytes from 192.168.1.93: icmp_seq=1 ttl=64 time=0.620 ms
64 bytes from 192.168.1.93: icmp_seq=2 ttl=64 time=1.80 ms
^C
--- 192.168.1.93 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1025ms
rtt min/avg/max/mdev = 0.620/1.211/1.802/0.591 ms

```

Hacemos lo mismo desde el webserver y comprobamos que en ambos casos es correcta.

```

jag@IDS-CETI:~$ ping 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.747 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=64 time=1.09 ms
...
--- 192.168.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1022ms
rtt min/avg/max/mdev = 0.747/0.916/1.086/0.169 ms
jag@IDS-CETI:~$ ping google.com
PING google.com (142.250.168.174) 56(84) bytes of data.
64 bytes from madd07c3-in-f14.1e100.net (142.250.168.174): icmp_seq=1 ttl=115 time=11.3 ms
64 bytes from madd07c3-in-f14.1e100.net (142.250.168.174): icmp_seq=2 ttl=115 time=11.5 ms
...
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 11.291/11.406/11.522/0.115 ms
jag@IDS-CETI:~$ ping 192.168.1.93
PING 192.168.1.93 (192.168.1.93) 56(84) bytes of data.
64 bytes from 192.168.1.93: icmp_seq=1 ttl=64 time=0.563 ms
64 bytes from 192.168.1.93: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 192.168.1.93: icmp_seq=3 ttl=64 time=1.14 ms
...
--- 192.168.1.93 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 0.563/0.919/1.140/0.254 ms
jag@IDS-CETI:~$ ping 192.168.1.85
PING 192.168.1.85 (192.168.1.85) 56(84) bytes of data.
64 bytes from 192.168.1.85: icmp_seq=1 ttl=63 time=1.32 ms
64 bytes from 192.168.1.85: icmp_seq=2 ttl=63 time=3.43 ms
64 bytes from 192.168.1.85: icmp_seq=3 ttl=63 time=0.845 ms
...
--- 192.168.1.85 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.845/1.863/3.425/1.121 ms
jag@IDS-CETI:~$ ping 192.168.1.91
PING 192.168.1.91 (192.168.1.91) 56(84) bytes of data.
64 bytes from 192.168.1.91: icmp_seq=1 ttl=64 time=0.620 ms
64 bytes from 192.168.1.91: icmp_seq=2 ttl=64 time=1.00 ms
...

```

Hechas las comprobaciones, podemos pasar al siguiente apartado para instalar **Snort**.

✓ Apartado 2: Configuración del IDS para registrar el tráfico de red. (SNORT)

Deberás efectuar las siguientes tareas:

- Instalar y configurar SNORT en el IDS para poder escuchar y guardar todo el tráfico de la red DMZ.

Comenzamos la instalación de **Snort**. Vamos a evitar primero problemas comunes de esta instalación con las fuentes añadiendo estos enlaces.

`sudo nano /etc/apt/sources.list`

```

jag@IDS-CETI:~$ sudo nano /etc/apt/sources.list
#deb cdrom:[Debian GNU/Linux 12.5.0 _Bookworm_ - Official amd64 DVD Binary-1 with Firmware 2]>
deb http://deb.debian.org/debian/ bookworm main non-free-firmware
deb-src http://deb.debian.org/debian/ bookworm main non-free-firmware

deb http://security.debian.org/debian-security bookworm-security main non-free-firmware
deb-src http://security.debian.org/debian-security bookworm-security main non-free-firmware

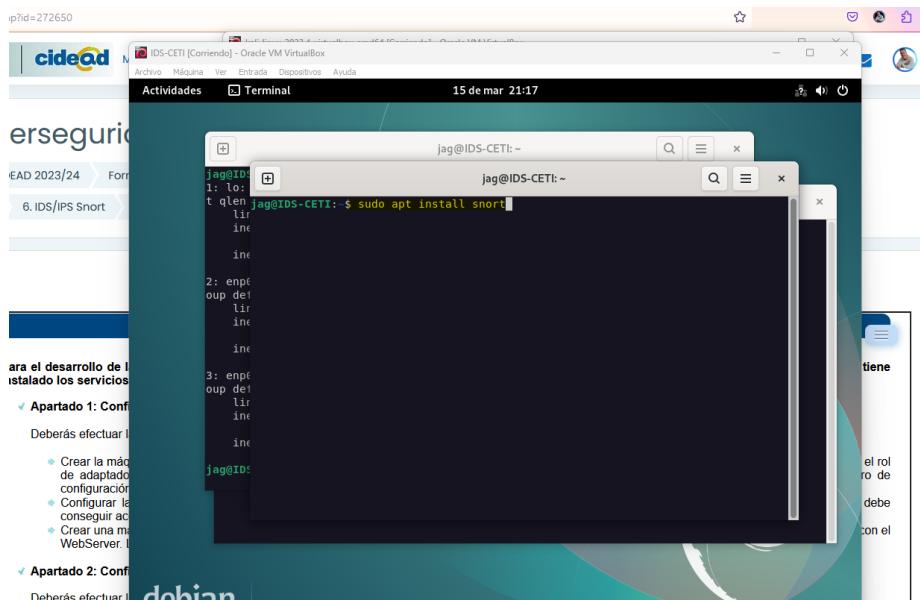
deb http://deb.debian.org/debian/ bookworm-updates main non-free-firmware
deb-src http://deb.debian.org/debian/ bookworm-updates main non-free-firmware

deb http://ftp.de.debian.org/debian sid main

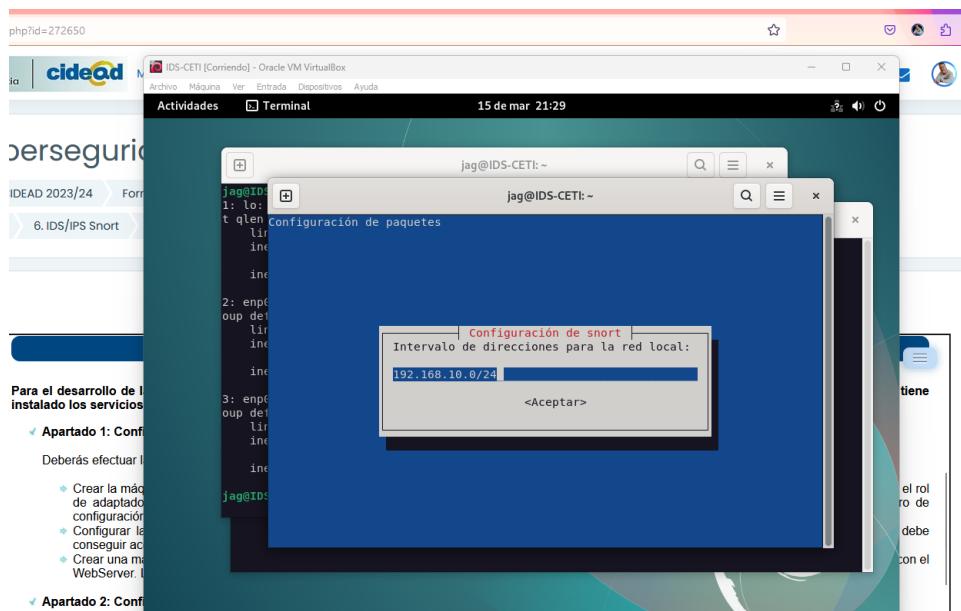
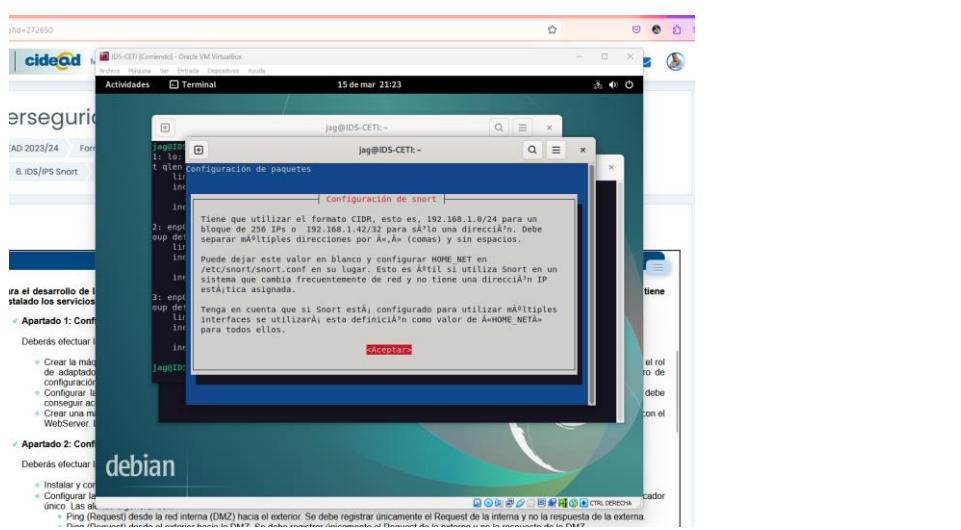
```

Tras esto, podemos iniciar la instalación en nuestra máquina **IDS** de la aplicación **Snort**.

sudo apt install snort



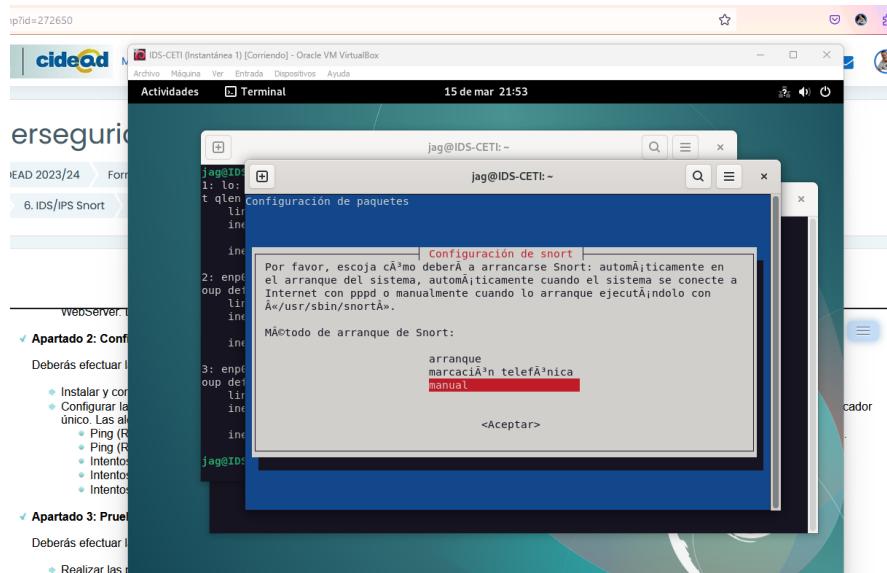
Durante la instalación, nos pedirá indicarle la red en la que estará ubicado.



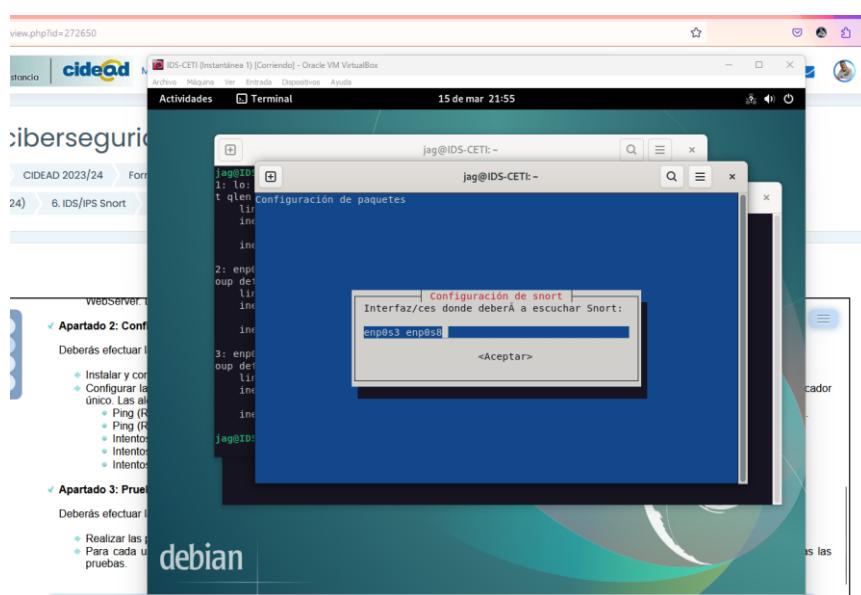
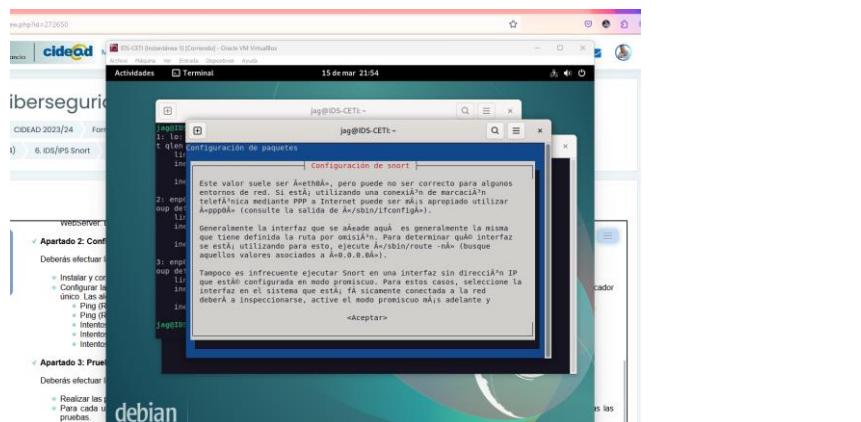
Instalado, continuamos con su configuración.

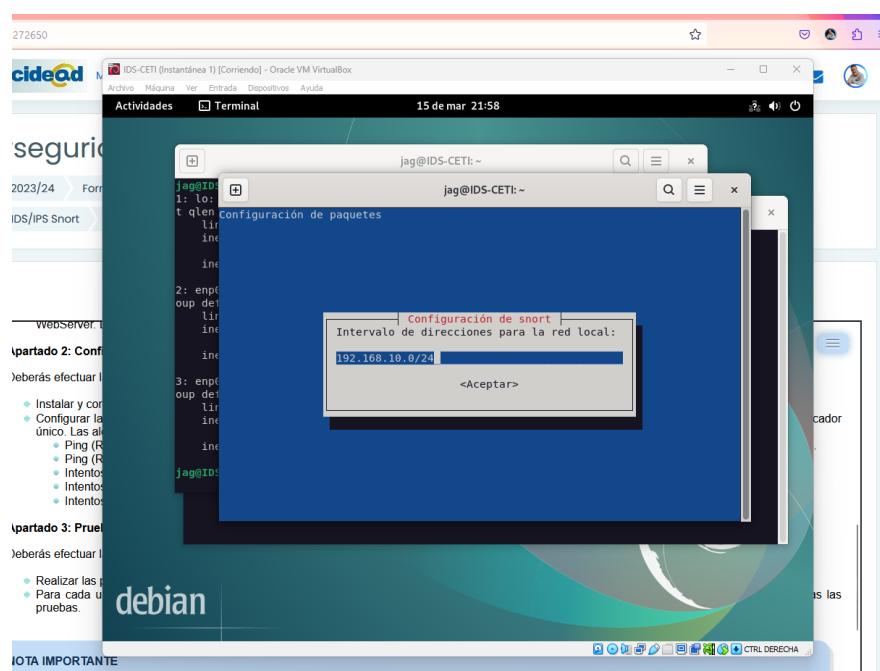
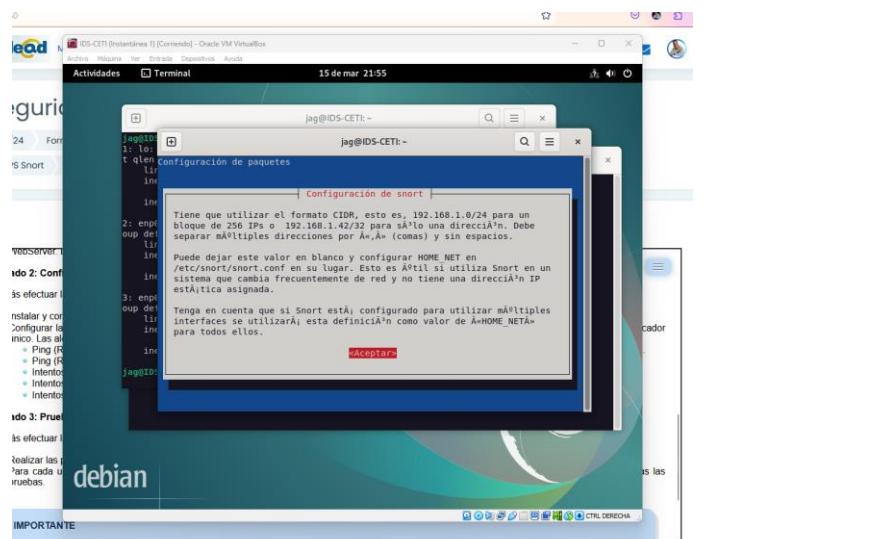
sudo dpkg-reconfigure snort

Lo primero es indicarle que queremos controlar su funcionamiento, dejando el arranque en **manual**.

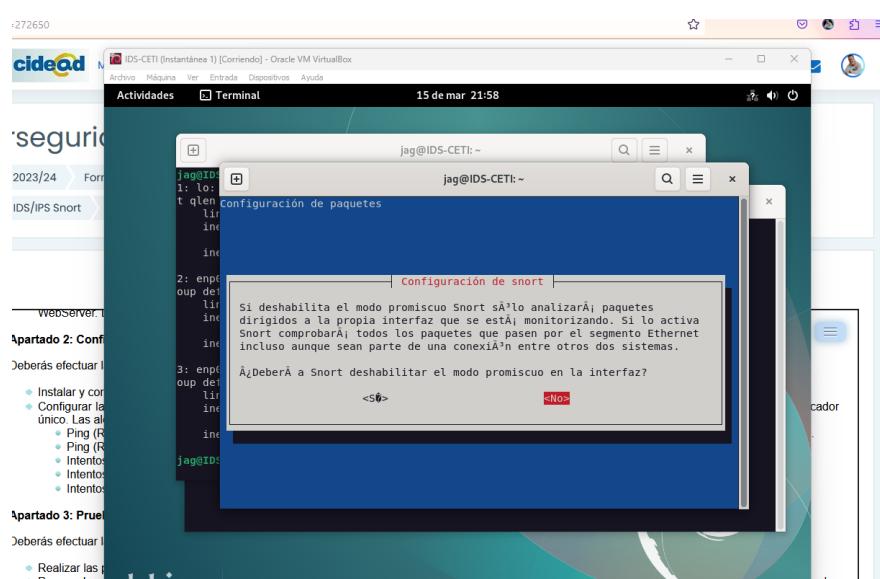


El siguiente apartado es para saber desde que interfaz y red escucha.





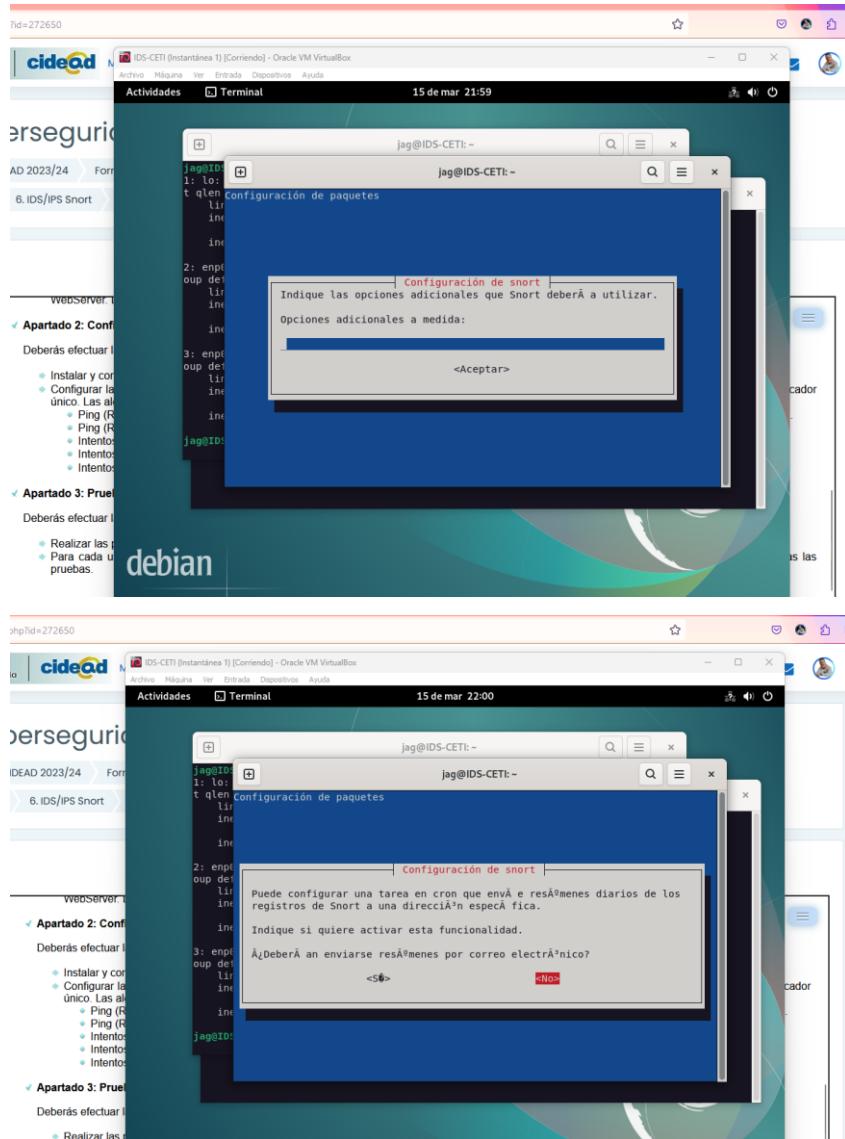
Deshabilitamos el modo promiscuo.



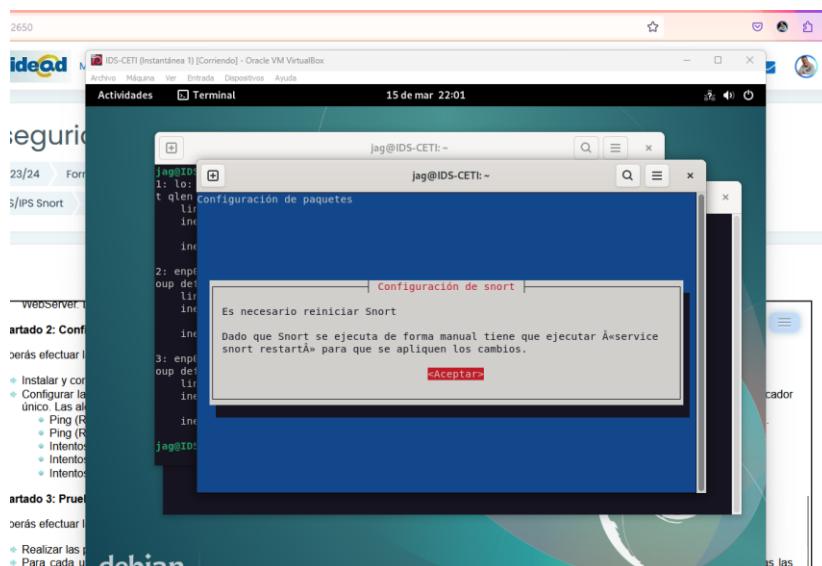
Incidentes de Ciberseguridad

Tarea Online UD06.

Dejamos en blando las opciones adicionales e indicamos que no queremos el envío de correo.



Para finalizar, necesitamos reiniciar el servicio.



- Configurar las reglas de detección de Snort, cada una de ellas debe recoger un mensaje indicando el tipo de conexión que se establece y un identificador único. Las alertas a generar son:

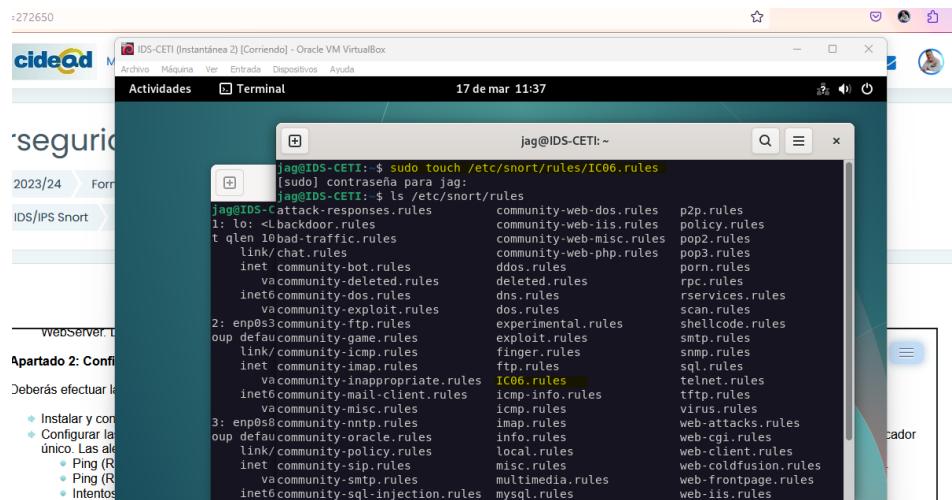
- Ping (Request) desde la red interna (DMZ) hacia el exterior. Se debe registrar únicamente el Request de la interna y no la respuesta de la externa.

Antes de crear las reglas, y a efectos de que puedan ser controladas de manera más sencilla, creamos un archivo donde alojaremos las reglas de esta tarea.

```
sudo touch /etc/snort/rules/IC06.rules
```

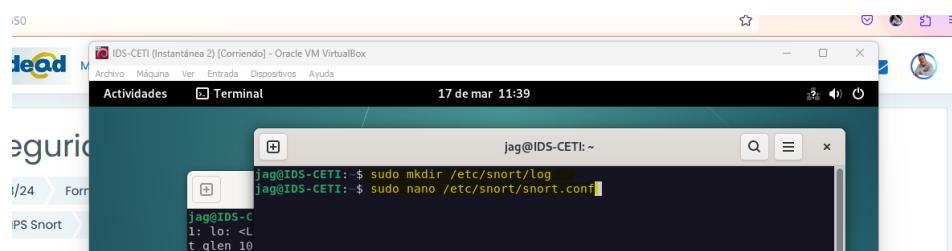
Y podremos ver que está disponible junto a los demás generados por defecto junto con la instalación de este IDS.

```
ls /etc/snort/rules
```



Con el archivo creado, vamos a configurar algunas variables para trabajar de forma más automática.

```
sudo nano /etc/snort/snort.conf
```



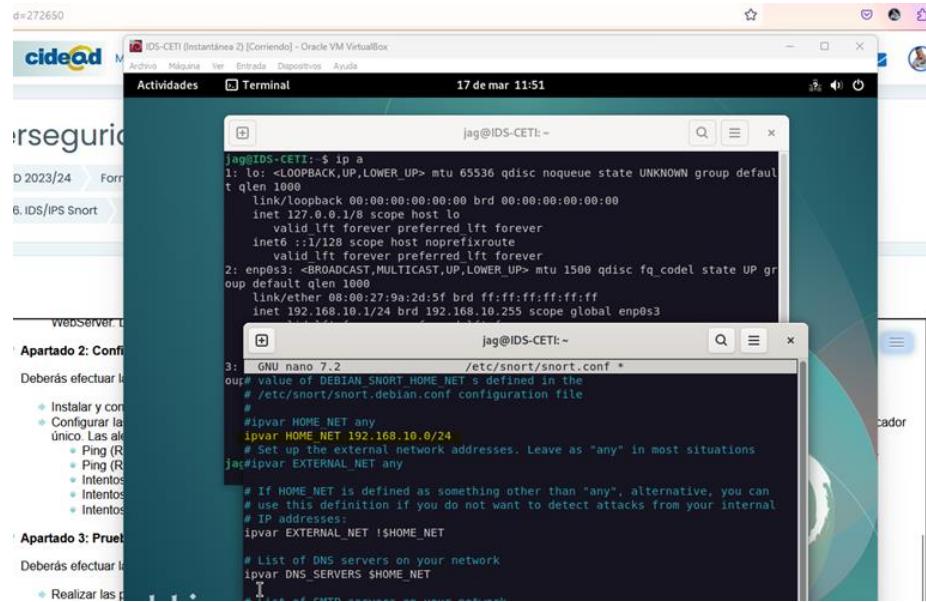
Para la red interna

```
ipvar HOME_NET 192.168.10.0/24
```

Para la red externa, le diremos que sea cualquier IP que no sea la interna.

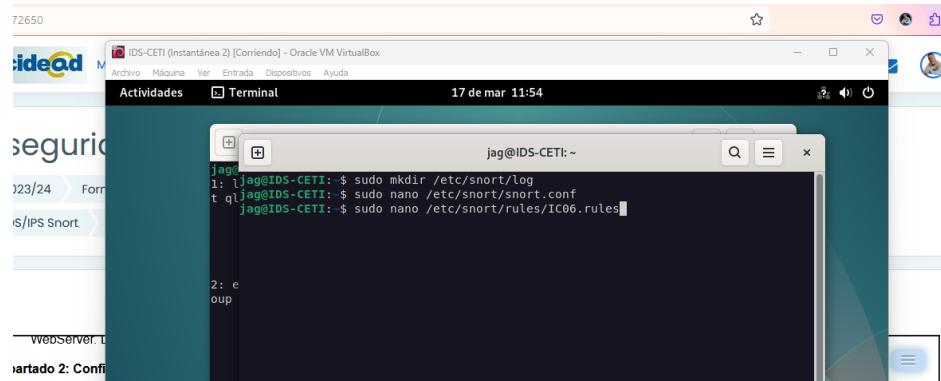
```
ipvar EXTERNAL_NET !$HOME_NET
```

Podemos desde aquí configurar, además de estas dos que suelen ser las más utilizadas, las variables que creamos que puedan ser de utilidad por utilizarlas con asiduidad.



Creamos ahora la primera regla, desde el archivo creado antes:

```
sudo nano /etc/snort/rules/IC06.rules
```



Utilizamos las variables creadas para capturar desde la red interna a la externa por cualquier puerto, a través de un eco (**itype:8**)

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"Ping (Request) desde DMZ hacia el exterior"; sid:10001; itype:8;)
```

- Ping (Request) desde el exterior hacia la DMZ. Se debe registrar únicamente el Request de la externa y no la respuesta de la DMZ.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Ping (Request) desde el exterior hacia DMZ"; sid:10002; itype:8;)
```

- Intentos de las conexiones SSH hacia WebServer. Solamente registra el primer paquete de sincronización de este intento de conexión.

Recogemos todas las solicitudes que vayan a la IP de nuestro webserver por su puerto 22, recogiendo sólo el primer paquete con **flags:S**.

```
alert tcp any any -> 192.168.10.100 22 (msg:"Intentos de conexion SSH a webserver"; flags:S; sid:10003;)
```

- Intentos de las conexiones **HTTP** hacia **WebServer**. Solamente registra el primer paquete de sincronización de este intento de conexión.

Similar al caso anterior, pero para **HTTP** recogemos el puerto **80**.

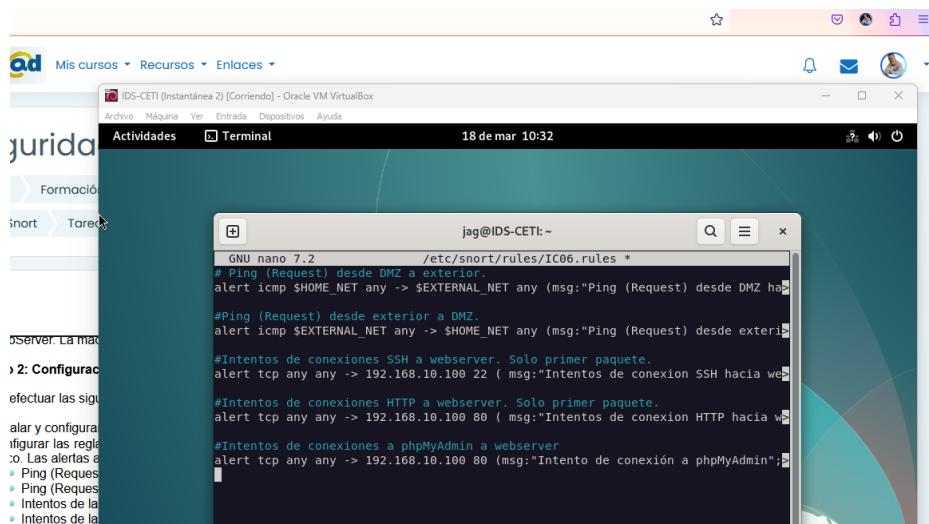
```
alert tcp any any -> 192.168.10.100 80 (msg:"Intentos de conexión HTTP hacia WebServer"; flags:S; sid:10004;)
```

- Intentos de las conexiones a **phpMyAdmin** hacia **WebServer**. La ruta hacia la base de datos es <http://ip-de-WebServer/phpmyadmin>.

Necesitamos pasarle la ruta a la regla anterior:

```
alert tcp any any -> 192.168.10.100 80 (msg:"Intento de conexión a phpMyAdmin"; sid:10005; content:"/phpmyadmin";)
```

El archivo queda finalmente así:

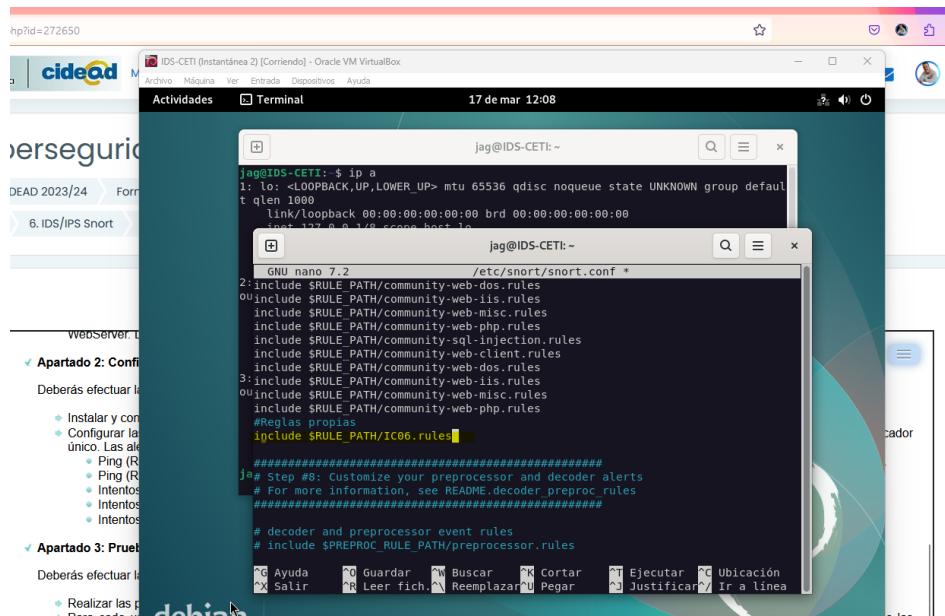


```
GNU nano 7.2 /etc/snort/rules/IC06.rules *
# Ping (Request) desde DMZ a exterior.
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"Ping (Request) desde DMZ ha>
#Ping (Request) desde exterior a DMZ.
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Ping (Request) desde exterior a DMZ";)
#Intentos de conexiones SSH a webserver. Solo primer paquete.
alert tcp any any -> 192.168.10.100 22 ( msg:"Intentos de conexión SSH hacia webser>
#Intentos de conexiones HTTP a webserver. Solo primer paquete.
alert tcp any any -> 192.168.10.100 80 ( msg:"Intentos de conexión HTTP hacia webser>
#Intentos de conexiones a phpMyAdmin a webserver
alert tcp any any -> 192.168.10.100 80 (msg:"Intento de conexión a phpMyAdmin"; content:>
```

Debemos ahora indicar en el archivo de configuración de **Snort**, que debe utilizarlas. Incluimos las reglas creadas al final de nuestro **snort.conf**

```
sudo nano /etc/snort/snort.conf
```

```
include $RULE_PATH/IC06.rules
```



```
jag@IDS-CETI:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
            inet6 ::1/128 brd :: scope host lo
                valid_lft forever preferred_lft forever
jag@IDS-CETI:~$ cat /etc/snort/snort.conf
# 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-client.rules
3: include $RULE_PATH/community-web-iis.rules
#include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/community-web-php.rules
#Reglas propias
#include $RULE_PATH/IC06.rules
#####
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README_decoder_preproc_rules
#####

# decoder and preprocessor event rules
# include $PREPROC_RULE_PATH/preprocessor.rules
```

✓ **Apartado 3: Pruebas de las alertas generadas.**

Deberás efectuar las siguientes tareas:

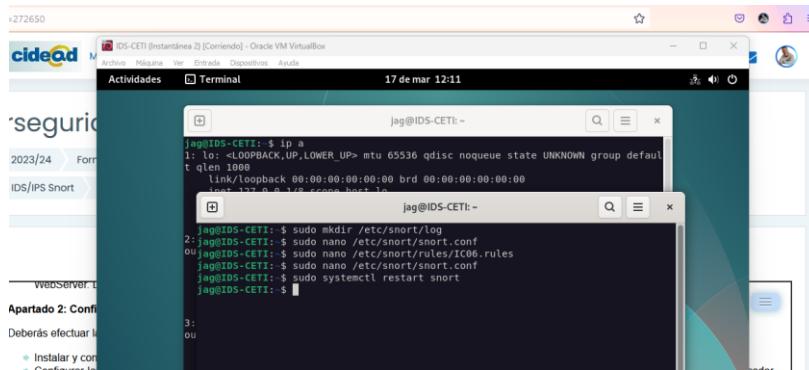
- Realizar las pruebas pertinentes donde se demuestren las diferentes alertas generadas en el apartado 2.

Reiniciamos snort para hacer las pruebas

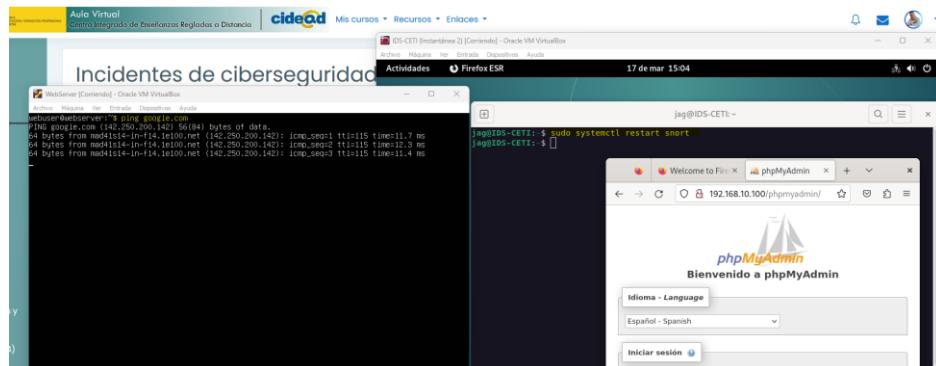
```
sudo systemctl restart snort
```

Si aparece algún error, podemos comprobar donde hemos fallado con

```
sudo systemctl status snort
```



Comprobamos lanzando un **ping** desde la consola de nuestro **webserver**, **ping** desde el **SOC** al **webserver** y abriendo **phpMyAdmin** en el navegador.

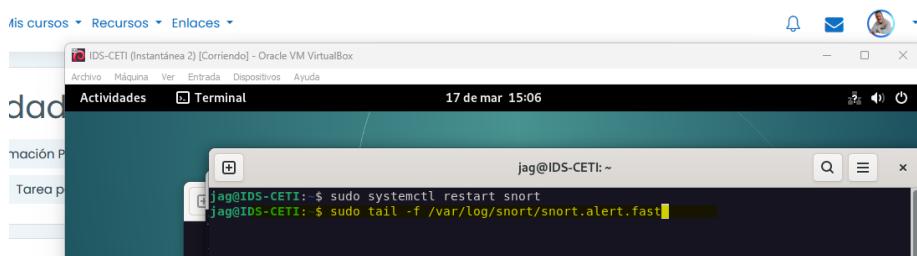


- Para cada una de estas alertas se debe recoger pantallazo con las acciones realizadas y un volcado final del archivo de log resultante tras todas las pruebas.

Como puede verse en la última captura, vamos a probar el funcionamiento mediante la captura del tráfico desde y hacia el **webserver**, además, de abrir desde el **SOC**, una ventana del navegador y una complementaria a **phpMyAdmin**.

Mostraremos los logs recogidos mediante

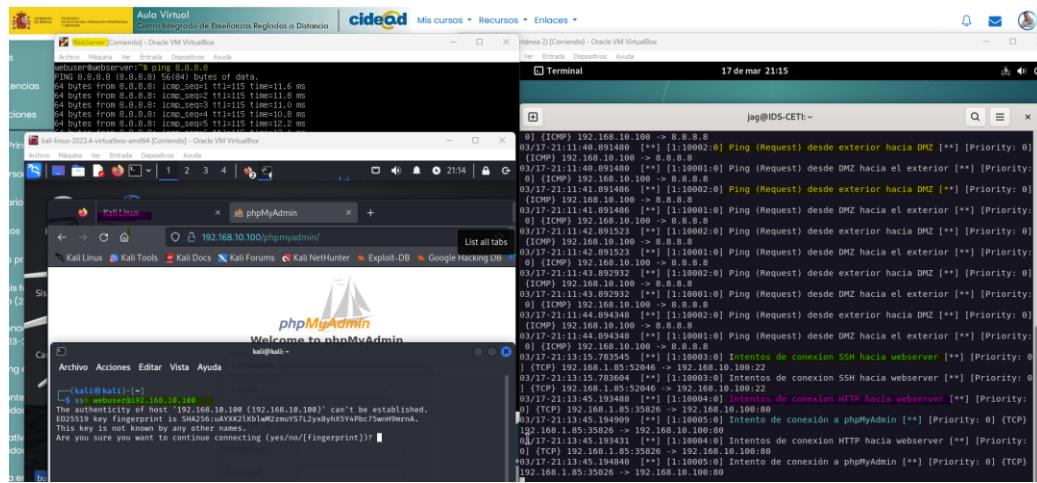
```
sudo tail -f /var/log/snort/snort.alert.fast
```



Incidentes de Ciberseguridad

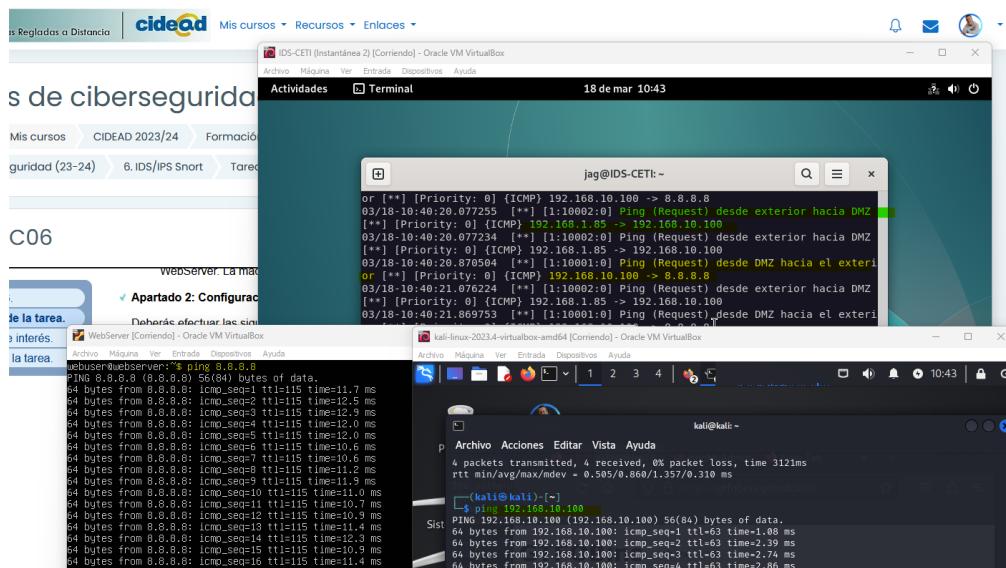
Tarea Online UD06.

Si ahora lanzamos un ping hacia la red desde el **webserver**, podremos ver como muestran los **logs** todos los mensajes configurados.

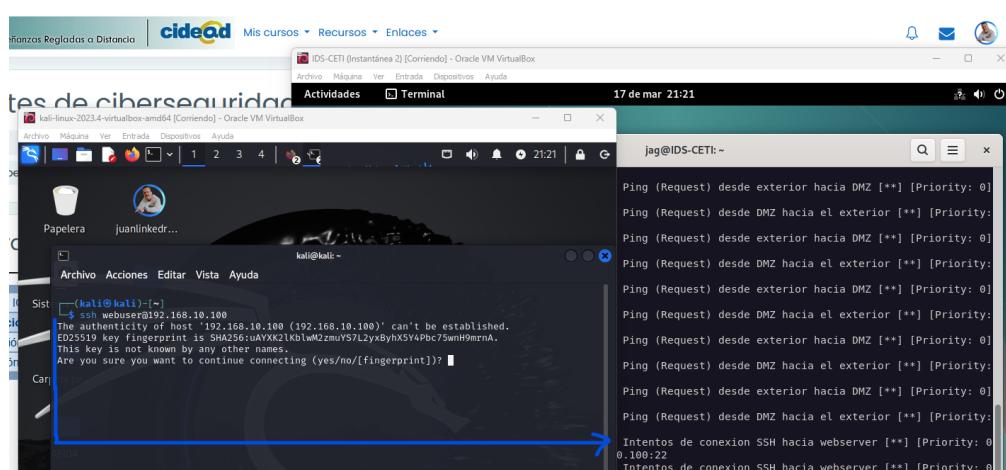


Aunque he resaltado con color los distintos tipos, vamos a verlos de forma más aislada para evitar dudas por la posible resolución de las capturas.

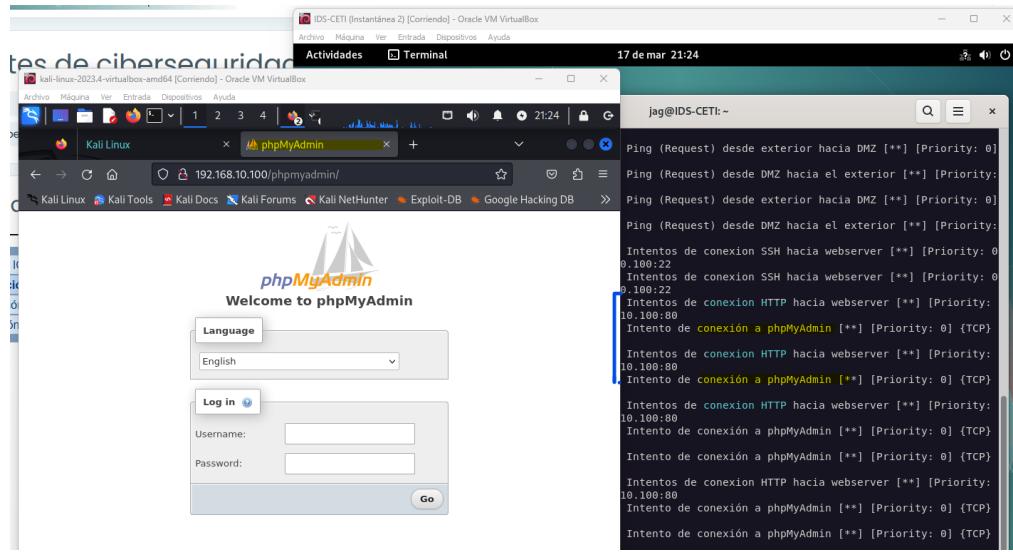
Captura de tráfico desde y hacia webserver:



Captura de tráfico SSH desde el soc.



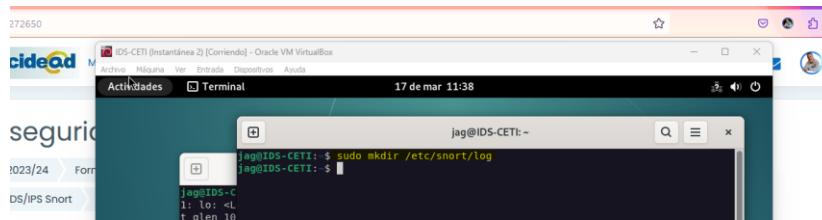
Captura de intentos de conexiones HTTP y phpMyAdmin.



Recordemos que antes, hemos creado un archivo para manejar nuestras reglas de forma independiente para esta tarea.

Podríamos hacer lo mismo con el registro de logs, creando un directorio propio para aislarlos

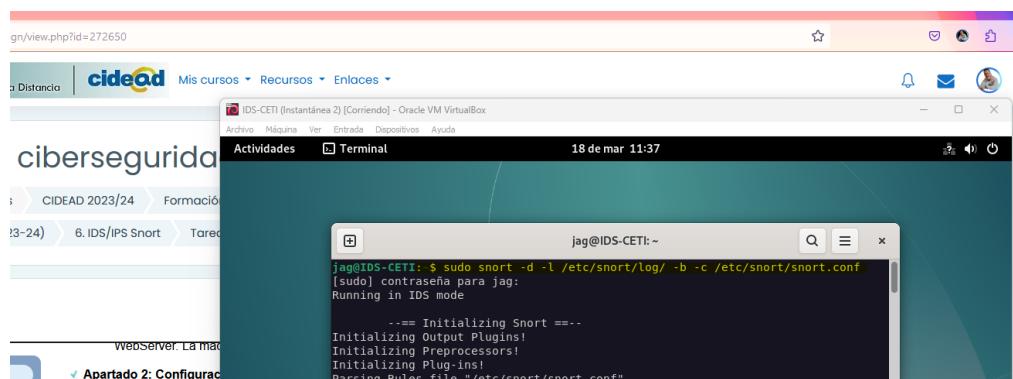
```
sudo mkdir /etc/snort/log
```



Configuraríamos las variables y reglas como hemos hecho antes (no necesita de nada especial).

A la hora de reiniciar snort, primero le indicaremos la variación de nuestro proceso con:

```
sudo snort -d -l /etc/snort/log/ -b -c /etc/snort/snort.conf
```

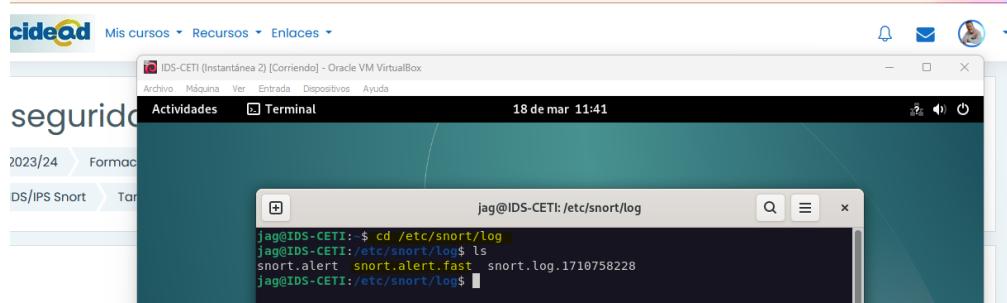


Y arrancamos de nuevo para hacer las comprobaciones:

```
sudo systemctl restart snort
```

Nos habrá creado en el nuevo directorio los mismos archivos binarios y de texto que en la carpeta original de logs. Por tanto, podremos comprobar los registros desde:

```
sudo cat /etc/snort/log/snort.alert.fast
```



Ambos archivos, podemos descargarlos o abrirlos aparte para su comprobación y análisis.

Webgrafía.

<https://www.youtube.com/watch?v=MwqdKevOas0>

<https://seguridadyredes.wordpress.com/2008/02/20/analisis-capturas-trafico-de-red-interpretacion-trafico-icmp-i/>

<https://seguridadyredes.wordpress.com/2008/11/13/snort-opciones-reglas-no-relacionadas-con-el-contenido-ii-parte/>